GUIDEd Agents: Enhancing Navigation Policies through Task-Specific Uncertainty Abstraction in Localization-Limited Environments

Anonymous authors

***Abstract*— Autonomous robots performing navigation tasks in complex environments often face significant challenges due to uncertainty in state estimation. In settings where the robot faces significant resource constraints and accessing high-precision localization comes at a cost, the planner may have to rely primarily on less precise state estimates. Our key observation is that different tasks, and different portions of tasks require varying levels of precision in different regions: a robot navigating a crowded space might need precise localization near obstacles but can operate effectively with less precise state estimates in open areas. In this paper, we present a planning method for integrating task-specific uncertainty requirements directly into navigation policies. We introduce Task-Specific Uncertainty Maps (TSUMs), which abstract the acceptable levels of state estimation uncertainty across different regions. TSUMs align task requirements and environmental features using a shared representation space, generated via a domain-adapted encoder. Using TSUMs, we propose Generalized Uncertainty Integration for Decision-Making and Execution (GUIDE), a policy-conditioning framework that incorporates these uncertainty requirements into the robot's decision-making process, enabling the robot to reason about the context-dependent value of certainty and adapt its behavior accordingly. We show how integrating GUIDE into reinforcement learning frameworks allows the agent to learn navigation policies that effectively balance task completion and uncertainty management without the need for explicit reward engineering. We evaluate GUIDE on a variety of real-world robotic navigation tasks and find that it demonstrates significant improvement in task completion rates compared to baseline methods that do not explicitly consider task-specific uncertainty. Experiments, code, dataset available at: https://guided-agents.github.io.**

I. Introduction

In complex environments where robots must balance task completion with resource usage, managing uncertainty in state estimation becomes a critical challenge. Consider an autonomous surface vehicle (ASV) conducting a mapping mission in stealth-critical settings where each GPS fix risks detection, or a ground robot operating in environments where high-precision localization requires costly computation. In such resource-constrained settings, the robot must not only complete its assigned task but also carefully manage when and where to use expensive localization operations. Importantly, constantly striving to reduce uncertainty through high-precision localization is neither necessary nor efficient for successful task execution. When operating in regions critical to the task objectives or near obstacles, precise localization becomes crucial for mission success. However, in areas less relevant to the immediate task goals, the robot can often tolerate higher positional uncertainty, allowing it to focus on efficient task execution rather than achieving pinpoint accuracy. This scenario underscores a fundamental observation in robotic navigation with limited access to high-precision localization: the acceptable level of uncertainty at any location is inherently *task-specific* and *context-dependent*.

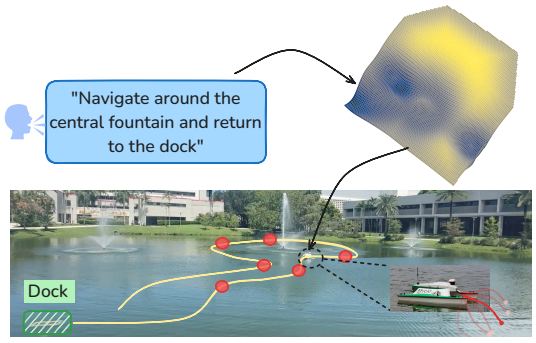


Fig. 1: GUIDEd Agents in action. Top Left: The ASV is assigned a navigation task. Top Right: GUIDE interprets the task and generates a representation highlighting areas where the ASV needs higher positional certainty (dark blue). Bottom: The policy executed by the ASV. The white line represents the trajectory taken by the ASV, the red dots indicate locations where the ASV actively reduces its state estimation uncertainty to satisfy task-specific requirements.

Traditional approaches to navigation in localization-limited environments often fall into two categories: minimizing uncertainty universally through frequent use of expensive localization [1], [2] or enforcing fixed uncertainty thresholds across the environment [3], [4]. While these strategies may be effective in settings where localization resources are abundant, they become impractical when high-precision state estimation comes at a cost. Existing methods fail to account for how the value of precise localization varies across different phases of the mission. Adapting these methods to localization-limited environments would require extensive reward engineering to capture the complex relationship between task objectives and localization costs.

The key issue lies in the disconnect between task requirements and resource-constrained uncertainty management. Current frameworks treat localization decisions independently from task specifications, leading to either excessive resource consumption or compromised task performance when different regions demand different levels of certainty.

Statement of contributions: (i) TSUM: With the aim of grounding high-level task descriptions and low-level uncertainty handling, we propose the concept of a ***T**ask-**S**pecific **U**ncertainty **M**ap (TSUM)*. A TSUM serves as an intermediate abstraction that represents the acceptable levels of state estimation uncertainty across different regions of the environment for a specific task. We present a CLIP-

based approach for computing TSUMs that effectively aligns visual environmental features with textual task specifications in a shared embedding space. (ii) GUIDE: Building upon TSUM, we introduce ***G**eneralized **U**ncertainty **I**ntegration for **D**ecision-Making and **E**xecution (GUIDE)*, a policy-conditioning framework that incorporates TSUMs into navigation policies. The core proposition of GUIDE is that by conditioning navigation policies on TSUMs, robots can better reason about the *context-dependent value of certainty*. (iii) GUIDEd agents: We demonstrate how GUIDE can be integrated into a reinforcement learning framework by adapting the Soft Actor-Critic algorithm, resulting in the *GUIDEd Soft Actor-Critic (G-SAC)* method. G-SAC learns navigation policies that effectively balance task completion and uncertainty management without the need for explicit reward engineering. (iv) Evaluation: We evaluate GUIDE in real-world in-the-wild deployment using an Autonomous Surface Vehicle (ASV) in the context of marine autonomy. Our results show significant improvements in task performance when compared to methods that do not explicitly consider task-specific uncertainty requirements. To spur further research, we also release the marine dataset and code.

II. Related Works

a) Uncertainty Modeling: Uncertainty management is a fundamental aspect of robot navigation [5]. Probabilistic techniques [6], [7] have been widely used to enable localization in stochastic environments. However, these approaches often treat uncertainty uniformly across the environment, without considering how different regions may require varying levels of certainty depending on the task. Belief-space planners framed as Partially Observable Markov Decision Processes (POMDPs) further couple estimation with action by optimizing over the distribution, explicitly balancing task progress against the value of information-gathering [8], [9]. While POMDP approximations improve scalability, they cannot capture location- or context-dependent tolerance.

b) RL for Navigation: Reinforcement Learning (RL) has been successfully applied to robotic navigation tasks, enabling agents to learn navigation policies [4], [10]. While these methods can learn effective policies in controlled environments, they often perform suboptimally in real-world scenarios due to factors like partial observability and dynamic changes [11], [12]. Recent work has focused on developing navigation policies conditioned on specific goals [13], [14] capable of handling objectives, adjusting the robot's behavior according to the task. These approaches often neglect the varying importance of uncertainty management across different tasks and environments.

c) Uncertainty-Aware RL: Incorporating uncertainty into RL for navigation has been investigated to enhance exploration, robustness, and safety [10]. Some approaches introduce uncertainty penalization into the reward function [15], [16]. Others utilize Bayesian RL methods to model uncertainty in value estimation [17], [18]. Bootstrapped ensembles maintain multiple value functions to capture epistemic uncertainty, leading to informed exploration strategies [19], [20]. While these methods consider uncertainty, they often do so globally [3] across the environment and do not tailor to specific navigation tasks or spatial regions. This uniform treatment of uncertainty becomes particularly problematic in resource-constrained settings, lacking mechanisms to reason about when consuming these limited resources would provide the most value for task completion.

d) Risk-Aware Planning: Risk-aware planning introduces measures to balance performance and safety by considering the risks associated with different actions [21], [22]. In the context of RL, risk-sensitive approaches adjust the policy to avoid high-risk actions, often through modified reward functions or policy constraints [23], [24]. Although effective in managing risk, they apply uniform risk thresholds and do not adapt to the task-specific uncertainty requirements.

e) Task-Specific Navigation: Recent works proposed large pre-trained models to interpret complex instructions and generate corresponding navigation behaviors [25]–[27]. While these approaches enable robots to execute a wider range of tasks, they lack a systematic method for integrating task-specific uncertainty requirements into the navigation policy. This limitation reduces their effectiveness in localization-limited environments where the importance of uncertainty can vary across different regions.

In contrast to prior works, we place our research at the intersection of task-specific, resource-constrained navigation. GUIDE addresses a specific gap in the literature: *how to complete navigation tasks in environments where high-precision localization is a limited resource that must be strategically utilized*. While previous approaches either assume unlimited access to state estimation or treat uncertainty uniformly, GUIDE grounds task requirements, environmental features and uncertainty into the navigation policy.

III. The GUIDE Framework

Problem setting: Consider a robot operating in a state space $\mathcal{S}$, where each state $s \in \mathcal{S}$ encodes the robot's pose. The robot can execute control inputs from a action space $\mathcal{A}$. Operating in a localization-limited environment [2], the robot's default state estimate is subject to uncertainty (e.g., communication constraints or environmental factors). The robot can access high-precision state estimates at significant costs—for instance, a robot conducting stealth-critical missions where each GPS fix risks detection.

We assume the availability of a visual representation of the operating environment *a priori*, such as an overhead map. This assumption is justified in many real-world applications where geographic information systems provide knowledge of the environment (e.g., static obstacles). A *navigation task* τ, specified in natural language, describes the robot's objectives and constraints with respect to environmental features.

Objective: Synthesize a navigation policy $\pi(a \mid s)$ that enables task completion while explicitly reasoning about the robot's state-estimation uncertainty $u(s)$. The policy must balance two competing demands: efficient task execution and strategic uncertainty management.

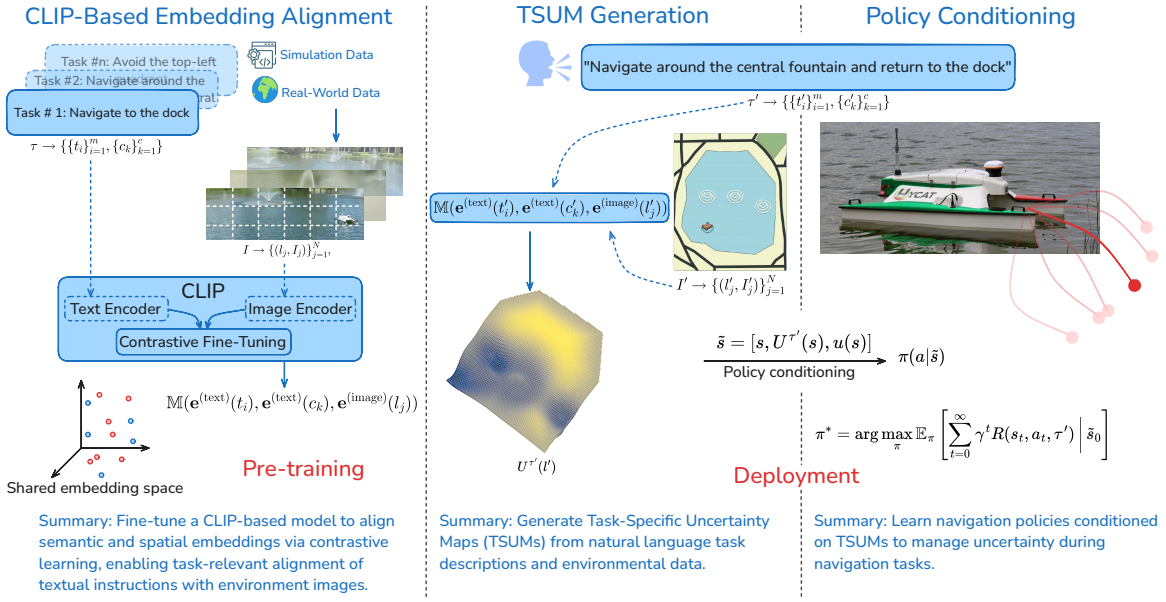


Fig. 2: GUIDE consists of two phases. (i) Pre-training: semantic and spatial embeddings are fine-tuned and aligned using a CLIP-based model. (ii) Deployment: TSUMs are generated from task descriptions and policies are conditioned on the TSUMs to manage uncertainty.

GUIDE aims to operationalize this balance by conditioning the navigation policy on a task-specific uncertainty abstraction, enabling the robot to reason *when* and *where* to invest in costly localization. Fig. 2 presents an overview of GUIDE. We now detail the specifics of these components.

A. Task-Specific Uncertainty Map (TSUM)

To effectively integrate task-specific uncertainty considerations into robotic navigation policies, we introduce the abstraction of *Task-Specific Uncertainty Maps* (TSUMs). A TSUM serves as an intermediate representation that bridges the gap between high-level task specifications and low-level uncertainty management decisions. Formally, a TSUM is a function $U^\tau : L \to \mathbb{R}^+$ that assigns an acceptable level of state estimation uncertainty to each location $l \in L$ for a given navigation task τ. The TSUM $U^\tau(l)$ is defined as:

$$U^\tau(l) = w_\Phi \Phi^\tau(l) + w_\mathcal{C} \mathcal{C}^\tau(l) + w_\mathcal{E} \mathcal{E}(l), \tag{1}$$

where $\Phi^\tau(l)$ quantifies the task-specific importance of location l, $\mathcal{C}^\tau(l)$ represents task constraints at l, and $\mathcal{E}(l)$ captures environmental factors. The weights w_Φ, $w_\mathcal{C}$, and $w_\mathcal{E}$ control the relative influence of each component, allowing us to balance different aspects of the task and environment. These weights are determined through validation on a held-out dataset of annotated scenarios.

1) CLIP-Based TSUM Generation: To generate TSUMs that effectively capture the relationship between visual features and task requirements, we propose a vision-language approach using the CLIP (Contrastive Language-Image Pre-training) model [28]. GUIDE processes both visual and textual inputs to produce a spatially-varying uncertainty map.

a) Pre-processing: (i) Environment representation: As established in Section III, we assume access to a visual representation of the environment. To process this spatial information systematically, we discretize the continuous space L into a grid of N image patches $\{I_1, I_2, \ldots, I_N\}$, where each patch I_j corresponds to a discrete spatial region $l_j \subset L$. The discretization parameters are chosen to balance computational efficiency with feature preservation. *(ii) Task Processing:* Given a task τ specified in natural language, we employ a hierarchical parsing approach to extract structured task representations. We utilize a dependency parser with a context-free grammar [29] to decompose τ into atomic statements that capture objectives and their associated constraints. These atomic statements are organized into m specifications $\{t_i\}_{i=1}^m$ and c auxiliary specifications $\{c_k\}_{k=1}^c$.

b) CLIP Encoding: We leverage CLIP's vision-language architecture to project both spatial and semantic information into a shared representational space. The model consists of two parallel encoders:

$$E_{\text{text}} : \mathcal{T} \to \mathbb{R}^d, \quad E_{\text{image}} : \mathcal{I} \to \mathbb{R}^d \tag{2}$$

where $\mathcal{T}$ and $\mathcal{I}$ denote the spaces of tokenized text and normalized image patches respectively, and d is chosen to optimize the balance between representational capacity and computational efficiency. These encoders employ self-attention mechanisms and cross-modal transformers to generate context-aware embeddings $\mathbf{e}^{(\text{text})}(t_i), \mathbf{e}^{(\text{text})}(c_k) \in \mathbb{R}^d$ for textual components and $\mathbf{e}^{(\text{image})}(l_j) \in \mathbb{R}^d$ for spatial regions.

c) Cross-Modal Alignment: The shared embedding space facilitates direct semantic comparison between specifications and visual features through cosine similarity:

$$\text{sim}(t, l) = \text{cosine}\big(\mathbf{e}^{(\text{text})}(t),\ \mathbf{e}^{(\text{image})}(l)\big)$$

This metric quantifies the semantic relevance between spatial regions and task components, enabling the construction of spatially-varying relevance maps. While pre-trained CLIP provides general-purpose understanding, domain-specific applications would require additional fine-tuning (Sec. III-A.3).

2) Computing Task Relevance and Constraints: Having established our CLIP-based pipeline, we now detail how we compute each component of the TSUM. Using the encoder functions defined in Equation (2), we obtain embeddings $(\mathbf{e}^{(\text{text})}(t_i), \mathbf{e}^{(\text{text})}(c_k) \in \mathbb{R}^d$ for text and $\mathbf{e}^{(\text{image})}(l_j) \in \mathbb{R}^d$ for image patches) that capture the semantic features of both textual descriptions and visual characteristics in a shared d-dimensional space. This shared representation allows us to meaningfully compare how well visual features at each location align with the requirements expressed in the task.

a) Task Relevance Function $\Phi^\tau(l)$: To compute the task relevance at each location, we first calculate similarity scores between each subtask and location

$$\rho_{t_i}(l_j) = \text{cosine}\big(\mathbf{e}^{(\text{text})}(t_i),\ \mathbf{e}^{(\text{image})}(l_j)\big). \tag{3}$$

These raw similarities are converted to attention weights $\alpha_i(l_j) = \exp\big(\rho_{t_i}(l_j)\big)/\sum_{i'=1}^{m} \exp\big(\rho_{t_{i'}}(l_j)\big)$. The task relevance function $\Phi^\tau(l_j)$ is then computed as a weighted sum

$$\Phi^\tau(l_j) = \sum_{i=1}^{m} \alpha_i(l_j)\, \rho_{t_i}(l_j).$$

This formulation creates a natural prioritization mechanism–locations that strongly match with important subtasks receive higher relevance scores. The attention weights α_i ensure that the most pertinent subtasks for each location dominate the relevance calculation, while the weighted sum combines evidence from all subtasks proportional to their importance.

b) Constraint Function $\mathcal{C}^\tau(l)$: For each constraint c_k,

$$\rho_{c_k}(l_j) = \text{cosine}\big(\mathbf{e}^{(\text{text})}(c_k),\ \mathbf{e}^{(\text{image})}(l_j)\big).$$

Constraint attention weights are computed as $\beta_k(l_j) = \exp\big(\rho_{c_k}(l_j)\big)/\sum_{k'=1}^{c} \exp\big(\rho_{c_{k'}}(l_j)\big)$. The final constraint function aggregates these similarities

$$\mathcal{C}^\tau(l_j) = \sum_{k=1}^{c} \beta_k(l_j)\, \rho_{c_k}(l_j).$$

Higher values of $\mathcal{C}^\tau(l_j)$ indicate locations where constraints are relevant and uncertainty management is critical.

c) Environmental Factors $\mathcal{E}(l)$: While CLIP embeddings capture visual features, certain navigation-relevant attributes may not be directly observable from imagery. We incorporate these factors through a vector of environmental features env_features(l_j), which encodes spatial properties. These features are mapped to a scalar value through

$$\mathcal{E}(l_j) = w_{\text{env}}^\top \text{env_features}(l_j) + b_{\text{env}},$$

where w_{env} and b_{env} are parameters optimized during training.

The final TSUM value at each location is computed by combining these components according to Equation (1). The complete pipeline is illustrated in Fig. 3.

3) Implementation Recipe: We now present the practical details of our domain-specific implementation.

a) Dataset: Our dataset consists of 2500 annotated overhead images (2000 images for training and 500 for validation) collected from real-world environments and high-fidelity simulations, covering a diverse range of navigation scenarios. Each image is discretized to 224 $\times$ 224 pixels at 0.5 meters per pixel resolution, chosen to align with CLIP's architecture. Each patch is annotated with $\mathcal{D} = \{(I_j, l_j, y_j^t, y_j^c, f_j)\}_{j=1}^N$, where I_j is the image patch, l_j represents location coordinates, y_j^t and y_j^c are binary relevance labels for a predefined set of subtasks and constraints, and f_j contains environmental features (e.g., bathymetry).

b) CLIP Fine-tuning: We adopt a selective fine-tuning approach to adapt the pre-trained CLIP (ViT-B/32) model while preserving its general vision-language understanding. Specifically, we freeze the first six transformer layers to maintain CLIP's foundational cross-modal understanding, while fine-tuning the remaining six layers to adapt to navigation-specific concepts. We optimize the loss function:

$$\mathcal{L}_{\text{CLIP}} = \mathcal{L}_{\text{cont.}} + \lambda_{\text{align}} \mathcal{L}_{\text{align}},$$

$\mathcal{L}_{\text{cont.}} = -\log(\exp(\text{sim}(t_i, l_i^+)/\tau)/\sum_j \exp(\text{sim}(t_i, l_j)/\tau))$ is the standard CLIP contrastive loss, and $\mathcal{L}_{\text{align}} = \sum_j \|y_j - \sigma(\text{sim}(t, l_j)/\zeta)\|^2$ is an alignment loss that encourages consistency between expert labels and embedding similarities. ζ is a temperature parameter set to 0.07, and σ is the sigmoid function. We freeze the first six transformer layers of CLIP and fine-tune the remaining layers for 10 epochs using AdamW optimizer with a learning rate of 1e-5.

c) Training Procedure: The training process involves:

(i) Fine-tune CLIP using $\mathcal{L}_{\text{CLIP}}$ as described above.

(ii) Train w_{env} and b_{env} by minimizing $\mathcal{L}_{\text{env}} = \sum_j \|\mathcal{E}(l_j) - \hat{e}_j\|^2$, where $\hat{e}_j$ represents ground truth annotations.

(iii) Optimize the weights w_Φ, $w_\mathcal{C}$, and $w_\mathcal{E}$ using a validation set of TSUMs $\mathcal{L}_{\text{TSUM}} = \sum_j \|U^\tau(l_j) - U^\tau_{\text{ref}}(l_j)\|^2$.

The reference TSUMs (U^τ_{ref}) were generated using a combination of automated and manual annotation processes. We utilized a high-fidelity Unity-based simulation environment to automatically generate initial uncertainty requirements based on task execution traces. We employ early stopping based on validation performance with a patience of 5 epochs. Data augmentation includes random rotations and flips of the image patches. The pipeline is implemented in PyTorch using `TorchScript` for optimal GPU utilization. Training is performed on distributed NVIDIA A100 GPUs using PyTorch's `DistributedDataParallel`, with environment simulation and data generation parallelized across multiple nodes to maximize throughput.

B. Conditioning Navigation Policies Using TSUMs

To enable robots to manage uncertainty in a task-aware manner, we propose conditioning the navigation policy on the Task-Specific Uncertainty Map (TSUM). To condition the policy, we augment the state representation ($\tilde{s}$) to include both the acceptable uncertainty levels from the TSUM and

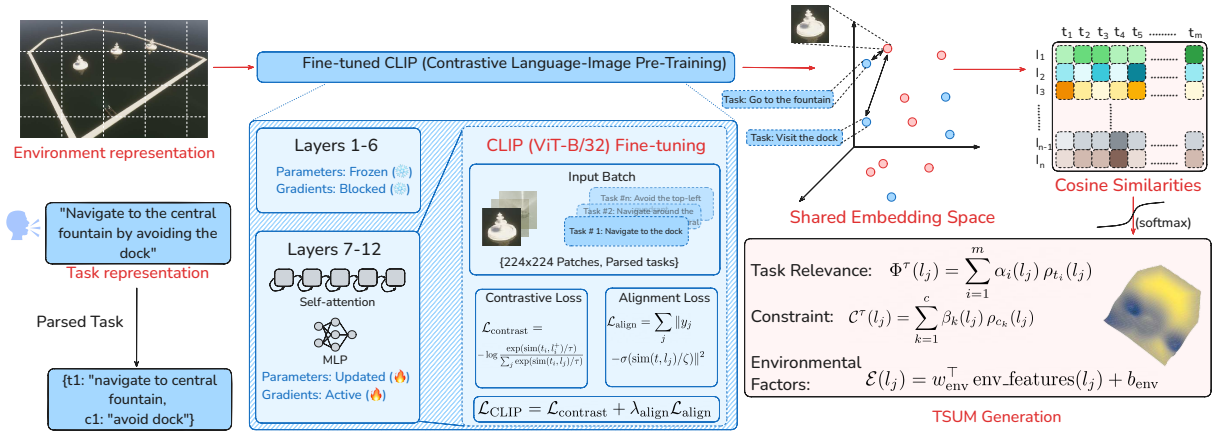


Fig. 3: CLIP-based TSUM generation pipeline. Environment imagery and task description are encoded by a fine-tuned CLIP; their joint embeddings produce task-relevance and constraint maps that, fused with environment priors, yield the final TSUM.

the robot's current estimation of its state uncertainty:

$$\tilde{s} = [s, U^\tau(s), u(s)],$$

where s is the original state representing the robot's observation of the environment, $U^\tau(s)$ is derived from the TSUM, and $u(s)$ is the robot's current state estimation uncertainty.

By conditioning the policy $\pi(a|\tilde{s})$ on the augmented state $\tilde{s}$, the agent makes decisions based not only on the environmental state but also on the acceptable, actual and future uncertainty levels at each location. This allows the agent to adjust its actions to meet the task-specific uncertainty requirements. The RL problem is formulated in the augmented state space. The objective is to learn an optimal policy $\pi^*(a|\tilde{s})$ that maximizes the expected cumulative reward:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, \tau) \,\middle|\, \tilde{s}_0 \right], \tag{4}$$

where $\gamma \in [0, 1)$ is the discount factor, and $R(s_t, a_t, \tau)$ is the task-specific reward function. Standard RL algorithms can be employed to solve this optimization problem in the augmented state space. In the next section, we show how a standard RL algorithm can be adapted using our framework.

1) GUIDEd SAC: While any RL algorithm can be used to learn the optimal policy, we adopt the Soft Actor-Critic (SAC) algorithm [30] due to its sample efficiency and robustness. Our adapted version, referred to as *GUIDEd SAC*, conditions the policy and value function on the augmented state $\tilde{s}$. In the SAC framework, the objective is to maximize the expected cumulative reward augmented by an entropy term, which encourages exploration:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, \tau) + \alpha \mathcal{H}(\pi(\cdot|\tilde{s}_t)) \right) \,\middle|\, \tilde{s}_0 \right],$$

where $\mathcal{H}(\pi(\cdot|\tilde{s}_t)) = -\mathbb{E}_{a_t \sim \pi(\cdot|\tilde{s}_t)} [\log \pi(a_t|\tilde{s}_t)]$ is the entropy at state $\tilde{s}_t$, and α is the temperature parameter balancing exploration and exploitation. GUIDEd SAC maintains parameterized function approximators for the policy $\pi_\theta(a|\tilde{s})$ and the soft Q-value functions $Q_{\phi_1}(\tilde{s}, a)$ and $Q_{\phi_2}(\tilde{s}, a)$, where θ and ϕ_i denote the parameters of the policy and value networks, respectively. The soft Q-value networks $Q_{\phi_i}(\tilde{s}, a)$ are updated by minimizing the soft Bellman residual:

$$\mathcal{L}_Q(\phi_i) = \mathbb{E}_{(\tilde{s}_t, a_t, r_t, \tilde{s}_{t+1}) \sim \mathcal{D}} \left[\left(Q_{\phi_i}(\tilde{s}_t, a_t) - y_t \right)^2 \right], \tag{5}$$

where $\mathcal{D}$ is the replay buffer, and target y_t is computed as:

$$y_t = r_t + \gamma \left(\min_{i=1,2} Q_{\bar{\phi}_i}(\tilde{s}_{t+1}, a_{t+1}) - \alpha \log \pi_\theta(a_{t+1}|\tilde{s}_{t+1}) \right),$$

with $a_{t+1} \sim \pi_\theta(\cdot|\tilde{s}_{t+1})$ and $Q_{\bar{\phi}_i}$ being the target Q-value networks with delayed parameters for stability. The policy network $\pi_\theta(a|\tilde{s})$ is updated by minimizing:

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{\tilde{s}_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\theta(\cdot|\tilde{s}_t)} \left[\alpha \log \pi_\theta(a_t|\tilde{s}_t) - Q_\phi(\tilde{s}_t, a_t) \right] \right], \tag{6}$$

where $Q_\phi(\tilde{s}_t, a_t) = \min_{i=1,2} Q_{\phi_i}(\tilde{s}_t, a_t)$. The temperature parameter α is adjusted by minimizing:

$$\mathcal{L}(\alpha) = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\tilde{s}_t)} \left[-\alpha \left(\log \pi_\theta(a_t|\tilde{s}_t) + \bar{\mathcal{H}} \right) \right], \tag{7}$$

where $\bar{\mathcal{H}}$ is the target entropy. Algorithm 1 summarizes the GUIDEd SAC algorithm.

Initialize policy network $\pi_\theta(a|\tilde{s})$, Q-value networks Q_{ϕ_1}, Q_{ϕ_2}, target Q-value networks $Q_{\bar{\phi}_1}$, $Q_{\bar{\phi}_2}$, temperature parameter α, and replay buffer $\mathcal{D}$.
for each environment interaction step **do**
  Obtain s_t, $U^\tau(s_t)$, $u(s_t)$.
  Form augmented state $\tilde{s}_t = [s_t, U^\tau(s_t), u(s_t)]$.
  Sample action $a_t \sim \pi_\theta(\cdot|\tilde{s}_t)$.
  Execute action a_t, observe r_t and s_{t+1}.
  Compute $\tilde{s}_{t+1} = [s_{t+1}, U^\tau(s_{t+1}), u(s_{t+1})]$.
  Store transition $(\tilde{s}_t, a_t, r_t, \tilde{s}_{t+1})$ in replay buffer.
for each gradient step **do**
  Sample minibatch of transitions from $\mathcal{D}$.
  Update Q_{ϕ_i} by minimizing $\mathcal{L}_Q(\phi_i)$ (Eq. (5)).
  Update π_θ by minimizing $\mathcal{L}_\pi(\theta)$ (Eq. (6)).
  Adjust α by minimizing $\mathcal{L}(\alpha)$ (Eq. (7)).
  Update target Q networks: $\bar{\phi}_i \leftarrow \tau\phi_i + (1-\tau)\bar{\phi}_i$.

Algorithm 1: GUIDEd SAC Algorithm

Tasks	Metric	SAC	SAC-P	B-SAC	CVaR	RAA	HEU	G-PPO	G-SAC
Goal reaching: waypoint	TCR (%) (↑)	67.2%	82.1%	75.9%	68.8%	35.3%	71.3%	83.8%	95.7%
`visit [coordinate]`	Reward (avg) (↑)	186.2	260.4	249.5	184.8	26.9	176.1	319.0	429.2
Goal reaching: context	TCR (%) (↑)	68.9%	84.3%	74.2%	66.8%	32.7%	73.5%	81.7%	91.5%
`navigate to dock`	Reward (avg) (↑)	144.1	241.8	189.2	134.6	53.5	137.5	308.3	400.2
Avoid tasks	TCR (%) (↑)	71.3%	83.2%	79.6%	78.4%	51.3%	62.4%	89.6%	87.4%
`avoid the central fountain`	Reward (avg) (↑)	177.8	199.2	277.6	220.4	107.8	174.4	437.6	480.2
Perimeter tasks	TCR (%) (↑)	44.3%	51.6%	56.3%	41.6%	39.8%	49.6%	74.9%	85.6%
`go around the left fountain`	Reward (avg) (↑)	84.4	132.8	170.4	32.8	111.2	146.8	449.2	599.6
Explore tasks	TCR (%) (↑)	88.6%	92.4%	87.6%	84.9%	71.3%	88.7%	93.8%	97.3%
`explore top-right quadrant`	Reward (avg) (↑)	486	474	526	449	402	537	548	595
Restricted areas	TCR (%) (↑)	70.8%	82.1%	78.4%	71.2%	51.1%	63.9%	90.0%	88.6%
`visit dock, avoid top-right quadrant`	Reward (avg) (↑)	266.4	306.8	307.2	269.6	201.4	261.2	570.0	634.6
Multi-goal tasks	TCR (%) (↑)	31.3%	42.9%	37.7%	30.9%	19.5%	42.1%	72.8%	81.7%
`combination of tasks (see Fig. 3)`	Reward (avg) (↑)	124.4	135.2	122.4	129.2	100.4	155.2	423.6	510.4

TABLE I: Task-completion rate (TCR) and mean episode reward for GUIDE vs baselines, averaged over 50 trials per task type. Task completion criteria: (i) goal-reaching tasks: within a 1.5m radius of the target, (ii) perimeter/exploration tasks: remain within a 3.5 m corridor of the path, (iii) avoid tasks: keep a 3 m distance. TCR is the fraction of these criteria met.

IV. Experiments

Motivating scenario: Consider a stealth mission where an autonomous surface vehicle (ASV) must navigate through previously unexplored waters. The ASV can either rely on noisy sensors or activate high-precision GPS. Each GPS fix incurs a penalty because it increases the chance of detection, yet completing the mission requires maintaining sufficient localization accuracy in critical regions.

A. Experimental Setup

1) Environment: The ASV operates in an open lake characterized by environmental variability and human-induced disturbances. The environment features fountains that serve as obstacles and introduce unmodeled water disturbances that create both physical challenges for navigation and contribute to the overall uncertainty in state estimation.

2) Hardware Setup: Our experiments utilize the SeaRobotics Surveyor Autonomous Surface Vehicle (ASV), which is equipped with a GPS, an IMU, and a EXO2 multiparameter water quality sonde. The GPS provides precise positioning, while the IMU assists with motion tracking and orientation. The YSI EXO2 sonde, primarily used for environmental monitoring, contributes to the overall state estimation by providing additional context. The action space of the ASV is defined as $a = (\lambda, \alpha, \eta)$, where λ is the propulsion torque, α denotes the steering angle, and η is a discrete variable indicating the mode of position estimation.

3) Position Estimation Modes: To represent the operational challenges of managing uncertainty during navigation, we model two modes of position estimation [2]:

(i) **Noisy Localization**: By default, the ASV estimates its position using the IMU and EXO2 sensors, which results in a less accurate position estimate. This mode represents the low-cost but high-uncertainty estimate.

(ii) **Exact Localization**: The ASV can request exact position data from GPS. This incurs a higher resource cost, leading to a reduced overall mission reward.

The reward structure is designed to encourage task completion while balancing the efficient management of positional uncertainty, without biasing the results in favor of GUIDE compared to the baselines. The reward function for task execution, r_{task}, incentivizes mission completion, while each use of exact position estimation incurs a penalty c_{exact} to reflect its associated cost. Conversely, noisy estimation incurs a smaller penalty c_{noisy}, encouraging the ASV to balance accuracy needs with operational efficiency.

B. Baselines and Ablations

We compare GUIDE against baselines that handle uncertainty differently. They interpret tasks using the same parser, converting them into algorithm-specific representations:

(i) *Standard RL without TSUMs (SAC)*: We train SAC using the original state s to assess the importance of TSUMs.

(ii) *GUIDEd PPO (G-PPO)*: We implement GUIDEd PPO [31], incorporating TSUMs and uncertainty, to examine the effect of the underlying RL algorithm.

(iii) *RL with Uncertainty Penalization (SAC-P)*: We modify the reward function in SAC to penalize high uncertainty: $R_{\text{SAC-P}} = R_{\text{base}} - \zeta u(s)$, where R_{base} includes task and localization costs and ζ is a weighting factor. This tests traditional reward shaping versus GUIDE's approach.

(iv) *Bootstrapped Uncertainty-Aware RL (B-SAC)*: We implement Bootstrapped SAC [32] to estimate epistemic uncertainty which guides exploration.

(v) *Heuristic Policy (HEU)*: A hand-crafted policy where the agent plans using SAC but switches to exact position estimation near obstacles or task-critical regions.

(vi) *Risk-Aware RL (CVaR)*: We use a CVaR-based RL algorithm [33], optimizing a risk-sensitive objective to focus on worst-case scenarios.

(vii) *Uncertainty-Aware Planning (RAA)*: We employ Risk-Aware A* [34] that minimize collision probabilities, aiming for overall uncertainty minimization.



GUIDEd SAC SAC GUIDEd PPO

Fig. 4: Comparison of navigation trajectories for the task: `Start at the dock, navigate around the central fountain, then around the left fountain, and finally around the right fountain.` Red dots mark locations where it reduced uncertainty, pink diamonds indicate collision points and the green rectangle marks the dock.

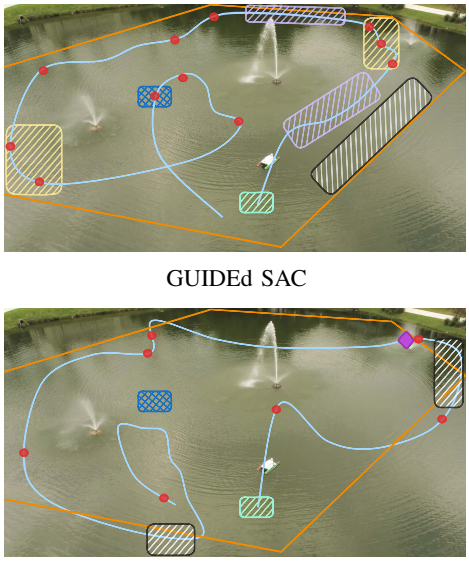
C. Results

The performance of GUIDE and the baseline methods is summarized in Table I. Our results show that GUIDE consistently outperforms all baselines across all task categories.

1) Impact of TSUMs: To assess the significance of TSUMs, we compare the standard SAC to G-SAC. As shown in Table I, conditioning on TSUMs significantly enhances performance across all tasks. Without TSUMs, SAC cannot manage positional uncertainty in a task-specific manner, leading to suboptimal decisions and lower rewards. To illustrate the effect of TSUM integration, we show in Fig. 4 the trajectories with and without TSUMs. We observe that SAC often overuses high-precision localization in areas where it is unnecessary, incurring additional costs without significant benefits. In critical regions requiring precise navigation, SAC fails to reduce uncertainty, leading to collisions. In contrast, the G-SAC uses TSUMs to adapt its uncertainty management, switching to GPS in areas where the TSUM indicates low acceptable uncertainty, enabling the ASV to navigate safely around obstacles and complete tasks efficiently.

2) Effect of RL Algorithm: We investigate the impact of the underlying RL algorithm by comparing G-PPO to G-SAC. This difference in the performance can be attributed to factors inherent to the algorithms. G-SAC, based on the SAC framework, is an off-policy method that leverages entropy regularization to encourage exploration while maintaining stability. Its off-policy nature allows for efficient sample utilization, which is particularly beneficial in continuous action spaces and when data collection is costly or limited. In contrast, PPO relies on proximal updates to prevent large policy shifts, using a clipped objective function. While PPO is stable, it can be less sample-efficient, as it requires new data for each policy update and may not explore as effectively in complex environments. Our empirical results suggest that the off-policy efficiency and exploration capabilities of G-SAC make it better suited for navigation tasks. The entropy term in G-SAC encourages the agent to consider a wider range of actions, enabling it to discover more optimal strategies for managing uncertainty in a task-specific context.

3) Comparison with Baselines: Standard RL methods like SAC and SAC-P lack task-specific uncertainty management.



GUIDEd SAC

SAC-P

Fig. 5: Trajectories for `Start and end at the dock. Go around the perimeter and visit (x,y).` Green and blue rectangles denote the dock and the coordinates (x,y).

SAC-P penalizes high uncertainty uniformly, resulting in overly conservative behavior where precision is unnecessary and insufficient caution in critical regions, ultimately leading to lower performance. B-SAC estimates epistemic uncertainty but fails to adapt to task-specific requirements, leading to inefficient exploration. Risk-Aware RL methods like CVaR are uniformly risk-averse, missing opportunities for calculated risk-taking that could improve task success. RAA aims to minimize overall uncertainty without considering task context, often generating inefficient paths. HEU switches to exact localization near obstacles, but lacks GUIDE's adaptability, failing to adjust to sudden changes in uncertainty requirements.

4) Behavior of GUIDEd Agents: Analyzing the specific task shown in Fig. 5, GUIDEd agents strategically adjust their reliance on precise position estimation versus noisier estimates. In areas where the TSUMs indicate high precision

is necessary–such as navigating near obstacles or close to the perimeters (Fig. 5, highlighted in yellow)–the ASV opt for exact positioning despite the higher operational cost. Conversely, in less critical regions (Fig. 5, highlighted in purple), they rely on less precise, cheap estimates. This adaptability allows GUIDEd agents to manage uncertainty more efficiently than baselines, resulting in better task completion rates. Although not perfect, occasionally missing sections of the perimeter (Fig. 5, highlighted in black), GUIDEd agents significantly outperform baselines with engineered rewards like SAC-P. Baselines frequently fail to complete tasks safely, colliding with obstacles (Fig. 5, pink diamond) or take inefficient paths (Fig. 5, highlighted in black).

V. Conclusion

We introduce GUIDE, a framework for limited-localization environments that integrates task-specific uncertainty requirements into robotic navigation policies. Central to our approach is the concept of Task-Specific Uncertainty Maps (TSUMs), which represent acceptable levels of state estimation uncertainty across regions of the environment based on the given task. By conditioning navigation policies on TSUMs, we enable robots to reason about the context-dependent importance of certainty. We demonstrated how GUIDE can be incorporated into RL frameworks by augmenting the state representation with TSUMs. Specifically, we adapted the Soft Actor-Critic (GUIDEd SAC) to operate in this augmented state space. Our experiments demonstrate GUIDE's effectiveness in real-world deployments across previously unseen environments, showing significant improvements in task completion rates compared to baselines that do not explicitly consider task-specific uncertainty.

References

[1] F. Gul, W. Rahiman, and S. S. Nazli Alhady, "A comprehensive study for robot navigation techniques," *Cogent Engineering*, 2019.

[2] G. Puthumanaillam, M. Vora, and M. Ornik, "ComTraQ-MPC: Meta-trained dqn-mpc integration for trajectory tracking with limited active localization updates," *arXiv preprint arXiv:2403.01564*, 2024.

[3] N. Paulig and O. Okhrin, "Robust path following on rivers using bootstrapped reinforcement learning," *Ocean Engineering*, 2024.

[4] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking reinforcement learning techniques for autonomous navigation," in *IEEE International Conference on Robotics and Automation*, 2023.

[5] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, 2019.

[6] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using gaussian process prediction and observation models," *Autonomous Robots*, 2009.

[7] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, "Particle filters for mobile robot localization," in *Sequential Monte Carlo methods in practice*. Springer, 2001.

[8] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, 1998.

[9] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *Neural Information Processing Systems*, 1999.

[10] C. Zhu, "Intelligent robot path planning and navigation based on reinforcement learning and adaptive control," *Journal of Logistics, Informatics and Service Science*, 2023.

[11] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "DWA-RL: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *IEEE International Conference on Robotics and Automation*, 2021.

[12] A. Curtis, G. Matheos, N. Gothoskar, V. Mansinghka, J. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Partially observable task and motion planning with uncertainty and risk awareness," *arXiv preprint arXiv:2403.10454*, 2024.

[13] S. Forestier, R. Portelas, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," *Journal of Machine Learning Research*, 2022.

[14] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *Neural Information Processing Systems*, 2018.

[15] S. Guo, X. Zhang, Y. Du, Y. Zheng, and Z. Cao, "Path planning of coastal ships based on optimized DQN reward function," *Journal of Marine Science and Engineering*, 2021.

[16] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, 2021.

[17] M. Budd, P. Duckworth, N. Hawes, and B. Lacerda, "Bayesian reinforcement learning for single-episode missions in partially unknown environments," in *Conference on Robot Learning*, 2023.

[18] M. Alali and M. Imani, "Bayesian reinforcement learning for navigation planning in unknown environments," *Frontiers in Artificial Intelligence*, 2024.

[19] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Neural Information Processing Systems*, 2016.

[20] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *Proceedings of the AAAI conference on artificial intelligence*, 2021.

[21] G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, and S. E. Li, "Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness," *Transportation Research Part C: Emerging Technologies*, 2022.

[22] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using cvar-constrained optimization," *IEEE Robotics and Automation letters*, 2019.

[23] B. Lütjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *2019 International Conference on Robotics and Automation*, 2019.

[24] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep RL with a safety critic," *arXiv preprint arXiv:2010.14603*, 2020.

[25] D. Shah, B. Osiński, S. Levine, *et al.*, "LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on Robot Learning*, 2023.

[26] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "ViNT: A foundation model for visual navigation," *arXiv preprint arXiv:2306.14846*, 2023.

[27] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A general navigation model to drive any robot," in *2023 IEEE International Conference on Robotics and Automation*, 2023.

[28] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*.

[29] E. F. Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," *arXiv preprint cs/0306050*, 2003.

[30] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, 2018.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[32] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy Q-Learning via bootstrapping error reduction," in *Neural Information Processing Systems*, 2019.

[33] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[34] C. Hu and Y. Jin, "Long-range risk-aware path planning for autonomous ships in complex and dynamic environments," *Journal of Computing and Information Science in Engineering*, 2023.