Scaling Long-Horizon Online POMDP Planning via Rapid State Space Sampling

Yuanchu Liang[1*], Edward Kim[1*], Wil Thomason[2*], Zachary Kingston[2*], Hanna Kurniawati[1], and Lydia E. Kavraki[2]

[1] Australian National University, Canberra ACT 2601, Australia,
{Yuanchu.Liang, Edward.Kim, hanna.kurniawati}@anu.edu.au
[2] Rice University, Houston TX 77005, USA,
{wbthomason, zak, kavraki}@rice.edu

Abstract. Partially Observable Markov Decision Processes (POMDPs) are a general and principled framework for motion planning under uncertainty. Despite tremendous improvement in the scalability of POMDP solvers, long-horizon POMDPs (*e.g.*, ≥ 15 steps) remain difficult to solve. , ROP-RAS3 is evaluated on various long-horizon POMDPs, including on a problem with a planning horizon of more than 100 steps and a problem with a 15-dimensional state space that requires more than 20 look ahead steps. In all of these problems, ROP-RAS3 *substantially outperforms* other state-of-the-art methods by up to *multiple folds*.

Keywords: Motion Planning, Sampling-Based Motion Planning, Planning under Uncertainty, POMDP, Hardware Acceleration

1 Introduction

Motion planning in the partially observed and non-deterministic world is a critical component of reliable and robust robot operation. Partially Observable Markov Decision Processes (POMDPs) [7,22] are a natural way to formulate such problems. The key insight of the POMDP framework is to represent uncertainty on the effects of actions, perception and initial state as probability distributions, and then reason about the best strategy to perform with respect to distributions over the problem's state space, called *beliefs*, rather than the state space itself.

Although a POMDP's systematic reasoning about uncertainty comes at the cost of high computational complexity [18], the POMDP framework is practical for many robotics problems [11], thanks in large part to sampling-based approaches. These approaches relax the problem of finding an optimal solution to an approximate one by sampling states from the belief space and computing the best strategies from only the samples. Scalable anytime methods under this approach (surveyed in [11]) have been proposed for solving large POMDP problems. However, computing a good solution to long-horizon (*e.g.*, ≥ 15 steps) POMDPs remains difficult.

Early results from the literature [12] indicate that Sampling-Based Motion Planning (SBMP)—sampling-based approaches designed for deterministic motion

*Authors have equal contributions.

planning—help alleviate the challenges of long-horizon problems in offline POMDP planning. Specifically, SBMPs can be used to generate suitable macro actions (*i.e.*, sequences of actions) to reduce the effective planning horizon for a POMDP solver. Macro actions generated via SBMP automatically adapt to geometric features of the valid region of the state space and tend to cover a diverse set of macro actions.

Although the above approach performs well for offline POMDP planning, it is often impractical for online planning for two reasons. First, the speed of SBMP, which historically required hundreds of milliseconds to tens of seconds to find a single motion plan. Second, most online POMDP planners [13,21,27] exhaustively enumerate each action at each sampled belief in computing the best action to perform. When macro actions are used, exhaustive enumeration is performed over all macro actions; thus, the number of macro actions should be kept low, thereby limiting macro action diversity in online POMDP planning.

However, the recently proposed Vector-Accelerated Motion Planning (VAMP) framework [25] enables SBMPs to find solutions on microsecond timescales, multiple orders of magnitude faster than prior approaches. Concurrently, recently proposed *reference-based POMDP planners* [9] remove the requirement for exhaustive enumeration of actions at a small cost to full optimality. Leveraging these two advances, we propose an online POMDP solver, called Reference-Based Online POMDP Planning via Rapid State Space Sampling (ROP-RaS3), which is a reference-based POMDP planner that employs a VAMP-enhanced macro action sampler as its underlying reference policy.

We evaluate ROP-RaS3 on multiple long-horizon POMDPs, including a problem that requires over 100 lookahead steps and a 15-dimensional problem with a planning horizon of over 20. Comparisons with state-of-the-art online POMDP planners—including POMCP [21], a POMCP modification that uses VAMP to generate macro actions, and DESPOT [27] with learned macro actions [15,16]—indicate ROP-RaS3 substantially outperforms state-of-the-art methods in all evaluation scenarios.

2 Background and Related Work

2.1 Sampling-Based Motion Planners

Sampling-based motion planning (SBMP) is a common, effective family of algorithms (*e.g.*, [8,10]) for solving *deterministic* motion planning problems [14]. They are empirically able to find collision-free motions for high degree-of-freedom (DoF) robots in environments containing many obstacles by drawing samples from a robot's *configuration space*, the set of all possible robot configurations, $q \in Q$. Constraints, such as (most commonly) avoiding collisions between the robot and the environment or itself, partition the configuration space into *valid*—Q_{free}—and *invalid*—$Q \setminus Q_{\text{free}}$—configurations. A deterministic *motion planning problem* is then a tuple $(Q_{\text{free}}, q_I, Q_G)$ representing the task of finding a continuous path, $p : [0, 1] \to Q_{\text{free}}$, from an initial configuration, q_I, to a goal region, $Q_G \subseteq Q_{\text{free}}$ (*i.e.*, $p(0) = q_I$ and $p(1) \in Q_G$).

SBMPs solve such problems by building a discrete approximation of Q_{free} as a graph or tree connecting sampled configurations in Q_{free} by short, local motions. Once both q_I and Q_G are connected by this structure, a valid robot motion plan can be found with graph search methods. The solutions are deterministic, open-loop plans, and may not be robust against uncertainties or changes in configuration spaces.

2.2 POMDP Background

An infinite-horizon POMDP is defined as the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, R, \gamma, b_0 \rangle$ where $\mathcal{S}$ denotes the set of all possible states of the robot, $\mathcal{A}$ denotes the set of all possible robot actions, and $\mathcal{O}$ denotes the set of all possible robot observations. In our context, the state space encompasses the robot's configuration space. The transition function $\mathcal{T}(s' \mid s, a)$ is the conditional probability that the robot will be in state $s' \in \mathcal{S}$ after performing action $a \in \mathcal{A}$ from state $s \in \mathcal{S}$. The observation function $\mathcal{Z}(o \mid s', a)$ is the conditional probability that the agent perceives $o \in \mathcal{O}$ when it is in state $s' \in \mathcal{S}$ after performing action $a \in \mathcal{A}$. The reward is a bounded real-valued function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The parameter $\gamma \in (0, 1)$ is the discount factor which ensures the objective function is well-defined.

In general, the agent does not know the true state. At each time-step, the agent maintains a *belief* about its state—a probability distribution over the state space. The space of all possible beliefs is denoted by $\mathcal{B} := \Delta(\mathcal{S})$. The agent starts from a given initial belief, denoted as b_0. If b' denotes the agent's next belief after taking action $a \in \mathcal{A}$ and receiving the observation $o \in \mathcal{O}$, then the updated belief is given by $b'(s') = \tau(b, a, o) \propto \mathcal{Z}(o \mid s', a) \sum_{s \in \mathcal{S}} \mathcal{T}(s' \mid s, a)\, b(s)$. For any given belief b and action a, the expected reward is given by $R(b, a) := \sum_{s \in \mathcal{S}} R(s, a) b(s)$.

A (stochastic) *policy* is a mapping $\pi : \mathcal{B} \to \Delta(\mathcal{A})$. We denote its distribution for any given input $b \in \mathcal{B}$ by $\pi(\cdot \mid b)$. A policy is *deterministic* if it only has support at a single point $a \in \mathcal{A}$. Let Π be the class of all policies. Given a policy $\pi \in \Pi$, we define the *value function* $V^\pi : \mathcal{B} \to \mathbb{R}$ to be the expected total discounted reward, $V^\pi(b) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R\big(b_t^\pi, A_t^\pi\big)]$. In this paper, a *solution* to the POMDP is a deterministic policy $\pi^* \in \Pi$ satisfying $V^{\pi^*}(b) = \sup_{\pi \in \Pi} V^\pi(b)$ on $\mathcal{B}$. The Bellman equation

$$V(b) = \max_{a \in \mathcal{A}} \Big[R(b, a) + \gamma \sum_{o \in \mathcal{O}} P(o \mid a, b) V\big(\tau(b, a, o)\big) \Big] \tag{1}$$

is satisfied by the optimal value function V^{π^*}. The intrinsic conditional probability $P(o \mid a, b) := \sum_{s' \in \mathcal{S}} \mathcal{Z}(o \mid s', a) (\sum_{s \in \mathcal{S}} \mathcal{T}(s' \mid s, a) b(s))$ is the probability that the agent perceives o, having performed the action a, under the belief b. Since beliefs are sufficient statistics of the entire history of action-observation pairs from the initial belief b_0 [5], beliefs and histories $h_t := \{(a_i, o_i)\}_{i=0}^t$ are interchangeable.

2.3 Long-Horizon POMDPs

Solving long-horizon POMDPs remains a significant challenge as the set of possible future events grows exponentially with respect to the planning horizon. Planning

over *macro actions*—a set of action sequences—is a common approach to deal with long-horizon POMDPs [4,6,12,15,16,24]. Although this reduces the effective planning horizon, choosing a set of good macro actions for planning is critical and automatic construction of macro actions can require significant computational effort, *e.g.*, the learning time for MAGIC [15] can be on the order of hours.

Irrespective of whether macro actions are used, most online POMDP planners [13,27,21] exhaustively enumerate all actions from each sampled belief in order to estimate the optimal action. Most efficient optimization methods, which are generally based on gradient ascent, cannot be used effectively because computing the gradient of the POMDP value function is very expensive. Therefore, existing planners seldom explore long-term information and tend to perform poorly for horizons greater than 15 steps.

Recently, the authors of [9] have mitigated this limitation by solving a *reference-based POMDP*: a reformulated POMDP whose reward objective incurs a KL-penalty for deviating too far from a given stochastic *reference belief-to-belief transition*: $\bar{U}(a, o \mid b) := P(o \mid a, b)\bar{\pi}(a \mid b)$ where $\bar{\pi}$ is a given stochastic reference policy. The reward with respect to U is defined as $R(b, U) := \sum_{a,o} R(b, a)$ $U(a, o \mid b)$. The value of a belief is given by

$$\mathcal{V}^*(b) = \sup_{U \in \mathcal{U}(b)} \left(R(b, U) - \mathrm{KL}(U \,\|\, \bar{U}) + \gamma \sum_{a,o} U(a, o \mid b) \mathcal{V}^*(\tau(b, a, o)) \right) \tag{2}$$

where $\mathcal{U}(b) \subseteq \Delta(\mathcal{A} \times \mathcal{O})$ is the set of admissible transitions at belief b. The RHS can be optimized analytically, effectively removing the need for enumerative optimization in online POMDP solving and thereby reducing the branching factor of the belief tree. Preliminary results from [9] indicate that policies generated by this procedure can outperform state-of-the-art POMDP benchmarks on long-horizon POMDPs. However, the paper [9] employs relatively crude reference policies nor does it incorporate macro actions into planning.

3 POMDP Planning with SBMP-Generated Trajectories

We begin by presenting a general mechanism in Algorithm 1 to integrate belief-space sampling of a canonical online POMDP planner with state space sampling to generate macro-actions using SBMP. In principle, Algorithm 1 can use any SBMP planner and can generalize to most existing online particle-based POMDP planners, such as POMCP [21], DESPOT [27] and their derivatives [3,13,15,16].

To infer the best macro action, SBMPExpandTree in Algorithm 1 samples a set of beliefs reachable from its current belief and represents them as a *belief tree*, T—i.e. a tree whose nodes store beliefs (typically represented as sampled state particles) and whose edges represent action-observation pairs. Methods vary in the construction of the belief tree. Generally-speaking, online POMDP planners search the tree forward by judicious sampling—the choice of sampler is the main difference between planners. When an unvisited node is encountered during the search, existing planners expand every macro action to receive a crude

Algorithm 1 Online POMDP Planner with SBMP-Generated Macro Actions

1: **Initialize** T from belief particles
2: **while** time permitting **do**
3: $T \leftarrow$ SBMPExpandTree(T)
4: Re-estimate values on T and propagate back up to the root node
5: **end while**

SBMPExpandTree(T)

1: Search T via SBMP and store node data
2: Arrive at a (set of) nodes H in T to expand
3: **for** $h \in H$ **do**
4: Rapidly sample SBMP paths p between particles in h to selected targets
5: Add nodes corresponding to actions crafted from p to T
6: Sample new observations and histories, update state particles and add to T
7: **while** not stopping criteria (e.g. desired depth reached) **do**
8: Repeat lines 1–6 on the newly expanded tree T
9: **end while**
10: **end for**

value estimate before propagating newly obtained information back to the root node.

The choice of sampler for forward search is an essential component for achieving high-quality policies online. The development of hardware-accelerated SBMP via SIMD-vectorization in [25] provides a powerful capability to generate computationally cheap, high-quality trajectories in the state space. It enables SBMP planners to rapidly generate constraint-satisfying (*e.g.*, collision-free) state space paths to goals or to highly informative states, which in turn can be used as macro actions for the POMDP planner. Thus, promising macro actions can be dynamically created as a subroutine (line 4 in SBMPExpandTree) *within* a POMDP planner itself in fractions of a second.

However, fast macro action sampling alone is not sufficient, as current online POMDP planners perform numerical optimization via enumeration of *all* (macro) actions at each sampled belief, which results in a time and space complexity of $\mathcal{O}(\#\text{macro\_actions}^h)$, where h is the planning horizon. This complexity significantly limits the number of macro actions and the depth of the tree it can construct, thereby significantly limiting the benefit of SBMP for POMDP solving. To tackle this problem, our planner ROP-RAS3 draws on insights from the reference-based POMDP [9] formulation while, in tandem, exploiting the speed of VAMP [25]—an implementation of SIMD-vectorized SBMP—to create a rich set of diverse macro actions, which the planner uses to efficiently explore relevant parts of the belief space. Section 4 provides the details of ROP-RAS3 and Section 5 empirically verifies the claims made above.

4 ROP-RaS3

This section introduces our online anytime planner Reference-Based Online POMDP Planning via Rapid State Space Sampling (ROP-RaS3). To this end, we motivate it by briefly outlining VAMP's capability to induce high-quality reference policies before adapting the reference-based POMDP formulation from [9] to our context. The algorithm is then constructed on top of these components.

4.1 Vector Accelerated Motion Planning (VAMP)

Towards achieving fast motion planning, recent works have introduced new perspectives on *hardware-accelerated* SBMPs, using either CPU single-instruction, multiple-data (SIMD) [25] or GPU single-instruction, multiple-thread (SIMT) [23] parallelism to find complete motion plans in tens of microseconds to tens of milliseconds. In particular, the authors of VAMP [25] proposed a SIMD-vectorized approach to computing SBMP primitives (*i.e.*, local motion validation) that applies to all SBMPs and is available on any modern computer without specialized hardware; thus, it is now possible to generate probabilistically-complete, global, collision-free trajectories for high-DoF systems at kilohertz rates—on the scale of tens of thousands of plans per second. The key insight of this work is to lift the "primitive" operations of the SBMP to operate over *vectors* of configurations in parallel. Robot-specific code that uses these vector primitives is generated from a URDF using a tracing compiler—this pre-processing step is general to any robot.

Functionally, this enables checking validity of a spatially distributed set of configurations over a candidate motion in parallel for the cost of a single collision check, massively lowering the expected time it takes to find colliding configurations along said motion. This development has called into question previously held perceptions that SBMPs are relatively time-expensive subroutines and—in the context of planning under uncertainty—opens the door to using SBMPs to guide POMDP planning on the fly.

4.2 Reference-Based POMDPs over Stochastic Actions

The concept of a Reference-Based POMDP, as reviewed in Sect. 2.3, was introduced in [9] as a generalization of the MDP formulations using KL-penalization in [2,26] to POMDPs. One limitation is that the formulation given by (2) is somewhat artificial in that reference policies are stated with respect to *belief-to-belief* transitions (see discussion in Sect. 8 of [9]). In this paper, we will use a more natural formulation of a Reference-Based POMDP over *stochastic actions* rather than *belief-to-belief transitions* as in [9]. Note that in doing so, all the benefits of the formulation in [9] (as detailed in Sect. 2.3) are still preserved.

Specifically, a Reference-Based POMDP over stochastic actions is completely specified by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, \mathcal{T}, R, \gamma, \eta, \bar{\pi} \rangle$ where, in addition to the standard data, we have a *temperature* parameter $\eta > 0$ and a given (stochastic) *reference*

policy $\bar{\pi}(\cdot \,|\, b)$. Its value $\mathcal{V}$ for a given $b \in \mathcal{B}$ satisfies the Bellman equation

$$\mathcal{V}(b) = \sup_{\pi \in \Pi} \Big[R(b,\pi) - \frac{1}{\eta} \mathrm{KL}(\pi \,\|\, \bar{\pi}) + \gamma \sum_{a,o} P(o \,|\, a, b) \pi(a \,|\, b) \mathcal{V}\big(\tau(b,a,o)\big) \Big] \quad (3)$$

where $R(b,\pi) := \sum_{a,s} R(s,a)\pi(a \,|\, b) b(s)$ is the reward estimate. A *solution* is a stochastic policy $\pi \in \Pi$ that maximizes the value. The problem can therefore be viewed as a KL-penalized POMDP whose objective is modified to trade off two (potentially competing) objectives: (1) abide by the reference policy, and (2) maximize reward. The trade-off is balanced by η and the *quality* of the reference policy—higher-quality reference policies are those that reduce the KL-divergence between the solution of the unpenalized POMDP (1) and the reference policy. The supremum in (3) can be attained analytically by extending an argument of [1,2] to POMDPs. This yields an equivalent form of (3)—*i.e.*,

$$\mathcal{V}(b) = \frac{1}{\eta} \log \Big[\sum_{a} \bar{\pi}(a \,|\, b) \exp \Big\{ \eta \big[R(b,a) + \gamma \sum_{o} P(o \,|\, a, b) \mathcal{V}\big(\tau(b,a,o)\big) \big] \Big\} \Big]. \quad (4)$$

Moreover, the *exact* solution of the Reference-Based POMDP is given by

$$\pi^*(a \,|\, b) \propto \bar{\pi}(a \,|\, b) \exp \Big\{ \eta \Big[R(b,a) + \gamma \sum_{o} P(o \,|\, a, b) \mathcal{V}^*\big(\tau(b,a,o)\big) \Big] \Big\}. \quad (5)$$

The main point is that enumerative maximization can be avoided; instead, the solution can be approximated by successively iterating the analytic Bellman backup (4) exactly. This procedure is guaranteed to converge to a unique solution $\mathcal{V}^*$ from which the policy can be read off from (5). Unfortunately, needing *exact* backups is a significant requirement, given the cost of computing exact beliefs and, consequently, the immediate expected rewards $R(b,a)$ for every node in the belief tree. Still, the expectations $P\mathcal{V}(b,a) := \sum_o P(o \,|\, a, b) \mathcal{V}\big(\tau(b,a,o)\big)$ and $\mathcal{W}(b) := \sum_a \bar{\pi}(a \,|\, b) \exp\{\eta[R(b,a) + \gamma P\mathcal{V}(b,a)]\}$ can be approximated by *sampling* the reference policy and generative model. Improving value estimates is extremely efficient as backups can be computed by *maintaining* sums rather than enumeratively maximizing over actions.

Even so, the simplicity offered by the formulation comes at a cost. If the optimal policy of the POMDP with an unmodified objective is too far from the reference policy (in the sense of the KL-divergence), pure reward maximization can be compromised. Of course, the optimal policy and hence this divergence are not known a priori. Instead ROP-RAS3 assumes that reference policies generated by accelerated SBMPs provide a reasonable starting point (see Sect 4.1), leveraging them to rapidly sample high-quality deterministic policies to induce a reasonable partially observed reference policy which it deforms online. We emphasize that this modification is not negligible. Indeed, our results in Sect. 5 show that the performance deteriorates with problem complexity if the agent only executes the SBMP-induced reference policy as an open-controller without accounting for uncertainty.

4.3 ROP-RaS3

ROP-RaS3 (Algorithm 2) adds new history nodes h to the search tree by sampling macro actions $\vec{a}$ from a reference policy induced from a SIMD-vectorized SBMP up to a predefined depth D, keeping track of the states s, (macro) observations $\vec{o}$ and rewards r sampled under a *generative model* $\mathscr{G}$—*i.e.*, a simulator for the POMDP environment. When the required depth is reached, ROP-RaS3 obtains a crude estimate of the root node's value via a value heuristic (typically in the form of a rollout function). The planner then approximates the exact backup (MaintainExpectation) by carefully *maintaining* an empirical expectation, repeating backups on the simulated belief-tree path up to the root node—this form of backup is justified by our discussion in Sect. 4 and, in particular, Eq. (4). The above procedure is repeated until timeout, at which point the optimal policy's empirical estimate is read off from the tree's root node and executed—the form of estimate is given by (5).

The subroutine SampleMacroActionSBMP leverages VAMP as the reference policy to sample macro actions. Our implementation of ROP-RaS3 instantiates this subroutine with an instance of SampleHeuristics which selects meaningful target points, towards which VAMP's planners can be used to rapidly generate a path. For environments with well-defined goal and informative states, examples of SampleHeuristics include:

- **Uniform** Uniformly sample goal configurations with a given probability; otherwise, sample an informative state configuration uniformly.
- **Distance**. Uniformly sample a goal configuration with a given probability; otherwise, sample an informative state with a probability inversely proportional to a distance between the configurations of the current and informative states.
- **Entropy**. Given the normalized belief entropy $\mathcal{H}$, sample goal configurations with probability $1 - \mathcal{H}$; otherwise sample remaining informative configurations with probability inverse to a distance between the configurations of the current and informative states.

5 Experiments

We evaluated ROP-RaS3 on four different long-horizon POMDP problems systematically analyzing the effects of its respective components on its performance.

5.1 Scenarios and Benchmark Methods

Light-Dark (Figure 1a). A variation of the classical Light-Dark problem. The state and observation space are continuous and identical to those in [15,16,19], but we modify the problem to make the action space discrete—at each step, the robot can move in four cardinal directions with a fixed step-size of 0.5. Moreover, the agent receives feedback when it reaches the goal. The step and goal rewards are -0.1 and 100 respectively. The light stripe, the goal and the initial positions are randomly drawn from an 8×8 unit box but are constrained to be 4 units

Algorithm 2 ROP-RAS3

Initialize tree T rooted at h
while time permitting **do**
    SIMULATE(h)
end while
return T

SIMULATE(h)

Sample s from belief particles of h
if depth(h) $> D$ **then**
    return VALUEHEURISTIC(h, s)
end if
$\vec{a} \leftarrow$ SAMPLEMACROACTIONSBMP(h, s)
Sample $(s', \vec{o}, r(s, \vec{a}; \gamma))$ from generative model $\mathscr{G}(s, \vec{a})$
Create nodes for $h\vec{a}$ and $h\vec{a}\vec{o}$ if not created already
Add s' to belief particles of $h\vec{a}\vec{o}$
return $\mathcal{V}(h) \leftarrow \log($MAINTAINEXPECTATION$(h, \vec{a}, \vec{o}, r))/\eta$

SAMPLEMACROACTIONSBMP(h, s)

Get the current configuration q_{start} from s (assert free)
$q_{\text{goal}} \leftarrow$ SAMPLEHEURISTICS() (assert free)
Plan path p from q_{start} to q_{goal} using fast SBMP
return (macro) action $\vec{a}$ "fashioned" from p

MAINTAINEXPECTATION($h, \vec{a}, \vec{o}, r$)

$w \leftarrow N(h)\mathcal{W}(h) - N(h\vec{a}) \exp\left(\eta \left[r(h\vec{a}) + \gamma^{|\vec{a}|} P\mathcal{V}(h\vec{a})\right]\right)$
$p \leftarrow N(h\vec{a})P\mathcal{V}(h\vec{a}) - N(h\vec{a}\vec{o})\mathcal{V}(h\vec{a}\vec{o})$
$N(h\vec{a}) \leftarrow N(h\vec{a}) + 1$
$r(h\vec{a}) \leftarrow r(h\vec{a}) + r - r(h\vec{a})/N(h\vec{a})$
$P\mathcal{V}(h\vec{a}) \leftarrow \big(p + N(h\vec{a}\vec{o})$ SIMULATE$(h\vec{a}\vec{o})\big)/N(h\vec{a})$
$N(h) \leftarrow N(h) + 1$
return $\mathcal{W}(h) \leftarrow \left\{w + N(h\vec{a}) \exp\left(\eta \left[r(h\vec{a}) + \gamma^{|\vec{a}|} P\mathcal{V}(h\vec{a})\right]\right)\right\}/N(h)$

away from each other. Hence, a minimum of 8 primitive actions are required to navigate to the goal.

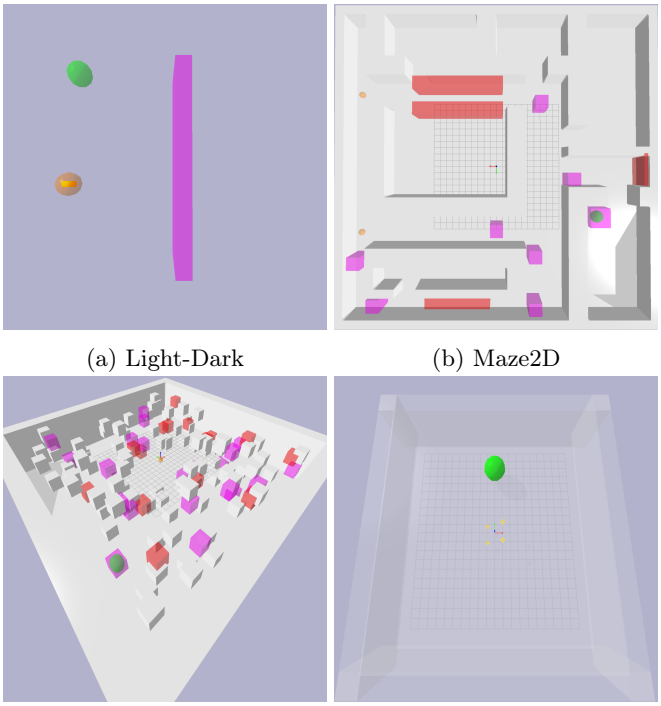
Maze2D (Figure 1b). This is a very long-horizon problem, modified from the discrete 2D Navigation scenario in [12]. A 2-DOF mobile cuboid robot needs to navigate from one of the spawn points to a goal region without colliding with obstacles or entering a danger zone. The robot's state is its position on the map ($\mathcal{S} = [0, 50]^2$). At each time step, the robot can move in one of the four cardinal directions ($|\mathcal{A}| = 4$) with a fixed step size, but there is a 20% chance that the robot moves in either direction orthogonal to the intended direction; if the resulting action would cause a collision with an obstacle, it does not move. The robot can only localize itself in landmark regions receiving position readings with small Gaussian noise inside the landmarks and no observations otherwise. The two spawn points are marked by orange circles in the maze. Each step costs the

robot -0.1, the danger zone penalty is -800 and the goal reward is 800. To perform well over the long horizon—a minimum of 100 primitive actions is required to reach the goal—this scenario requires the robot to take detours to localize and avoid danger zones before navigating to the goal.

Random3D (Figure 1c). This is a 3D navigation problem with uniformly randomized obstacles, landmarks, danger zones and goals; the only constraint is that the goal needs to be at least 40 units from the robot's spawn position at the map's center and a valid path must exist to the goal. The POMDP's state and action spaces are respectively $\mathcal{S} = [0, 50]^2 \times [0, 6] \subset \mathbb{R}^3$ and the 3D cardinal directions $\mathcal{A} = \{\text{North}, \text{South}, \text{East}, \text{West}, \text{Up}, \text{Down}\}$. The observation, transition and reward models are essentially the same as that of Maze2D except that the error directions are randomized from the 3D cardinal ones. Our aim is to systematically evaluate ROP-RAS3 on environments with progressively increasing obstacle density. The chances that the SBMP fails to find a path within a given time limit increases with the obstacle density and a POMDP planner needs to consider more diverse macro actions to find a good motion strategy.

Multi-Drone with A Teleporting Target (Figure 1d). This scenario with a high-dimensional state space and complex environment dynamics is similar to the multi-robot tag and predator-prey problems considered in [17,20]. Four drones, initialized at the center of the environment, need to work together to firstly detect and capture a point target (in green) whose initial position is not known to the drones. The POMDP's state space is $\mathcal{S} = ([0, 30]^2 \times [0, 4])^5 \subset \mathbb{R}^{15}$ consisting of the valid positions of the holonomic drones and an evading target. The action space is the space of all 4-tuples consisting of the 3D cardinal directions ($|\mathcal{A}| = 24$). Both the drones and the target move with a step size of 0.5. The target has a detection radius of 4; if any drones are within this range, the target will move in the direction furthest from the closest drone's location. Otherwise, it moves randomly. Each drone has a detection radius of 5 units and receives noisy readings of the target's position when the target is within this radius, otherwise no observation is received. If any drone receives an observation, all drones share this collective knowledge. The target is captured when at least one drone is within 1.5 units distance from the target. However, the target can *teleport* to the opposite of the map once it collides with the map boundaries but drones cannot. This problem requires drones to cooperate to capture the target by either surrounding it within the map's boundaries or by anticipating teleportation elsewhere. A reward of 500 is given if the target is captured, otherwise a -0.1 penalty is incurred by the drones for each step. A run terminates after 40 steps. This is a long-horizon problem—even if the target's initial position is known, at least 20 primitive steps are required to capture the tag.

We compare ROP-RAS3 with several instantiations of Algorithm 1 with state-of-the-art POMDP solvers, as well as our reference-based instantiation ROP-RAS3. Specifically, we compare with the following:



(a) Light-Dark (b) Maze2D

(c) Random3D (d) Multi-Drone Tag

Fig. 1: Benchmark environments with obstacles (**grey**), landmarks (**purple**), danger zones (**red**), starting locations (**orange**) and goals and targets (**green**)

Belief-VAMP (B-VAMP). The planner maintains the belief of the current state of the agent but only takes actions returned by VAMP's macro action sampler *without* planning.

Ref-Basic. The reference-based POMDP planner from [9] (no VAMP).

POMCP [21]. A standard benchmark for online POMDP planning.

R-POMCP. An instantiation of Algorithm 1 using POMCP with macro actions generated by VAMP in the same way as ROP-RAS3, but a finite set of macro actions are fixed for each belief node over which POMCP optimizes.

MAGIC [15]. A variation of DESPOT [27] that uses learnable macro actions generated via an actor-critic approach. MAGIC uses fixed-length macro actions, so comparisons are for varying lengths (e.g. MAGIC-x means the macro action length is x).

RMAG [16]. DESPOT with macro action learning boosted by recurrent neural networks. We compare against RMAG over different macro-actions lengths.

5.2 Experimental Setup

For evaluation, all variants of ROP-RAS3 are implemented in Cython following [28]. VAMP is implemented in C++ and is used via a Python API. We avoid implementing benchmark methods from scratch for a fairer comparison; the POMCP implementation is from [28] and the implementations of POMCP, MAGIC, RMAG and Ref-Basic all use the code implemented by their respective authors [15,16].

All methods are provided with the same planning time of **1s** across scenarios with the exception of Light-Dark where the planning time is **0.1s**. We also use the same discount factor ($\gamma = 0.99$) for all planners. The temperature parameter η for ROP-RAS3 and Ref-Basic is $\eta = 0.2$. MAGIC and RMAG are trained on a 4070 GPU with data collected across 500,000 runs (∼3 hours of training). Parameters have been tuned to optimize performance for each problem. Aforementioned sampling heuristics (see Section 4) are tested in all environments where appropriate. For the multi-drone environment, the sampling heuristic randomly samples a state from the belief. For that sampled state, the drone nearest to the target converges to it while other drones spread to a uniformly sampled free configuration.

5.3 Results and Discussions

We ran the method 30× for each scenario and method. The results are summarized in Tables 1, 2, and 3. Note that the number of episodes and steps recorded in the tables are average statistics of the methods regardless of success or failure. MAGIC and RMAG were only run on Light-Dark and Maze2D because they do not naturally extend to high-dimensional state spaces. POMCP and Ref-Basic failed on all runs of Random3D and Multi Drone Tag and are therefore excluded from the tables.

Table 1: Results on Light-Dark and Maze2D
(Light-Dark: x=4, y=8, z=16. Maze2D: x=8, y=16, z=24)

		Light-Dark				**Maze2D**			
	Sampling Heuristics	**# Episodes**	**Succ. %**	**E[Tot.Reward] (stdErr)**	**Steps**	**# Episodes**	**Succ. %**	**E[Tot.Reward] (stdErr)**	**Steps**
B-VAMP	entropy	N/A	76	70.2 (7.9)	46	N/A	40	-65.6 (68)	385
POMCP	N/A	282	77	72.6 (7.9)	42	314	0	-80 (0)	800
Ref-Basic	entropy	44	50	45.1 (9.3)	49	33	0	-807 (3.2)	75
R-POMCP	N/A	262	83	79 (6.9)	31	279	0	-146 (11)	784
MAGIC-X	N/A	N/A	88	85.0 (1.1)	29	N/A	0	-80 (0)	800
MAGIC-Y	N/A	N/A	85	81.9 (1.2)	31	N/A	0	-106 (17)	800
MAGIC-Z	N/A	N/A	78	75.9 (1.3)	35	N/A	1.1	-408 (12)	716
RMAG-X	N/A	N/A	88	84.8 (1.1)	29	N/A	0	-80 (0)	800
RMAG-Y	N/A	N/A	83	80.4 (1.2)	32	N/A	7.2	244 (13)	586
RMAG-Z	N/A	N/A	80	77.1 (1.3)	35	N/A	0	-80 (0)	800
ROP-RAS3	uniform	124	100	97.2 (0.1)	20	115	90	596 (57)	319
ROP-RAS3	distance	119	97	94.2 (3.3)	24	253	93	698 (38)	382
ROP-RAS3	entropy	26	100	97.0 (0.3)	23	46	100	761 (2.9)	293

The results indicate that all variants of ROP-RAS3 substantially outperform all baseline methods in all evaluation scenarios. The improvement provided by ROP-RAS3 is smallest for Light-Dark (the simplest evaluation scenario) where all variants of ROP-RAS3 achieved a success rate of up to 28% higher than R-POMCP, MAGIC, and RMAG. In the other scenarios, all variants of ROP-RAS3 improved the success rate of the benchmark methods by many folds. The reason is Light-Dark requires much lower planning horizon and has less uncertainty, compared to other scenarios. This simplicity is indicated by the good performance of methods that do not use macro actions, such as POMCP and Ref-Basic, and by the good performance of B-VAMP, which reasons with respect to only a sampled state of the belief.

However, when the scenario requires a much longer planning horizon (e.g., Maze2D) or has higher uncertainty and more complex geometric structures, the benefit of ROP-RAS3 increases. By using VAMP, ROP-RAS3 can quickly generate much more diverse macro actions that capture the geometric features of the problem well. Simultaneously, the reference-based POMDP solver enables ROP-RAS3 to utilize the sampled macro-actions more efficiently than other benchmark methods. Learning-based methods perform poorly in Maze2D due to the difficulty in learning suitable macro actions with fixed length. In contrast, by using SBMP, ROP-RAS3 is able to better capture the twists and turns the robot needs to perform to navigate the geometric features of the environment.

Table 2: Results on Random3D
(#1: 100 obstacles, 15 danger zones. #2: 200 obstacles, 10 danger zones.
#3: 300 obstacles, 10 danger zones. #4: 400 obstacles, 5 danger zones.)

	Sampling Heuristics	**Exp #**	**Ref. Policy Fail %**	**Success %**	**E[Tot.Reward] (stdErr)**	**Steps**	**Exp #**	**Ref. Policy Fail %**	**Success %**	**E[Tot. Reward] (stdErr)**	**Steps**
B-VAMP	N/A	1	N/A	0	-632 (52)	435	2	N/A	10	-307 (65)	526
R-POMCP	N/A		N/A	7	-89 (52.2)	741		N/A	7	-157 (64)	700
ROP-RAS3	uniform		6	60	236 (121)	287		12	63	224 (123)	319
ROP-RAS3	distance		6	63	266 (118)	272		12	60	204 (123)	373
ROP-RAS3	entropy		1	50	25.2 (130)	345		2	47	124 (116)	388
B-VAMP	N/A	3	N/A	3.3	-337 (70)	561	4	N/A	0	-243 (56)	750
R-POMCP	N/A		N/A	7	-158 (62.1)	665		N/A	0	-117 (28)	764
ROP-RAS3	uniform		15	63	297 (112)	463		24	67	353 (101)	504
ROP-RAS3	distance		17	67	367 (96.6)	452		18	50	237 (97)	505
ROP-RAS3	entropy		4	43	132 (105)	529		5	33	61.6 (97)	560

ROP-RAS3 is also robust to performance degradation from the underlying reference policies. We ran 4 variants of Random3D with an increasing number of obstacles. The results in Table 2 indicate that ROP-RAS3's performance (especially with uniform random sampling heuristics) does not degrade much even when the reference policy's (i.e. RRT-Connect) fail percentage increases due to increasingly narrower passages in the maze. This is because ROP-RAS3 uses the results of RRT-Connect only to provide a set of alternative sequences of actions to perform. As long as there is sufficient diversity of macro-actions (*i.e.*, sufficient support

of the reference policy), the reference-based POMDP planner can converge to a reasonable policy of the POMDP problem fast.

Table 3: Results on Multi-Drones

	Episodes	**Succ. %**	**Acc. Rewards**	**Steps**
B-VAMP	N/A	7.5	-1.43 (25)	390
R-POMCP	99	20	64 (38)	345
ROP-RAS3	106	89	423 (29)	171

For the high-dimensional planning problem—Multi-Drone Tag, ROP-RAS3 performs at least 4 times better than the rest of the methods as it is the only method that exhibits strategies where drones actively discover and surround the tag. Both B-VAMP and POMCP cannot easily adapt to the uncertainties from deterministic planning. B-VAMP does not incorporate any uncertainty in the effects of actions and POMCP fixes a set of macro actions from the reference policy for each belief node based only on a *single* sampled particle. Hence, the expanded set could be insufficient to fully represent the support of the belief, which is a problem that can be further compounded if the reference policy keeps failing.

Table 4: Ablation Study of ϵ-Exploration for ROP-RAS3 in Maze2D

ϵ	**success %**	**E[Tot.Reward] (stdErr)**	**#Steps**	ϵ	**success %**	**E[Tot.Reward] (stdErr)**	**#Steps**
0	100	761 (2.9)	293	0.1	87	603 (79)	378
0.2	95	699 (44)	314	0.3	87	573 (83)	374
0.4	87	579 (75)	444	0.5	80	503 (82)	460

One may be concerned that limiting sub-goal sampling to only the set of states where good observations or rewards can be gained is too restrictive. Therefore, we also evaluate ROP-RAS3 when the sub-goals are sampled in an ϵ-greedy fashion, where with probability ϵ ROP-RAS3 samples sub-goals from the entire state space and, with $(1-\epsilon)$ probability, it follows the dynamic entropy sampling heuristic mentioned in Section 4. We evaluated this ϵ-greedy sampling strategy on the Maze2D scenario Table 4. In this scenario, ROP-RAS3 performs better with lower ϵ. However, as ϵ increases, its performance does not drop by much.

6 Summary

This paper presents a new online approximate POMDP solver, Reference-Based Online POMDP Planning via Rapid State Space Sampling (ROP-RAS3), which uses VAMP to sample the state space and rapidly generate a large number of macro actions online. These macro-actions reduce the effective planning horizon

required and are adaptive to geometric features of the robot's free space. Since reference-based POMDP planners use macro actions only to bias belief-space sampling and do not require exhaustive enumeration of macro actions, ROPRAS3 can efficiently exploit many diverse macro actions to compute good POMDP policies fast. Evaluations on various long horizon POMDPs indicate that ROPRAS3 outperforms state-of-the-art methods by multiple factors. The method can potentially benefit from many learning paradigms and will be explored in the future.

7 Acknowledgements

YL, EK, and HK have been partially supported by the ANU Futures Scheme. YL has been supported by Australia RTP scholarship. HK has been partially supported by the SmartSat CRC. ZK, WT and LEK have been supported in part by NSF 2008720, 2336612, and Rice University Funds. WT has been supported by NSF ITR 2127309—CRA CIFellows Project.

References

1. M. G. Azar, V. Gómez, and H. Kappen. Dynamic policy programming with function approximation. In *Proc. of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 119–127, 2011.
2. M. G. Azar, V. Gómez, and H. Kappen. Dynamic policy programming. *JMLR*, 13:3207–3245, 2012.
3. P. Cai, Y. Luo, D. Hsu, and W. S. Lee. Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty. *IJRR*, 40(2-3):558–573, 2021.
4. G. Flaspohler, N. A. Roy, and J. W. Fisher III. Belief-dependent macro-action discovery in pomdps using the value of information. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *NeurIPS*, volume 33, pages 11108–11118. Curran Associates, Inc., 2020.
5. M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 2000.
6. R. He, E. Brunskill, and N. Roy. PUMA: planning under uncertainty with macro-actions. In M. Fox and D. Poole, editors, *AAAI*, 2010.
7. L. Kaebling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.
8. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
9. E. Kim, Y. Karunanayake, and H. Kurniawati. Reference-based POMDPs. In *NeurIPS*, 2023.
10. J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *ICRA*, volume 2, pages 995–1001, 2000.
11. H. Kurniawati. Partially observed Markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:254–277, 2022.

12. H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee. Motion planning under uncertainty for robotic tasks with long time horizons. *IJRR*, 30(3):308–323, 2011.
13. H. Kurniawati and V. Yadav. An online POMDP solver for uncertainty planning in dynamic environment. In *ISRR*, pages 611–629, 2013.
14. S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
15. Y. Lee, P. Cai, and D. Hsu. MAGIC: Learning Macro-Actions for Online POMDP Planning . In *RSS*, Virtual, July 2021.
16. Y. Liang and H. Kurniawati. Recurrent macro actions generator for POMDP planning. In *IROS*, pages 2026–2033, 2023.
17. S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *IJRR*, 29(8):1053–1068, 2010.
18. C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.
19. R. Platt Jr, R. Tedrake, L. P. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *RSS*, volume 2, 2010.
20. J. Schwartz, R. Newbury, D. Kulić, and H. Kurniawati. Posggym. https://github.com/RDLLab/posggym, 2023.
21. D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *NeurIPS*, volume 23. Curran Associates, Inc., 2010.
22. R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.
23. B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox. CuRobo: Parallelized collision-free robot motion generation. In *ICRA*, pages 8112–8119, 2023.
24. G. Theocharous and L. Kaelbling. Approximate planning in POMDPs with macro-actions. In S. Thrun, L. Saul, and B. Schölkopf, editors, *NeurIPS*, 2003.
25. W. Thomason, Z. Kingston, and L. E. Kavraki. Motions in microseconds via vectorized sampling-based planning. In *ICRA*, 2024.
26. E. Todorov. Linearly-solvable Markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *NeurIPS*, volume 19. MIT Press, 2006.
27. N. Ye, A. Somani, D. Hsu, and W. S. Lee. DESPOT: Online POMDP planning with regularization. *JAIR*, 58:231–266, 2017.
28. 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.