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,
{Edward.Kim, Yuanchu.Liang, hanna.kurniawati}@anu.edu.au
[2] Rice University, Houston TX 77005, USA,
{zak, wbthomason, 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. This paper proposes a new approximate online POMDP solver, called Reference-Based Online POMDP Planner via Rapid State Space Sampling (ROP-RAS3). ROP-RAS3 uses novel extremely fast sampling-based motion planning techniques to sample the state space and generate a diverse set of macro actions online which are then used to bias belief-space sampling and infer high-quality policies *without* requiring exhaustive enumeration of the action space—a fundamental constraint for modern online POMDP solvers. 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 *several times*.

Keywords: Motion Planning, Planning under Uncertainty, POMDP, Sampling-based Motion Planning, 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) [6,22] 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 [10], thanks in large part to sampling-based approaches. These

Authors with * have equal contributions

approaches relax the problem of finding an optimal solution to an approximate one by sampling of the belief space and computing the best strategies from only the sampled beliefs. Scalable anytime methods under this approach (surveyed in [10]) have been proposed for solving large POMDP problems. However, computing a good solution to long-horizon (*e.g.*, ≥ 15 steps) POMDPs remain difficult.

Early results [11] indicate that sampling based motion planning (SBMP)—sampling-based approaches designed for deterministic motion 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 alternative 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 [12,21,27] 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 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* [8] can be used to drop 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 Planner via Rapid State Space Sampling (ROP-RAS3). ROP-RAS3 uses VAMP to generate diverse macro actions online and uses these macro actions to bias belief space sampling for *Reference-Based* POMDP

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 [27,15] with learned macro actions—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.*, [7,9]) for solving *deterministic* motion planning problems [13]. 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 states, $q \in Q$.

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*—$Q \setminus Q_{\text{free}}$—states. 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 state, 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 states in Q_{free} by short, local motions. Once both q_I and Xg are connected by this structure, a valid robot motion plan can be found with graph search methods. The solutions may not be robust against uncertainties or changes in configuration spaces.

2.2 POMDP Background

An infinite-horizon POMDP is defined as the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, \mathcal{T}, R, \gamma \rangle$ where $\mathcal{S}$ denotes the set of all possible states of the agent, $\mathcal{A}$ denotes the set of all possible actions the agent can perform, and $\mathcal{O}$ denotes all the possible observations that the agent can perceive. The reward is a real-valued function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and is such that $|R(s,a)| \leq R_{\max}$ uniformly in $\mathcal{S} \times \mathcal{A}$. The parameter $\gamma \in (0,1)$ is the discount factor and sets the meaningful horizon of the POMDP.

In general, the agent does not directly observe the true state. So, at each time-step, the agent maintains a *belief* about its state—a probability distribution over the state space. More precisely, if b' denotes the agent's next belief after taking action a and receiving the observation o, then the updated belief is given by $b'(s') \propto \mathcal{Z}(o \,|\, s', a) \sum_{s \in \mathcal{S}} \mathcal{T}(s' \,|\, s, a)\, b(s)$ and we write $b' = \tau(b, a, o)$ with the belief update operator τ. The space of all possible beliefs is denoted by $\mathcal{B} := \Delta(\mathcal{S})$. For any given belief b and action a, the expected reward is given by $R(b,a) := \sum_{s \in \mathcal{S}} R(s,a) b(s)$. The quantity $P(o \,|\, a, b)$ is the probability that the agent perceives $o \in \mathcal{O}$, having performed the action $a \in \mathcal{A}$, under the belief $b \in \mathcal{B}$, and is intrinsic to the POMDP model.

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 $\pi(\cdot \,|\, b)$. A policy is *deterministic* if it only has support at a single point $a \in \mathcal{A}$. Let Π be the class of all (stochastic) policies and Π_d be the class of all deterministic policies. By definition, it is clear that $\Pi_d \subset \Pi$. 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, $V^\pi(b) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(b_t^\pi, A_t^\pi))]$. A *solution* to the POMDP is a deterministic policy $\pi^* \in \Pi_d$ satisfying $V^{\pi^*}(b) = \sup_{\pi \in \Pi_d} V^\pi(b)$ on $\mathcal{B}$.

The Bellman equation

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

is satisfied by the optimal value function V^{π^*}. Since the belief update is completely determined by an action-observation pair, each reachable belief can be identified with a *history* of action-observation sequences. Hence, (1) can be equivalently expressed by interchanging the belief b with a history h.

2.3 Long-Horizon POMDPs

Solving large long-horizon POMDPs remains a significant challenge due to the dual curses of dimensionality and history—*i.e.*, respectively, the exponential growth of the beliefs and histories with respect to the state space and time horizon. The former challenge is partially addressed by recently proposed sampling-based approximate solvers [12,21,27] that plan over action-observation histories and maintain a state particle estimate of the true belief at each node. However, the curse of history persists because of the exponential complexity associated with the belief space due to long planning horizons.

A common approach to evade the curse is to plan over *macro actions*—which are a pre-selected set of primitive action sequences [4,5,11,15,16,24]—and then to employ an out-of-the-box POMDP planner on the abstracted problem. Although this reduces the effective planning horizon, the approach is not without its limitations. The choice of class is critical and, depending on the method, automatic construction of such classes can require significant computational effort–*e.g.*, the learning time for MAGIC [15] can be on the order of hours. Further, irrespective of whether macro actions are used, most online POMDP planners [12,27,21] exhaustively enumerate all actions from each sampled belief in order to compute the gradient of the POMDP value function. Reference-Based POMDP Planners remove this limitation by solving a related POMDP optimization problem analytically, reducing the branching factor of the belief tree and the numerical computation in planning to estimating expectations over the state, action, and observation spaces.

3 POMDP Planning with SBMP-Generated Trajectories

We begin by presenting a general mechanism to integrate belief-space sampling of a typical online POMDP planner with state space sampling and macro-action generation via SBMP. To infer the best action, the POMDP planner samples a set of beliefs reachable from its current belief and represents them as a *belief tree*—i.e. a tree whose nodes represent beliefs (or, equivalently, histories) and whose edges represent action-observation pairs. Methods vary in the construction of the belief tree. Typically, state-of-the-art online planners are *particle-based* planners—*i.e.*, they maintain a value and a belief at each history node where beliefs are represented as sampled state *particles* at each node. The planner then searches the tree forward by judicious sampling—indeed, the choice of sampler is the main difference between planners. When an unvisited node is encountered during search, existing planners expand every action to receive a crude value estimate before propagating newly obtained information back to the root node via the Bellman equation.

The choice of sampler for forward search is the essential component for achieving high-quality policies online. Indeed, being able to exploit prior knowledge about policies that cover the solution is key [14]. The development of hardware-accelerated SBMP via SIMD-vectorization in [25] provides a powerful capability to do this. Being able to generate computationally cheap, high-quality trajectories

Algorithm 1 Online POMDP Planner with SBMP-Generated Macro Actions

1: Initialize search tree T complete with belief state particles
2: **while** time permitting **do**
3: $\quad T \leftarrow \text{SBMPExpandTree}(T)$
4: $\quad$ Re-estimate values on T and propagate back up to the root node
5: **end while**

SBMPExpandTree(T)

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

at frequencies in the range of kilohertz enables a planner to sample a belief and rapidly generate constraint-satisfying (*e.g.*, collision-free) paths to goals or to highly informative states; thus, promising macro actions can be dynamically created as a subroutine *within* a POMDP planner itself in fractions of a second. Algorithm 1 broadly outlines how accelerated SBMP can integrate with any off-the-shelf online particle-based POMDP planner. Although, in principle, any planner could be used to sample trajectories in Algorithm 1, the requirement for speed is critical; indeed, the speed makes it tractable to create a rich covering set of the POMDP's solution space fast, over which a planner can then optimize.

Moreover, Algorithm 1 can generalize a large class of existing online particle-based planners including (but not limited to) POMCP [21], DESPOT [27] and their derivatives [3,12,15,16]. While this procedure certainly enhances the power of these algorithms—as demonstrated by our results in Sect. 5—the requirement of existing planners to expand *all* actions before proceeding to deeper ones significantly hinders the scalability of Algorithm 1 to large-scale POMDPs with complex domains. Typically, the belief tree coverage is good for short search depths, but long-term information is seldom exploited and such planners perform poorly for long horizon problems. Unfortunately, the requirement of exhaustive enumeration seems baked in because of the form of the Bellman equation (1).

Notwithstanding this, the limitation can be softened by employing a *Reference-Based* POMDP *Planner* [8]. The underlying idea is to solve a POMDP whose reward objective is penalized for deviating too far from a given stochastic reference policy $\bar{\pi}(\cdot \mid b)$—in the context of Algorithm 1, this can be viewed as a SBMP-induced (macro) action sampler for any given belief (or history). The formulation yields an *analytic* Bellman equation which allows Reference-Based Planners to *always* search to a predefined depth by *sampling* the reference policy and *maintaining* the empirical backups rather than enumeratively maximizing value estimates.

Therefore, reference-based planners trade off algorithmic parsimony and long-horizon exploitation for pure reward maximization—see [8] and Sect. 4.1 for more discussion. Capitalizing on gains from SIMD-vectorized SBMP mitigates this trade-off by inducing reasonable (albeit sub-optimal) stochastic reference policies which can be deformed towards the solution by an online reference-based planner. As with all approximate solvers, this does not yield truly optimal rewards. However, policies generated by this procedure outperform benchmarks in complex domains under uncertainty and are reasonably robust with respect to the choice of reference. We emphasize that this modification is not negligible. Indeed, in Sect. 5, experimental results show that the performance deteriorates with problem complexity if the agent only executes policies derived from SBMP that do not account for uncertainty.

4 ROP-RaS3

In this section, we proceed to instantiate Algorithm 1 with a specific online anytime Reference-Based Planner—namely, Reference-Based Online POMDP Planner via Rapid State Space Sampling (ROP-RAS3). To this end, we motivate it by briefly recounting the *Reference-Based* POMDP formulation from [8] and adapt it to our context before outlining VAMP's capability to induce high-quality reference policies. The algorithm is then constructed on top of these techniques.

4.1 Reference-Based POMDPs

The concept of a *Reference-Based* POMDP was introduced in [8] as a generalization of the MDP formulations using KL-penalization in [2,26] to POMDPs. The formulation provides a framework to simplify approximate POMDP planning. Specifically, a *Reference-Based* POMDP 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* $\bar{\pi}$. The value $\mathcal{V}$ of a *Reference-Based* POMDP with respect to the stochastic policy $\bar{\pi}(\cdot \mid b)$ for a given $b \in \mathcal{B}$ satisfies the following Bellman equation

$$\mathcal{V}(b) = \sup_{\pi \in \Pi} \Big[R(b, \pi) - \frac{1}{\eta} \mathrm{KL}(\pi \,\|\, \bar{\pi}) + \gamma \sum_{a,o} P(o \mid a, b) \pi(a \mid b) \mathcal{V}\big(\tau(b, a, o)\big) \Big] \tag{2}$$

where $R(b, \pi) := \sum_{a,s} R(s, a) \pi(a \mid b) b(s)$ is the reward estimate. A *solution* is a stochastic policy $\pi \in \Pi$ that maximizes the value. A *Reference-Based* POMDP can therefore be viewed as a 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 (2) can be attained analytically by extending an argument of [1,2] to POMDPs. This yields an equivalent form of (2)—*i.e.*,

$$\mathcal{V}(b) = \frac{1}{\eta}\log\Big[\sum_a \bar{\pi}(a\,|\,b)\exp\Big\{\eta\big[R(b,a) + \gamma\sum_o P(o\,|\,a,b)\mathcal{V}\big(\tau(b,a,o)\big)\big]\Big\}\Big]. \quad (3)$$

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

$$\pi^*(a\,|\,b) \propto \bar{\pi}(a\,|\,b)\exp\Big\{\eta\Big[R(b,a) + \gamma\sum_o P(o\,|\,a,b)\mathcal{V}^*\big(\tau(b,a,o)\big)\Big]\Big\}. \quad (4)$$

The main point is that enumerative maximization can be avoided; instead, the solution to a *Reference-Based* POMDP can be approximated by successively iterating the analytic Bellman backup (3) exactly. This procedure is guaranteed to converge to a unique solution $\mathcal{V}^*$ from which the policy can be read off from (4). Unfortunately, requiring *exact* backups is a significant one, 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. Hence, there is no limitation to exhaustively enumerate actions and improving value estimates is extremely efficient as backups can be compute 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 4.2), leveraging them to rapidly sample high-quality deterministic policies to induce a reasonable partially observed reference policy.

4.2 Vector Accelerated Motion Planning

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 states in parallel. Functionally, this enables checking validity of a spatially distributed set of states over a candidate motion in parallel for the cost of a single collision check, massively lowering the expected time it takes to find colliding states 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.

4.3 ROP-RaS3

We are now in a position to describe ROP-RaS3 in detail. Algorithm 2 presents the pseudocode. ROP-RaS3 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 (this is largely handled by SIMULATE in Algorithm 2). When the required search depth is reached, ROP-RaS3 obtains a crude estimate of the root node's value (e.g. the A* goal distance). The planner then approximates the exact backup (function MAINTAINEXPECTATION in Algorithm 2) 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. (3). The above procedure is repeated until timeout, at which point the optimal policy's empirical estimate is read off and executed—the form of estimate is given by (4).

The subroutine SAMPLEMACROACTIONSBMP leverages VAMP as the reference policy to sample macro actions. Our implementation of ROP-RaS3 instantiates this subroutine with different strategies to select or design meaningful target points, towards which VAMP's planners can be used to rapidly generate a path. We emphasize that executing a reference policy designed only in this way performs poorly under uncertainty (see Sect. 5), the point being to improve this policy by planning under uncertainty. For environments with well-defined goal and informative states (e.g. landmarks), examples include the following:

- **Uniform** Uniformly sample goal configurations with a given probability; otherwise, sample an informative state configuration uniformly.
- **Distance**. Uniformly sample a goal configuration with 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 evaluate ROP-RaS3 on long-horizon POMDP problems systematically analysing the effects of its components—*i.e.*, VAMP and our instantiation of a reference-based planner—on its performance.

Algorithm 2 ROP-RAS3

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

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 $\log($MAINTAINEXPECTATION$(h, \vec{a}, \vec{o}, s', r))/\eta$

SAMPLEMACROACTIONSBMP(h, s)

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

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

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

5.1 Scenarios and Benchmark Methods

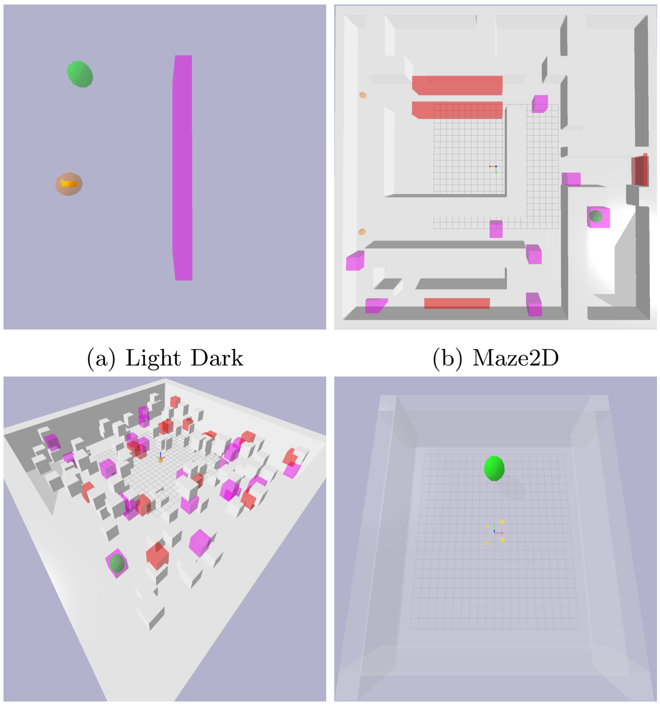
To achieve our aims, we evaluated ROP-RAS3 on 4 different scenarios:

Light Dark (Figure 1a). This is 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 and -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 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 [11]. 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 robot starts from one of the two spawn points 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 random 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 Teleporting Target (Figure 1d). This scenario 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 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 elswhere. 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



(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**)

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:

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 [8] (no VAMP).
POMCP [21]. A standard benchmark for online POMDP planning.
R-POMCP. 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 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++ but 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 the their respective authors [15,16].

All methods are provided with the same planning time of 1 second across scenarios with the except of Light Dark where the planning times 0.1s. We also use the same discount factor ($\gamma = 0.99$) for all planners. The temperature parameter η for the Reference-Based Planners ROP-RAS3 and Ref-Basic is $\eta = 0.2$. MAGIC and RMAG are trained on a 4070 GPU with data collected across 500000 runs ($\sim$3 hours of training). Parameters have been tuned to optimize performance for each problem. Aforementioned sampling strategies (see Section 4) are tested in all environments where it is appropriate. For the multi-drone environment, the sampler randomly samples a belief particle. For that particle, the drone nearest to the target converges to it while other drones spread to a uniformly sampled free configuration.

5.3 Results and Discussions

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			
Planners	**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	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 (0)	784
MAGIC-x	N/A	N/A	88	85.0 (1.1)	29	N/A	0	-80 (11)	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

For each scenario and each method, we ran the method 30×, and present the statistics of these runs in Table 1, Table 2, and Table 3. Note that MAGIC and RMAG were only run on Light Dark and Maze2D because they do not naturally extend to high-dimensional state spaces. Whilst, POMCP and Ref-Basic fails in

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.)

	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		N/A	0	-632 (52)	435		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	1	6	60	236 (121)	287	2	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		N/A	3.3	-337 (70)	561		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	3	15	63	297 (112)	463	4	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

Table 3: Results on Multi Drones Tag

	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

all runs of Random3D and Multi Drone Tag, and therefore excluded from the tables for brevity.

The results indicate that all variants of ROP-RAS3 substantially outperform all comparator methods in all evaluation scenarios.

The improvement provided by ROP-RAS3 is lowest in Light Dark (the simplest evaluation scenario). All variants of ROP-RAS3 achieves a success rate of up to 28% higher than R-POMCP, MAGIC, and RMAG, whilst in other scenarios, all variants of ROP-RAS3 improves the success rate of the comparator methods by many folds. The reason is Light Dark problem 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.

When the scenario requires 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. Whilst, Reference-based POMDP solver enables ROP-RAS3 to utilise much more diverse macro-actions efficiently, compared to other comparator 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 twist and turns the agent needs to perform due to geometric features of the environment.

ROP-RAS3 is also robust to performance degradation from the underlying reference policies. We ran 4 variants of Random3D with increasing number of obstacles. The results shown in Table 2 demonstrates that ROP-RAS3's performance (especially with uniform random sampling heuristics) do not degrade much even when the reference policy (i.e. RRT-Connect) fail percentage increases due to increasingly narrower passages in the maze. The reason is 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 (aka., sufficient support of the reference policy), the Reference-based POMDP solver component of ROP-RAS3 can converge to a good strategy for solving the POMDP problem fast.

For the high dimensional planning problem – Multi-Drone Tag, ROP-RAS3 performs at least 4 times better than the rest of the methods. It is the only method that exhibits strategies among drones to actively discover and surround the tag.

Both B-VAMP and POMCP cannot easily adapt to the uncertainties from deterministic planning as the former do not incorporate any probability calculations associated with actions and the latter fixes a set of macro actions from the reference policy for each belief node which could either perform poorly if the the reference policy keeps failing or the approximated values associated with the action is poor due to limited number of simulated episodes within a tight planning time.

Table 4: Ablation Study of Exploration Constants 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

Now, one may be concern 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 ϵ probability ROP-RAS3 samples sub-goals from the entire state space and $(1-\epsilon)$ probability, it follows the dynamic entropy sampling herusitcs mentioned in Section 4. We evaluate this ϵ-greedy sampling strategy on the Maze2D scenario. The results are shown in 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 Planner 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 thanks to SBMP. Since Reference-Based POMDP Planners use macro actions only to bias belief-space sampling and do not require exhaustive enumeration of macro actions, ROP-RAS3 can efficiently exploit many diverse macro actions to compute good POMDP policies fast. Evaluations on various long horizon POMDPs show that ROP-RAS3 outperforms state-of-the-art methods by multiple factors.

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. R. He, E. Brunskill, and N. Roy. PUMA: planning under uncertainty with macro-actions. In M. Fox and D. Poole, editors, *AAAI*, 2010.
6. L. Kaebling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.
7. 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.
8. E. Kim, Y. Karunanayake, and H. Kurniawati. Reference-based POMDPs. In *NeurIPS*, 2023.
9. J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *ICRA*, volume 2, pages 995–1001.
10. H. Kurniawati. Partially observed Markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:254–277, 2022.
11. 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.

12. H. Kurniawati and V. Yadav. An online POMDP solver for uncertainty planning in dynamic environment. In *ISRR*, pages 611–629, 2013.
13. S. M. LaValle. *Planning Algorithms*. Cambridge University Press.
14. W. Lee, N. Rong, and D. Hsu. What makes some POMDP problems easy to approximate? In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NeurIPS*, volume 20. Curran Associates, Inc., 2007.
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.
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.