Stochastic Motion Planning Under Partial Observability for Mobile Robots With Continuous Range Measurements

Ke Sun, *Student Member, IEEE*, Brent Schlotfeldt, *Student Member, IEEE*, George J. Pappas, *Fellow, IEEE*, and Vijay Kumar, *Fellow, IEEE*

***Abstract*—In this article, we address the problem of stochastic motion planning under partial observability, more specifically, how to navigate a mobile robot equipped with continuous range sensors, such as LIDAR. In contrast to many existing robotic motion planning methods, we explicitly consider the uncertainty of the robot state by modeling the system as a partially observable Markov decision process (POMDP). Recent work on general purpose POMDP solvers is typically limited to discrete observation spaces, and does not readily apply to the proposed problem due to the continuous measurements from LIDAR. In this article, we build upon an existing Monte Carlo tree search method, partially observable Monte Carlo planning (POMCP), and propose a new algorithm POMCP++. Our algorithm can handle continuous observation spaces with a novel measurement selection strategy. The POMCP++ algorithm overcomes overoptimism in the value estimation of a rollout policy by removing the implicit perfect state assumption at the rollout phase. We validate POMCP++ in theory by proving it is a Monte Carlo tree search algorithm. Through comparisons with other methods that can also be applied to the proposed problem, we show that POMCP++ yields significantly higher success rate and total reward.**

***Index Terms*—Monte Carlo tree search (MCTS), motion and path planning, partially observable Markov decision process (POMDP).**

I. Introduction

THE problem of motion planning shows up in almost all robotic applications, including navigating mobile vehicles in complex environments, manipulating robotic arms to grasp objects, controlling microrobots for cellular and chemical delivery, and others. Depending on the modeling assumptions, motion planning problems can, in general, be categorized into the following three broad classes.

1) *Deterministic motion planning*: Problems in this category assume a deterministic motion model, a known initial state, and a known environment, for which an open-loop control policy suffices. Such problems are well addressed by search-based algorithms, such as A* [1], or sampling-based algorithms, such as RRT, RRT*, and Probabilistic RoadMap (PRM)* [2].
2) *Stochastic model with perfect state information:* In this class, uncertainty of the motion is taken into account while perfect state information is assumed to be accessible. Receding horizon control (RHC) [3], combined with deterministic motion planning algorithms, directly lends itself to the generation of a feedback control policy for such problems, which implements the idea of certainty equivalent control [4, Ch. 6.2]. However, it requires one to include motion uncertainty explicitly in cost computation of paths in order to obtain an optimal feedback policy [5]–[7].
3) *Stochastic model with imperfect state information:* Compared with the last two classes, problems in this category provide the most general modeling of a robotic system, which include stochastic motion and measurement models and a prior probability distribution over the state. Practitioners typically extend the idea of the separation principle, and rely on a state observer to provide an accurate estimate of the state. Then, methods from the second category may be applied.

In this article, we consider problems in the third category, and specifically refer to such problems as *stochastic motion planning*. Existing practical approaches to solving this problem come with limitations. Failure to provide accurate state estimate can lead to catastrophic consequences, such as collisions. Furthermore, for estimation methods, such as visual inertial odometry [8], maintaining observability of the states requires active uncertainty reduction as a part of the planning. These limitations motivate the exploration of formal approaches to

Manuscript received June 17, 2020; revised November 5, 2020; accepted November 20, 2020. Date of publication December 24, 2020; date of current version June 4, 2021. This work was supported by the Army Research Laboratory under Grant ARL DCIST CRA W911NF-17-2-0181. This article was recommended for publication by Associate Editor S.-J. Chung and Editor A. Billard upon evaluation of the reviewers' comments. *(Corresponding author: Ke Sun.)*

The authors are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: sunke@seas.upenn.edu; brentsc@seas.upenn.edu; pappasg@seas.upenn.edu; vijay.kumar@seas.upenn.edu).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material consists of a video, viewable with VLC Media Player (3.0.4 Vetinari) on Ubuntu 18.04, demonstrating hardware experiments of the proposed POMCP++ in a hallway environment. The size of the video is 10.6 MB. Contact sunke@seas.upenn.edu for further questions about this work.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2020.3042129.

Digital Object Identifier 10.1109/TRO.2020.3042129

1552-3098 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

explicitly address the stochastic motion planning problem without resorting to certainty equivalence [9]–[12].

In the context of stochastic motion planning, partially observable Markov decision processes (POMDPs) [13] arise naturally since they provide a mathematical framework that couples stochastic motion and measurement models found in stochastic motion planning problems. Notwithstanding, the prohibitive computation cost in solving POMDP problems optimally prevents their widespread application. The technical challenge is twofold. First, a policy for POMDP problems is known to be a function of belief state instead of simply the state estimate. Even for a system with finite state space of size n, the belief space is continuous and resides in $\mathbb{R}^{n-1}$. Second, in order to find an optimal policy, one also has to consider all future outcomes, the number of which grows exponentially with the search depth. Pineau *et al.* [14] summarize the two challenges vividly as *curse of dimensionality* and *curse of history*.

Many efforts from robotics community have attempted to approximate the stochastic motion planning problem by modeling the systems as Linear systems with Quadratic cost and Gaussian noise (LQG) [9], [15]. Efficient algorithms exist for finding a control policy for such models. Two major limitations of LQG prevent its wide application to stochastic motion planning problems. First, LQG methods require a Gaussian state distribution. In many problems, the belief state of the robot may not be a unimodal Gaussian distribution, especially in ambiguous environments. Application of LQG methods also requires differentiable objective, dynamical, and measurement functions. It may not be natural to formulate the task as a differentiable objective function. More fundamentally, some widely used sensors on robotic platforms, such as LIDAR, lack an explicit and differentiable measurement model.

In the AI community, there is also a strong body of literature on POMDP solvers for general systems [16], [17], where only a simulator of the system, instead of explicit models, is required. Recent developments on tree search based general POMDP solvers, partially observable Monte Carlo planning (POMCP) [11] and determinized sparse partially observable tree (DESPOT) [18], have demonstrated effective solutions to large-scale problems (PocMan with 10^{56} states).

Two difficulties follow immediately in applying the general tree-based POMDP solvers, such as POMCP and DESPOT, to solve stochastic motion planning problems. First, the general POMDP solvers usually assume a (finite) discrete system, but robotic systems are continuous. The continuity of the state space should not pose a problem. Instead of being represented explicitly as a finite-dimensional vector, the belief of the state can be approximated with samples. The continuity of the action space can also be removed by considering only a finite set of motion primitives. However, it remains unclear how to handle the continuous observation space. A naive application of the existing tree search methods will result in an overly shallow tree, which is unlikely to produce informed actions that account for a longer horizon, as is necessary in planning problems.

The second difficulty, which is less obvious, comes from estimating the value of the rollout policy. Consider using deterministic motion planning methods as the rollout policy. During the rollout phase of tree search based POMDP solvers, one has to find an action sequence based on a sample at a leaf node; apply the actions to the same sample; use the simulated results to estimate the value of the rollout policy. This is equivalent to the assumption of a known state at the start of rollout phase, resulting in an overly optimistic estimated value, which can lead to incorrect action selection in the tree policy.

A. Contributions

In this article, we address the general robotics problem of stochastic motion planning problem with continuous measurements. We formulate the problem for ground vehicle equipped with a range sensor (LIDAR), whose task is to navigate to a goal on a known occupancy grid map. We note that the methods proposed here may apply more generally to other robot models and sensor types, but we focus on one problem instance for concreteness. We solve the navigation problem by building upon POMCP [11], a tree search method for the solution of general POMDPs, and proposing a new algorithm, POMCP++. Our major contributions are summarized as follows.

1) Our POMCP++ algorithm addresses the aforementioned difficulties by introducing two major improvements to POMCP. First, measurement likelihood of the original system is distorted so that existing measurement branches in the tree can be revisited with nontrivial probability. Second, a group of samples, instead of one, is enforced to traverse downstream the same set of nodes in the tree simultaneously. The belief at a leaf node can be well represented, resulting in more accurate value estimate of the rollout policy. The correct update of the values at the traversed nodes is still guaranteed through the application of importance sampling.
2) Because of the distorted measurement likelihood, the estimated value from the simulation episodes is biased for the current policy. Notwithstanding, we prove that such an estimate is unbiased as the number of simulations and the size of sample group tend to infinity. Therefore, the proposed POMCP++ algorithm is shown to be a valid Monte Carlo tree search (MCTS) algorithm.
3) We present simulation results that compare POMCP++ with existing approaches that can also be applied to the stochastic navigation problem. The results indicate that the POMCP++ algorithm yields a higher summation of discounted reward, the objective to be maximized. Furthermore, POMCP++ achieves a higher success rate while reducing the rate of collisions and premature stops outside the goal region. In addition to simulations, we demonstrate the performance of POMCP++ on a real robotic platform in an ambiguous hallway environment.

II. Related Work

When addressing stochastic motion problems, most of the works from robotics community assume the robotic system is LQG. The assumption of the separation principle in LQG allows control and estimation to be performed independently,

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

simplifying the problem. There is also an independent body of literature from the AI community trying to solve POMDP in its most general form. In this section, we review literature from these two communities separately.

Earlier works on LQG systems, such as Prentice and Roy [9] and Platt *et al.* [19], assume maximum likelihood measurements, which help in identifying a unique posterior distribution, but introduce additional information. Du Toit and Burdick [20] prove that maximum likelihood measurement introduces the least amount of information comparing with other future measurement sequences. Van den Berg *et al.* [15] first figure out the actual dynamics of the belief state of an LQG system. He shows that the covariance of the posterior Gaussian distribution is fixed for arbitrary future measurements, whereas the mean follows a Gaussian distribution. Van den Berg *et al.*'s finding in [15] helps remove the maximum likelihood assumption in later works, such as [10], [21]–[25].

The LQG approaches can also be categorized based on its origin of the deterministic counterpart. Censi *et al.* [26] solve the problem with graph search. Bry and Roy [27] propose RRBT by extending RRT*. More works are based on PRM since the belief roadmap proposed by Prentice and Roy [9]. A known problem of belief roadmap is the violation of the principle of optimality, due to the lack of strict ordering of covariance matrices. Censi *et al.* [26] and Bry and Roy [27] address this problem by imposing partial ordering for beliefs. Agha-Mohammadi *et al.* [22], [28] introduce "belief stabilizers," which assume the existence of a local controller at each node in PRM that can bring the belief at a node to its neighbor nodes representing other belief. Shan and Englot [29] avoid the necessity of partial ordering by considering the largest eigenvalue of a covariance matrix. In addition to the aforementioned graph search methods, there are algorithms trying to find a control policy through optimization. van den Berg *et al.* [10] and Sun *et al.* [24] directly solve for the feedback control policy through dynamic programming. Vitus and Tomlin [21], Indelman *et al.* [23], and Rafieisakhaei *et al.* [25] find the next best action and form a feedback policy through RHC.

Efforts from the AI community try to solve POMDP problems in their most general form without the LQG system assumption. For finite discrete systems, Pineau *et al.* [14] first propose offline point-based method, which forms the basis for many later extensions [16]. There are two main ideas embedded in the point-based methods that break the *curse of dimensionality* and the *curse of history*, respectively. In the case that an initial belief is known, it is believed that only a small subset of the belief space is reachable. Therefore, one can only consider the reachable belief space, which can be approximated by a set of belief samples. The second idea is to exploit the fact that the optimal value function for a finite discrete system is a convex piecewise linear function of the belief, first proved by Sondik [30]. Although the number of piecewise linear regions grows with search depth, one can only keep a fixed number of "α-vectors" in order to bound the computation complexity. Extensions of point-based methods, such as [31], [32], are usually different in how to choose the belief samples and update the "α-vectors."

Offline POMDP solvers estimate the value function over the entire belief space. In contrast, online POMDP solvers focus on estimating the value for the current belief and generating the next best action. Most of the online POMDP solvers are based on tree search [17], alternating three steps: find the most promising leaf node, expand the node, and update the parents of the leaf node through dynamic programming. Variations of online POMDP solvers [33], [34] are usually different in what heuristics is applied in finding the next best leaf node to expand. Developments in sampling-based online POMDP solvers, such as POMCP [11] and DESPOT [12], have demonstrated their effectiveness in solving large-scale problems. It is worth noting the subtle difference between POMCP and tree search methods. POMCP is based on MCTS, where the tree and the predefined rollout policy, together, construct a single policy. As the number of simulations increases, the policy represented by the tree may hopefully converge to the optimal policy. In contrast, tree search methods, such as DESPOT, embed all policies in the tree. The job of the algorithm is to identify the optimal policy by accurately estimating the values at the nodes in the tree.

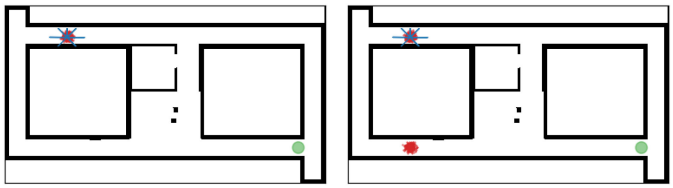
Most of the general POMDP solvers, either offline or online, are limited to finite discrete measurement space. A few works attempt to relax this limitation. Hoey and Poupart [35] propose the insight that observations should only be differentiated when they result in different posterior distributions, which provides a metric to partition large, or even continuous, observation space. Works in [36]–[38] try to solve the continuous POMDPs in general. Van den Berg *et al.* [36] generalize the work in [10], focusing on continuous POMDPs with Gaussian noise. Chaudhari *et al.* [37] propose a formal approach to discretize a continuous POMDP through incremental sampling. Sunberg and Kochenderfer [38] extend POMCP and utilize progressive widening [39] to handle both continuous action and observation spaces.

A few works try to apply general POMDP solvers for stochastic motion planning problems. Kurniawati *et al.* [40] propose milestone guided sampling, which is, at its root, a point-based method and is still limited to discrete problems. The specialty of milestone guided sampling is that it employs a PRM-like idea to guide the sampling of belief space. Lauri and Ritala [41] address the navigation of a ground vehicle with range sensors using POMCP. In [41], the continuous observation space is discretized by converting range sensor measurements to tuples consisting of cell indices and corresponding occupancy status. However, the number of possible discrete measurements is still huge, which, again, results in an overly shallow tree.

III. Problem Formulation

In this work, we address the problem of navigating a ground vehicle to a target location. Instead of assuming direct access to the perfect state information, the robot has to estimate the state through measurements from onboard range sensors, in this case, a 2-D LIDAR. We assume the robot is given a known 2-D occupancy grid map, which is necessary for computing measurement likelihood. Fig. 1 shows examples of such problem. Formally, the problem is described as follows.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 1. (a) and (b) Representative examples of stochastic motion planning problems, where the task is to navigate a ground vehicle to the target location (green patch) on an occupancy grid map. The state of the robot is unknown. However, the robot is provided with the initial belief (red particles) and has to update its belief through sparse range measurements (blue). (a) Configuration with the initial belief consisting of a single mode. The scenario in (b) is more challenging with two modes in the initial belief. The second scenario requires the robot to move out of the corridors in order to collapse the belief into a single mode distribution.

A. Motion Model

The discrete-time dynamics of the robot is modeled as in [42]

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} v_t\tau_t \cdot \mathrm{sinc}\left(\frac{\omega_t\tau_t}{2}\right)\cos(\theta_t + \frac{\omega_t\tau_t}{2}) \\ v_t\tau_t \cdot \mathrm{sinc}\left(\frac{\omega_t\tau_t}{2}\right)\sin(\theta_t + \frac{\omega_t\tau_t}{2}) \\ \omega_t\tau_t \end{pmatrix} \tag{1}$$

where $\boldsymbol{x} = (x, y, \theta)^\top \in \mathcal{X}$ represents the position and orientation, i.e., *state*, of the robot, whereas $\boldsymbol{a} = (v, \omega, \tau)^\top \in \mathcal{A}$ represents the velocity command, *action*. Furthermore, we note that the *belief* of the robot state is defined as, $\boldsymbol{b} \in \mathcal{B} : \mathcal{X} \mapsto \mathbb{R}$, $\boldsymbol{b}(\boldsymbol{x}) := p(\boldsymbol{x})$. To model the uncertainty of system dynamics, the input command $\boldsymbol{a}$ is distinguished from the executed command $\tilde{\boldsymbol{a}} = \tilde{v}, \tilde{\omega}, \tilde{\tau}$

$$\begin{gathered} \tilde{v} = v + n_v,\ \tilde{\omega} = \omega + n_\omega + n_\gamma,\ \tilde{\tau} = \tau \\ n_v \sim \mathcal{N}(0, \alpha_v(v)^2 + \beta_v(\omega)^2) \\ n_\omega \sim \mathcal{N}(0, \alpha_\omega(v)^2 + \beta_\omega(\omega)^2) \\ n_\gamma \sim \mathcal{N}(0, \alpha_\gamma(v)^2 + \beta_\gamma(\omega)^2) \end{gathered}$$

where $\alpha_{\{\alpha,\omega,\gamma\}}$ and $\beta_{\{\alpha,\omega,\gamma\}}$ are empirical parameters. In our work, we restrict the allowable actions to a finite subset $\mathcal{A}_m \subseteq \mathcal{A}$, i.e., *motion primitives*, of the action space in order to reduce the complexity of the problem.

B. Measurement Model

Measurements, $\boldsymbol{z} \in \mathcal{Z}$, are ranges of the beams produced by a 2-D LIDAR. Assuming the range of each beam, z_i $i = 0, 1, \ldots$ is independent, the measurement likelihood of $\boldsymbol{z}$ is the product of the probability density function (pdf) of z_i's, i.e.,

$$p(\boldsymbol{z}|\boldsymbol{x}, m) = \prod_i p(z_i|\boldsymbol{x}, m) \tag{2}$$

where $m : \mathcal{X} \mapsto \{0, 1\}$ is a known occupancy grid map. Therefore, it is sufficient to model a single beam. In this work, we use the beam model from [42]

$$p(z|\boldsymbol{x}, m) = \begin{pmatrix} z_h \\ z_s \\ z_m \\ z_r \end{pmatrix}^\top \cdot \begin{pmatrix} p_h(z|\boldsymbol{x}, m) \\ p_s(z|\boldsymbol{x}, m) \\ p_m(z|\boldsymbol{x}, m) \\ p_r(z|\boldsymbol{x}, m) \end{pmatrix}$$

where

$$\begin{aligned} p_h(z|\boldsymbol{x}, m) &= \begin{cases} \eta_h \cdot \mathcal{N}(z^*, \sigma_h^2), & \text{if } 0 \le z \le z_{\max} \\ 0, & \text{otherwise} \end{cases} \\ p_s(z|\boldsymbol{x}, m) &= \begin{cases} \eta_s \cdot \lambda_s \exp(-\lambda_s z), & \text{if } 0 \le z \le z^* \\ 0, & \text{otherwise} \end{cases} \\ p_m(z|\boldsymbol{x}, m) &= \delta(z - z_{\max}) \\ p_r(z|\boldsymbol{x}, m) &= \begin{cases} \frac{1}{z_{\max}}, & \text{if } 0 \le z \le z_{\max} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

are four different types of measurement errors due to Gaussian noise, unexpected objects, sensor failure, and unexplainable noise. The probability of each noise type is captured by (z_h, z_s, z_m, z_r) with $z_h + z_s + z_m + z_r = 1$. In (3), z^* and $z_{\max}$ are the nominal and maximum range measurement. For p_h and p_s, σ_h is the standard deviation for the normal distribution, λ_s is the rate of the exponential distribution, and $\eta_{\{h,s\}}$ are the normalization factors. For p_m, $\delta(\cdot)$ is the Dirac delta function.

C. Reward Model

It is necessary to describe the desired behavior through reward functions. To encode desired constraints on safety, energy, and total number of actions in our stochastic navigation task, we define a stage reward function as follows:

$$r(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{x}_{t+1}) = \begin{cases} -5 & \text{if } m(\boldsymbol{x}_{t+1}) = 1 \\ -5 & \text{if } \boldsymbol{a}_t = \boldsymbol{0} \text{ and } d(\boldsymbol{x}_{t+1}, \boldsymbol{x}_g) > \epsilon_g \\ 0 & \text{if } \boldsymbol{a}_t = \boldsymbol{0} \text{ and } d(\boldsymbol{x}_{t+1}, \boldsymbol{x}_g) \le \epsilon_g \\ -1 & \text{otherwise} \end{cases}$$

$$d(\boldsymbol{x}_{t+1}, \boldsymbol{x}_g) = \sqrt{\|x_{t+1} - x_g\|^2 + \|y_{t+1} - y_g\|^2} \tag{4}$$

where $\boldsymbol{x}_g$ is the target state. Cost is induced if the robot keeps moving, which encourages fewer actions. Colliding with obstacles, or stopping outside the target location, will introduce additional cost. The accumulation of cost only stops if the robot determines to stop within the target region, which is defined by ϵ_g. Note the orientation of $\boldsymbol{x}_g$ is ignored in determining task completion.

Problem (Stochastic Motion Planning): Given the initial belief $\boldsymbol{b}_0 \in \mathcal{B}$ of a robot, find a control policy $\pi^* : \mathcal{B} \mapsto \mathcal{A}_m$, which maximizes the discounted infinite horizon summation of the stage reward

$$V_{\pi^*}(\boldsymbol{b}_0) = \max_{\pi \in \Pi} \mathop{\mathbb{E}}_{\boldsymbol{x}_t, \boldsymbol{z}_t t=0,1,\ldots} \left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{x}_{t+1}) \right]$$

subject to the motion and measurement models given in (1) and (2), where $\gamma \in (0, 1)$ is the discount factor.

We assume the availability of an estimator able to track the belief of the robot at each time step. In the following of this work, a particle filer [43] is used to represent the belief using samples of different weights.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

IV. Background

In this section, we briefly overview POMDP [4, Ch.4] and POMCP [11]. The stochastic motion planning problem, considered in this work, is concerned with seeking a control policy for a partially observable system, which can be modeled by POMDPs. POMCP is an extension of MCTS [44] to solve POMDPs. The proposed algorithm in this work modifies POMCP addressing stochastic motion planning with continuous measurements.

A. Partially Observable Markov Decision Process

In POMDP, a system is modeled with an eight-element tuple, $\mathcal{X}, \mathcal{A}, \mathcal{Z}, t, o, r, b_0$, and γ. $\mathcal{X}$, $\mathcal{A}$, and $\mathcal{Z}$ are the state, action, and observation spaces. $t : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \mapsto \mathbb{R}$, $t(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{x}') := p(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{a})$, is the transition probability. $o : \mathcal{X} \times \mathcal{Z} \mapsto \mathbb{R}$, $o(\boldsymbol{x}, \boldsymbol{z}) := p(\boldsymbol{z}|\boldsymbol{x})$, is the measurement likelihood. $r : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \mapsto \mathbb{R}$ is a reward function.[1] In this work, we assume $r(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{x}')$ is bounded, i.e., $|r(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{x}')| < \infty$. $\boldsymbol{b}_t \in \mathcal{B} : \mathcal{X} \mapsto \mathbb{R}$, $\boldsymbol{b}_t(\boldsymbol{x}) := p(\boldsymbol{x}_t)$, represents the distribution of the state at time t, whereas $\mathcal{B}$ is the belief space. Specially, $\boldsymbol{b}_0 \in \mathcal{B}_0$ is the initial state distribution. $\gamma \in (0, 1)$ is a discount factor.

The goal in POMDP is to find a control policy $\pi := \{\mu_0, \mu_1, \ldots, \mu_{N-1}\} \in \Pi$, where $\mu_t : \mathcal{B} \mapsto \mathcal{A}$, such that the discounted summation of reward is maximized,[2] i.e.,

$$V_\pi(\boldsymbol{b}_0) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x_t}, \boldsymbol{z}_t \\ t=0,1,\ldots,N-1}} \left\{ \sum_{t=0}^{N-1} \gamma^t r(\boldsymbol{x}_t, \mu_t(\boldsymbol{b}_t), \boldsymbol{x}_{t+1}) \right\}$$

$$\pi^* = \arg\max_{\pi \in \Pi} V_\pi(\boldsymbol{b}_0), \quad V^*(\boldsymbol{b}_0) = V_{\pi^*}(\boldsymbol{b}_0).$$

In the case that the discounted summation of reward is over infinite horizon, there exists an optimal stationary policy, $\pi^* = \{\mu, \mu, \ldots\}$, which maximizes $\lim_{N\to\infty} V_\pi(\boldsymbol{b}_0)$ [45, Ch.1].

B. Partially Observable Monte Carlo Planning

Algorithm 1 shows the framework of POMCP. Some notations in Algorithm 1 are defined as follows. $h := (\boldsymbol{a}_0, \boldsymbol{z}_0, \ldots)$ represents historical information containing a sequence of actions and measurements. Together with $\boldsymbol{b}_0$, h uniquely determines the belief of the state, i.e., a node in the tree. Appending actions and measurements to h, such as $h\boldsymbol{a}$ or $h\boldsymbol{a}\boldsymbol{z}$, represents a node in the subtree of h. Specially, $h = \emptyset$ is the root node. h's in the form of $(\boldsymbol{a}_0, \boldsymbol{z}_0, \ldots, \boldsymbol{z}_t)$ ended with a measurement represent the belief nodes, whereas those ended with an action represent the belief-action nodes. Functions $V(\cdot)$ and $N(\cdot)$ keep track of the estimated value and the number of visits of a node.

For each iteration of the algorithm, a state sample $\boldsymbol{x}_0$ is drawn from the initial belief $\boldsymbol{b}_0$ (Algorithm 1 Line 5). The state sample is first simulated recursively following the *tree policy* (Algorithm 1, Lines 7 and 31). Once a leaf node in the tree is met, it is expanded (Algorithm 1, Line 20). The sample is forward simulated following the predefined *rollout policy* (Algorithm 1, Line 23) as a continuation. The traversed nodes are updated based on the simulation results (Algorithm 1, Line 34–36). The tree policy is, therefore, implicitly updated due to the addition of the new leaf nodes and the update of the corresponding parent nodes.

Algorithm 1: POMCP.

```
   Function a = search(b_0)
       h = NULL
       while not termination() do
5          x_0 ← draw one sample w.r.t. b_0
7          simulate(x_0, h, 0)
       end
       return arg max_a V(ha)
   end
12 Function a = selectAction(h)
       // c is a constant in UCB1 trading-off
       // exploitation and exploration.
       return arg max_a V(ha) + c · sqrt(log N(h) / N(ha))
   end
   Function r = simulate(x, h, d)
       if γ^d < ε then return 0
       if h is NULL then
           foreach a ∈ A do
20             ha ← (N_init, V_init)
           end
23         r = rollout(x, d)
           return r
       end
       a ← selectAction(h)
       x, r_s ← process(x, a)
29     z ← sampleMeasurement(x)
31     r_t ← simulate(x, haz, d + 1)
       r ← r_s + γ · r_t
34     N(h) ← N(h) + 1
36     N(ha) ← N(ha) + 1
38     V(ha) ← V(ha) + 1/N(ha) (r − V(ha))
   end
```

V. POMCP++

Algorithm 2 shows the proposed algorithm POMCP++, with a graphical illustration in Fig. 2. The overall procedure in POMCP++ follows POMCP in Algorithm 1. One obvious modification is the change from UCB1 [46] (Algorithm 1, Line 12) to ϵ-greedy [47, Ch. 5] (Algorithm 2, Line 29) in action selection, both of which trades off exploitation versus exploration. Kuleshov and Precup [48] show experimentally that the relative performance of the two methods vary with problem setups. In this work, ϵ-greedy policy is applied in order to ease theoretical analysis. Other major changes of POMCP++ include the following.

1) Replacing forward sampling measurements with measurement selection (Algorithm 2, Line 38).

[1] The reward function can be also defined as a function of state and action only, i.e., $r : \mathcal{X} \times \mathcal{A} \mapsto \mathbb{R}$.

[2] The posterior distribution of the state is one possible sufficient statistic for the feedback control policy [4, Ch. 4].

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2: POMCP++

Function $\boldsymbol{a}$ = search($\boldsymbol{b}_0$)
  h = *NULL*
  while *not* termination() **do**
5 $\mathcal{S} \leftarrow$ draw K samples w.r.t $\boldsymbol{b}_0$
    simulate($\mathcal{S}$, h, 0)
  end
  return $\arg\max_{\boldsymbol{a}} V(h\boldsymbol{a})$
end

11 **Function** $\boldsymbol{r}$ = rollout($\mathcal{S}$, d)
  if $\gamma^d < \epsilon$ **then return** $\mathbf{0}$
  sample $\boldsymbol{x} \sim \mathcal{S}$ w.r.t $\boldsymbol{w}$
  $\{\boldsymbol{a}_r\} \leftarrow$ AnytimeA*($\boldsymbol{x}$)
  $\boldsymbol{r} \leftarrow \mathbf{0}$
  foreach $\boldsymbol{x}_i, r_i \in \mathcal{S}, \boldsymbol{r}$ **do**
    foreach $\boldsymbol{a}_{r,j} \in \{\boldsymbol{a}_r\}$ **do**
      if $\gamma^{j+d} < \epsilon$ **then** break
      $\boldsymbol{x}_i, r \leftarrow$ process($\boldsymbol{x}_i$, $\boldsymbol{a}_{r,j}$)
      $r_i \leftarrow r_i + \gamma^j \cdot r$
    end
    if $\boldsymbol{x}_i$ *is not at the target location* **then**
      // $\bar{r}$ is the average stage reward.
      $r_i \leftarrow r_i + \gamma^{len(\boldsymbol{a}_d)} \cdot \bar{r}$
    end
  end
  return $\boldsymbol{r}$
end

29 **Function** $\boldsymbol{a}$ = selectAction(h)
  // random() generates random numbers from
  // the unit interval with uniform distribution.
  if random() $> \epsilon_a$ **then**
    $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}} V(h\boldsymbol{a})$
  else
    uniformly choose $\boldsymbol{a} \in \mathcal{A}$
  end
  return $\boldsymbol{a}$
end

38 **Function** $\boldsymbol{z}$ = selectMeasurement($h\boldsymbol{a}$, $\mathcal{S}$)
  if random() $< \left(|C(h\boldsymbol{a})|+1\right)^{\epsilon_z}$ **then**
    sample $\boldsymbol{x} \sim \mathcal{S}$ w.r.t $\boldsymbol{w}$
    $\boldsymbol{z} \leftarrow$ sampleMeasurement($\boldsymbol{x}$)
  else
    uniformly choose $\boldsymbol{z}$ s.t. $h\boldsymbol{a}\boldsymbol{z} \in C(h\boldsymbol{a})$
  end
  return $\boldsymbol{z}$
end

Function $\boldsymbol{r}, \boldsymbol{w}$ = simulate($\mathcal{S}$, h, d)
  if $\gamma^d < \epsilon$ **then return** $\mathbf{0}, \mathbf{1}$
  if h *is NULL* **then**
    foreach $\boldsymbol{a} \in \mathcal{A}_m$ **do**
      $h\boldsymbol{a} \leftarrow (N_{init}, V_{init})$
    end
    $\boldsymbol{r}$ = rollout($\mathcal{S}$, d)
    return $\boldsymbol{r}, \mathbf{1}$
  end
  $\boldsymbol{a} \leftarrow$ selectAction(h)
  $\mathcal{S}, \boldsymbol{r}_s \leftarrow$ process($\mathcal{S}$, $\boldsymbol{a}$)
  $\boldsymbol{z} \leftarrow$ selectMeasurement($h\boldsymbol{a}$, $\mathcal{S}$)
  $\boldsymbol{w}_s \leftarrow$ measurementLikelihood($\mathcal{S}$, $\boldsymbol{z}$)
  $\boldsymbol{w} \leftarrow \{w_i\}$ s.t. $(\boldsymbol{x}_i, w_i) \in \mathcal{S}$
  foreach $w_i, w_{s,i} \in \boldsymbol{w}, \boldsymbol{w}_s$ **do**
    $w_i \leftarrow w_i \cdot w_{s,i}$
  end
  $\boldsymbol{r}_t, \boldsymbol{w}_t \leftarrow$ simulate($\mathcal{S}$, $h\boldsymbol{a}\boldsymbol{z}$, $d+1$)
  $\boldsymbol{r} = \boldsymbol{r}_s + \gamma \cdot \boldsymbol{r}_t$
67 **foreach** $w_i, w_{s,i}, w_{t,i} \in \boldsymbol{w}, \boldsymbol{w}_s, \boldsymbol{w}_t$ **do**
    $w_i \leftarrow w_{s,i} \cdot w_{t,i}$
  end
  $\boldsymbol{w} \leftarrow \boldsymbol{w}/\|\boldsymbol{w}\|_2$
72 $r \leftarrow \boldsymbol{w}^\top \boldsymbol{r}$
  $N(h\boldsymbol{a}) \leftarrow N(h\boldsymbol{a}) + 1$
  $V(h\boldsymbol{a}) \leftarrow V(h\boldsymbol{a}) + \frac{1}{N(h\boldsymbol{a})}(r - V(h\boldsymbol{a}))$
  return $\boldsymbol{r}$, $\boldsymbol{w}$
end

2) Using a group of samples (Algorithm 2, Line 5), instead of a single one, to approximate belief while ensuring correct update of nodes through importance sampling.
3) Estimating the value of the rollout policy (Algorithm 2, Line 11) using the sample group in order to overcome overoptimism.

Details of the three major changes in POMCP++ are discussed in the following sections.

In addition to the notations defined for Algorithm 1, a few more are introduced in Algorithm 2. $\mathcal{S} := \{(\boldsymbol{x}, w)\}$ is a set consisting of pairs of states and corresponding weights, approximating the belief. Function $C(\cdot)$ returns the child nodes of the input node, whereas $|C(\cdot)|$ represent the number of child nodes.

A. Measurement Selection

In POMCP, measurements are forward sampled (Algorithm 1, Line 29) to determine whether to traverse along the tree or expand a new node. It is valid for cases of small-size finite discrete measurement space or large ones with concentrated distributions. In such cases, repeated measurements can be forward sampled with high probability, leading to the growth of the tree in depth. However, it is no longer true for continuous measurement space. The probability of generating a repeated measurement is 0 for a general continuous measurement pdf.

To overcome the issue, our proposed algorithm POMCP++ uses a new measurement selection strategy, which distorts the measurement likelihood and enforce the selection of existing measurements. In the function of measurement selection (Algorithm 2, Line 38), a new measurement branch is only created with probability $(|C(h\boldsymbol{a})|+1)^{\epsilon_z}$. Recall that $|C(h\boldsymbol{a})|$ is the number of child nodes, i.e., existing measurement branches, for the belief-action node $h\boldsymbol{a}$. Otherwise, an existing measurement is enforced. With $\epsilon_z < 0$, a new measurement is less likely to be generated as $C(h\boldsymbol{a})$ grows.

A similar notion of measurement selection appears in POMCPOW [38]. As action selection, measurement selection in [38]

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

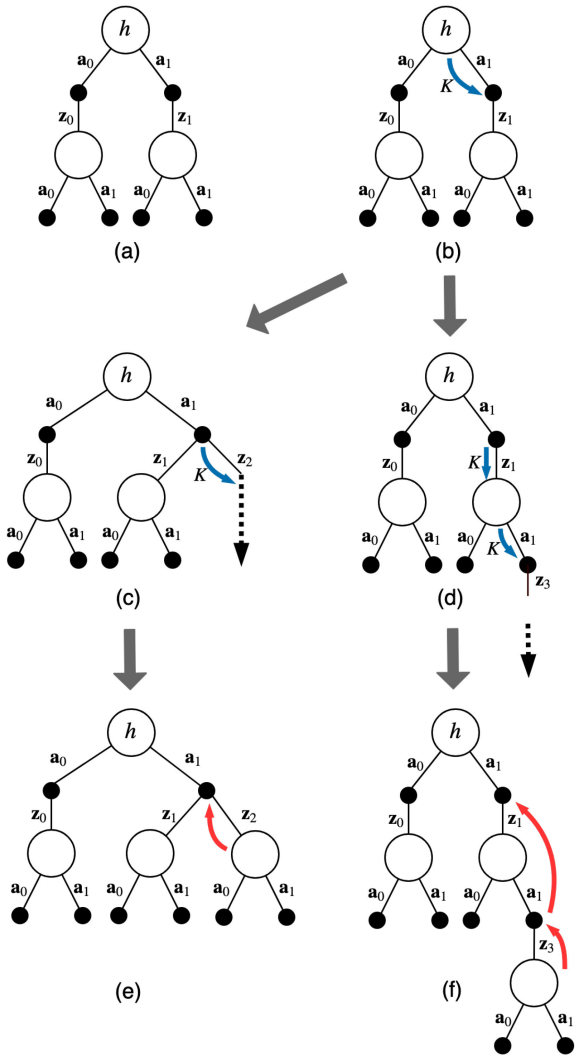


Fig. 2. (a) Starting tree configuration with two possible actions, $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$, rooted at h. "◯" and "•" denote the belief and belief action nodes, respectively. (b) assumes K samples are drawn from the initial belief and $\boldsymbol{a}_1$ is selected. The flow of the samples are marked by "\boldmath $\rightarrow$." After forward simulating the motion model, different cases happen depending on whether (c) a new measurement branch should be created or (d) an existing measurement branch should be followed. In the case of creating a new measurement branch as in (c), rollout policy is followed afterward, denoted as "$\cdots$." (e) Creation of the new leaf nodes. The simulation results are back propagated to the belief action node, shown as "\boldmath $\rightarrow$." The difference in (d) and (f) is that action and measurement selection procedure are repeated until a leaf node is encountered. The simulation results are, again, backpropagated to all traversed belief action nodes.

favors measurement branches that are often visited. This may result in a biased estimation of the policy value. In other words, the estimated policy value obtained by following the tree may not be a valid estimation of the actual value obtained by applying the policy to the real system. The underlying reason is that, unlike actions, the robot has no control over what measurement should be acquired in the future. Future measurement is completely determined by the measurement likelihood. In principle, measurement should be forward sampled to ensure that "measurement selection" exactly follows the measurement likelihood. However, this leads to an overly shallow tree, the problem to be addressed in the first place.

In POMCP++, an unbiased policy value estimation is guaranteed by two properties of the proposed measurement selection strategy. First, for each belief action node, new measurement branches are created *infinitely often (i.o.)* as the number of simulation episodes goes to infinity. Second, existing measurement branches are uniformly selected in determining the branch to follow further. These two properties ensure the statistics obtained through existing measurement branches serve as unbiased estimates for the ones obtained with the measurement likelihood. Appendix A provides a proof for the first point, and Appendix B provides a proof for the second.

B. Importance Sampling

Albeit the enforcement in selecting measurement branches, the selected measurement sequence may not be representative for a single sample from the initial belief. Instead, K samples are drawn from the initial belief (Algorithm 2, Line 5). All of the K samples are pushed downstream the tree, traversing the same set of nodes. Simply averaging the simulation results leads to inaccurate policy evaluation, since each initial state sample has different likelihood of generating the measurement sequence in the traversed path.

The theory of importance sampling comes to the rescue. Assume the selected action and measurement sequence following the belief node h_t is $\boldsymbol{a}_t, \boldsymbol{z}_t, \boldsymbol{a}_{t+1}, \boldsymbol{z}_{t+1}, \ldots$. Then, the current policy value at the belief action node $h_t\boldsymbol{a}_t$ is

$$\mathbb{E}_{X_t}\{R(X_t, A_t, Z_t)\} = \int_{X_t} R(X_t, A_t, Z_t) p(X_t|A_t, Z_t) dX_t$$

where $X_t := \boldsymbol{x}_{t:\infty}$, $A_t := \boldsymbol{a}_{t:\infty}$, and $Z_t := \boldsymbol{z}_{t:\infty}$ for simplicity. $R(\cdot)$ computes the discounted summation of reward of a simulation episode. Applying Bayes rule

$$p(X_t|A_t, Z_t) = \frac{p(Z_t|X_t, A_t)p(X_t|A_t)}{p(Z_t|A_t)}.$$

In the spirit of importance sampling, $p(X_t|A_t)$ is the proposal distribution, whereas $p(X_t|A_t, Z_t)$ is the target distribution. $p(X_t|A_t)$, the proposal distribution, is approximated by forward sampling. The measurement likelihood $p(Z_t|X_t, A_t)$ serves as the importance weight. $p(Z_t|A_t)$, as the normalization factor, is constant for all state sequences. Therefore, the current policy value at node $h_t\boldsymbol{a}_t$ can be approximated with

$$\frac{1}{\eta}\sum_{i=0}^{K-1} p(Z_t|X_t^i, A_t) \cdot R(X_t^i, A_t, Z_t) \tag{5}$$

with $\eta = \sum_i p(Z_t|X_t^i, A_t)$ (Algorithm 2, Lines 67–72).

C. Rollout Policy

It is desirable to take advantage of the well-developed deterministic motion planning methods to design the rollout policy. However, since such deterministic motion planning methods are usually state-dependent, direct application of them in the framework of POMCP or POMCPOW leads to overoptimism. More specifically, in POMCP and POMCPOW, a single state sample, $\boldsymbol{x}_0$, is simulated from the root to a leaf node to obtain $\boldsymbol{x}_t$,

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

which is then used to find a deterministic action sequence $\boldsymbol{a}^r_{t:T}$. T is implicitly determined by the length of the deterministic action sequence. In the case that a feasible rollout policy cannot be found, T can be set large enough so that $\gamma^T \ll 1$ and the reward-to-go beyond T can be ignored. The reward-to-go of the rollout policy is estimated with $R(\boldsymbol{x}_{t:T}, \boldsymbol{a}^r_{t:T})$, whose actual value is

$$\mathop{\mathbb{E}}_{p(\boldsymbol{x}_{t:T}|\boldsymbol{a}^r_{t:T})}\{R(\boldsymbol{x}_{t:T}, \boldsymbol{a}^r_{t:T})\} = \int_{\boldsymbol{x}_{t:T}} R(\boldsymbol{x}_{t:T}, \boldsymbol{a}^r_{t:T}) p(\boldsymbol{x}_{t:T}|\boldsymbol{a}^r_{t:T}) d\boldsymbol{x}_{t:T}.$$

Note that $\boldsymbol{x}_t$ may be from belief with high uncertainty at the leaf node. Since $\boldsymbol{a}^r_{t:T}$ is found using $\boldsymbol{x}_t$ in the first place, the rollout action sequence is expected to obtain high reward-to-go starting from $\boldsymbol{x}_t$. However, $\boldsymbol{a}^r_{t:T}$ may not perform well with states that are significantly different from $\boldsymbol{x}_t$. Therefore, $\mathbb{E}_{p(\boldsymbol{x}_{t:T}|\boldsymbol{a}^r_{t:T})}\{R(\boldsymbol{x}_{t:T}, \boldsymbol{a}^r_{t:T})\}$ may be much less than $R(\boldsymbol{x}_{t:T}, \boldsymbol{a}^r_{t:T})$. If $R(\boldsymbol{x}_{t:T}, \boldsymbol{a}^r_{t:T})$ is then back propagated to the parent nodes, the high reward-to-go may mislead action selection in later steps to favor this branch albeit the high uncertainty at the leaf node. In other words, the implicit assumption of perfect state information of $\boldsymbol{x}_t$ should be accused for the overoptimism.

Because of the measurement selection and importance sampling introduced previously, one has K state samples when reaching a leaf node. One of the K samples, $\boldsymbol{x}_t$, is chosen and used to find the open-loop action sequence $\boldsymbol{a}^r_{t:T}$. The estimated reward for the rollout policy is then $\sum_{i=0}^{K-1} R(\boldsymbol{x}^i_{t:T}, \boldsymbol{a}^r_{t:T})$ (Algorithm 2, Line 11). In this work, we use A^*, with inflation factor of the heuristic set to infinity [49], to find the rollout policy. Although the action sequence could be suboptimal, it ensures maximum efficiency of the rollout policy construction.

Removal of the implicit perfect state information assumption at the leaf nodes helps address the underlying active localization problem. Active localization requires the robot to actively reduce the state uncertainty in order to complete the task with high confidence. Since the action sequence $\boldsymbol{a}^r_{t:T}$ is constructed with $\boldsymbol{x}_t$, in POMCP++, the rollout policy only attains high reward-to-go when all samples are close to $\boldsymbol{x}_t$, which requires the belief at leaf nodes to have low uncertainty.

VI. Experiments

In the experiment section, we compare the performance of POMCP++ with RHC, DESPOT [18], and POMCPOW [38] through simulations. The idea of certainty equivalence control and separation principle is often extended and applied in the RHC framework to solve the stochastic motion planning problem in practice. In the following, methods of this kind is called RHC for short. Particularly, to solve the proposed problem, one state sample is drawn from the belief at each step, which is regarded as the true state. A^*, with inflated heuristics [49], is then used to find the next action assuming a deterministic motion model. DESPOT [12] is the state-of-the-art online POMDP solver designed for discrete systems. The simulation results of DESPOT serve as the benchmark for naively applying discrete POMDP solvers to continuous systems. POMCPOW [38], also an extension of POMCP, is the most similar to our work. Comparison with POMCPOW shows the importance of the improvements in POMCP++.

TABLE I
Hyperparameters for Different Methods

DESPOT	tree depth $= \{10, \mathbf{50}, 100\}$
	scenario # $= \{30, 100, \mathbf{500}, 1000\}$
	per-step planning time $= \{\mathbf{30}\}$
POMCPOW	$\epsilon_a = \{0.05, 0.1, 0.2, \mathbf{0.3}\}$
	$\alpha = \{0.1, 0.5, \mathbf{1}, 2\}$
	$\kappa = \{\mathbf{0.1}, 0.5, 1, 5\}$
	$N = \{1000, \mathbf{3000}, 5000\}$
POMCP++	$\epsilon_a = \{0.05, \mathbf{0.1}, 0.2, 0.3\}$
	$\epsilon_z = \{\mathbf{-1}, -3, -5, -7\}$
	$K = \{16, 32, \mathbf{64}\}$
	$N = \{\mathbf{3000}\}$

The bold entities refer to the selected hyperparameters for different methods.

There are minor modifications to POMCPOW and DESPOT to adapt to the proposed problem. POMCPOW is designed to handle continuous action space in addition to continuous state and observation spaces. The action space in the problem setup of this work is discrete and finite, consisting of motion primitives. Therefore, we apply the same ϵ-greedy action selection strategy (Algorithm 2, Line 29) in the POMCPOW framework. For both DESPOT and POMCPOW, the same rollout policy as in Algorithm 2 is used.

Section VI-A evaluates different methods in relatively simple simulated environments, where the optimal policy can be anticipated. The performances of the methods are compared in terms of the summation of discounted reward. The discounted reward is the objective to be maximized, and is widely used as a metric to quantify the performance of general POMDP solvers in the AI literature. In contrast, Section VI-B evaluates the methods in a more realistic hallway environment. The performance of the methods are compared in terms of metrics that are of more interest to robotics researchers, such as the rate of collisions, total number of actions, the entropy of the final distribution, etc. In addition, we compare the performance of POMCPOW and POMCP++ under different per-step planning time constraints. Finally, we demonstrate the usage of POMCP++ by transitioning the simulation in the hallway environment to hardware experiments.

A. Tests in Simulated Environments

In the simulated environments, the cell size of the used occupancy grid maps is set to 0.1 m. The action space consists of six different motion primitives, $\mathcal{A}_m = \{0, 0.2\}$ m/s $\times$ $\{-\frac{\pi}{2}\, 0, \frac{\pi}{2}\}$ rad/s $\times$ $\{0.5\}$ s. The robot models a differential drive ground vehicle with forward motion only. The range sensor installed on the robot has seven beams covering 2π uniformly. Each beam has a maximum range of $z_{\max} = 0.3$ m. ϵ_g in (4),

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

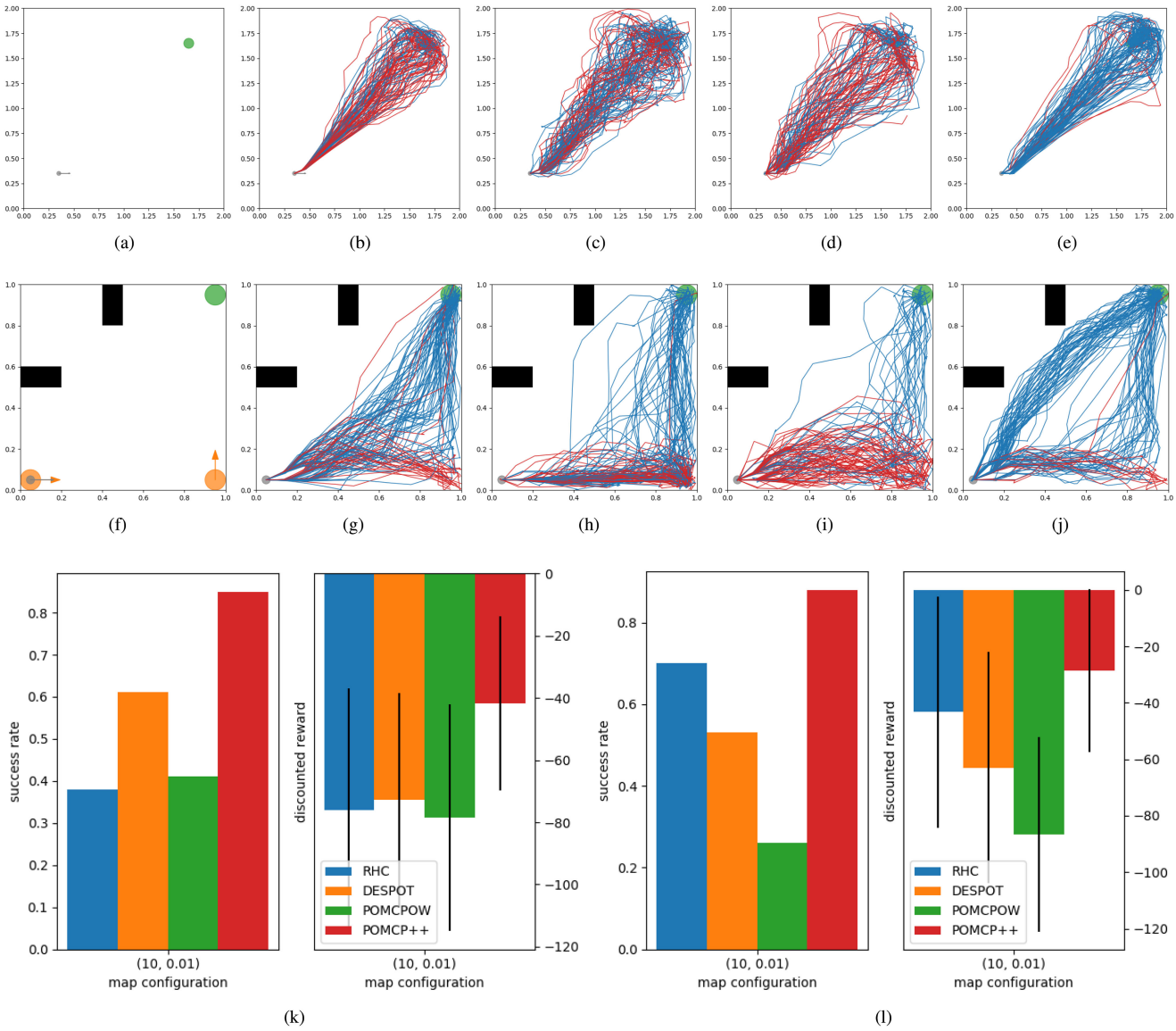


Fig. 3. (a) and (f) Initial configurations for the experiments in Section VI-A1. Occupied cells in the maps are marked with black squares. The green circular region is the target location. The gray circle is the initial state of the robot. The orange circles with the arrows in (f) are the two modes in the initial belief. (b)–(e) are the paths executed with RHC, DESPOT, POMCPOW, and POMCP++ in setup (a) over the 100 experiments. Blue lines are the paths for the successful trials, whereas red lines are the failed ones. The paths executed by the methods in setup (f) are shown in (g)–(j). (k) and (l) are the success rate and average discounted reward with standard deviation obtained through the 100 experiments for setup (a) and (f), respectively.

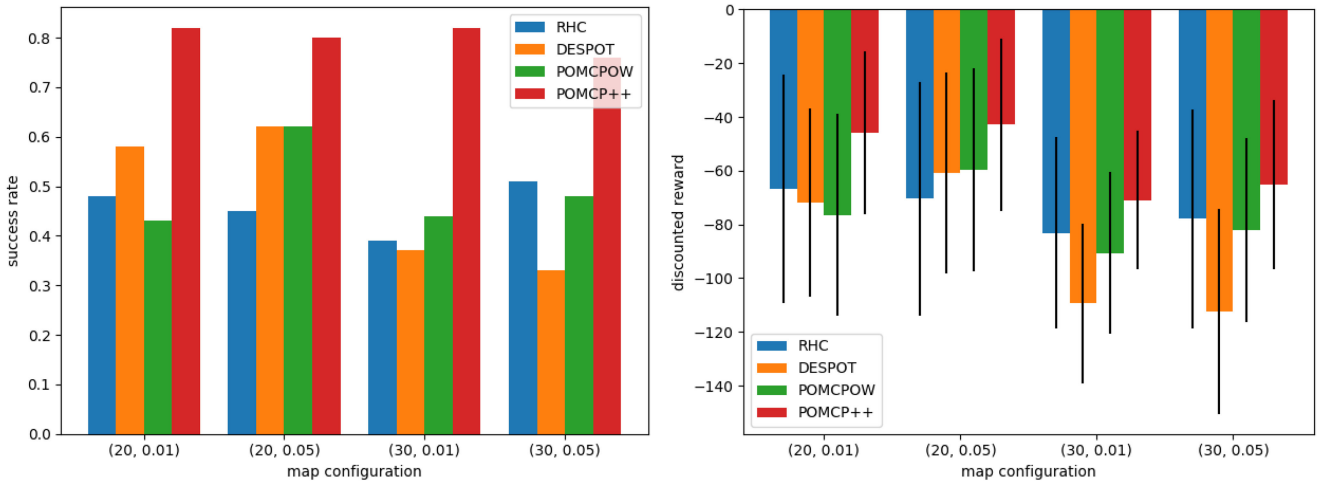
the radius of the goal region, is 0.05 m.[3] γ, the discount factor, is set to 0.99.

Table I lists the range and the selected values for parameters of each method. See Somani *et al.* [12] and Sunberg and Kochenderfer [38] for the functionality of the parameters. It should be specially noted that, for DESPOT, *tree depth* is set to be well beyond the maximum number of steps to reach the goal. *per-step planning time* is also chosen to be large enough such that time limitation is seldom used as a termination criterion for each planning step.

[3]The setup is merely for the ease to convert quantities to metric length units. All related quantities can be scaled proportionally.

1) Fixed Maps: In this section, we show both qualitative and quantitative results of applying the methods on two fixed representative maps. In the first setup [see Fig. 3(a)], we use a 20×20 map with no obstacles. The initial state is known to the robot. In order to reach the target location with high confidence, it is expected that the robot will take advantage of the map boundaries for localization. The second setup [see Fig. 3(f)] is a 10×10 map, designed to be more challenging in that the initial distribution is ambiguous, consisting of two modes located at the bottom corners. The obstacles in this setup are also set in a way such that the robot is not able to remove the state ambiguity passively, but has to actively approach the obstacles. We run each method 100 times on the two setups to obtain the average performance.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 4. For each method and map configuration, the performance of the method is measured with 100 experiments on randomly generated maps. (a) Success rate of different methods. Recall (4), a trial is successful when the robot actively chooses to stop around the target location. (b) Average summation of discounted reward with standard deviation. For failure trials, it is assumed that the robot will take feasible actions (with reward -1 per step) onward but never reach the target location.

Fig. 3(k) and (l) shows the success rate and the average discounted reward of different methods. More insightful observations can be obtained by plotting the paths (see Fig. 3). In the first setup, the proposed POMCP++ is able to reliably utilize the right upper corner to reduce the uncertainty accumulated during the movement, and then come back to the target location. This capability explains the high success rate of POMCP++. For the second setup, an immediate observation is that all methods try to go along the bottom and right side of the map. This is due to the best-first-search nature of the algorithms. Going along the bottom edge is the fastest way to move one of the belief modes to the target location. Occasionally, the planning algorithms may wrongly choose to stop at this moment. More often, exploring the policies helps the algorithms realize that moving along the right side can further reduce the uncertainty and eventually leave a single belief mode stopping at the target. A more interesting observation is that the proposed POMCP++ is able to find another cluster of paths by moving the robot diagonally across the map. The paths in this cluster are able to remove the ambiguity in the belief within the first few steps by utilizing the obstacles on the left. Although such paths deviate from the "promising" path (going right, then up) of the short horizon, they obtain higher overall reward.

2) Random Maps: In order to evaluate the average performance of different methods, they are also compared using random maps with different configurations. Four map configurations are included by permuting different map size, $\{20 \times 20, 30 \times 30\}$, and obstacle density, $\{0.01, 0.05\}$. One hundred experiments are performed for each method and map configuration. In each experiment, a random map is generated as per the map configuration. For all experiments, the robot starts with a known initial state.

Fig. 4 shows the success rate and summation of discounted reward for different methods. POMCP++ is able to achieve much higher success rate and reward, constantly outperforming the other three methods. It should be also noted that RHC performs reasonably well in this experiment, better than the POMDP-based methods, DESPOT, and POMCPOW, on some map configurations.

B. Tests in Hallway Environments

In this section, we compare the performance of POMCP++ with RHC and POMCPOW in an application of navigating a differential drive ground robot in a hallway environment. The dimension of the map is 153×277, with 0.1-m resolution. To adapt to the enlarged map and narrow corridors, the motion primitives are changed to $\mathcal{A}_m = \{0, 0.5\}$ m/s $\times \{-\frac{\pi}{6}\, 0, \frac{\pi}{6}\}$ rad/s $\times \{1.0\}$ s. The radius of the goal region ϵ_g is enlarged to 0.5 m. Seven LIDAR beams are used as sensor measurements covering $\frac{3}{2}\pi$, with $z_{\max} = 1.5$ m. As in Section VI-A, γ is 0.99. The algorithms are compared in two scenarios with different initial belief shown in Fig. 1. The hyperparameters for POMCPOW and POMCP++ remain the same as in Table I.

The comparison with DESPOT is omitted, since the robot fails to reach the goal for all trials with DESPOT. Considering the enlarged environment compared to Section VI-A, experiments for DESPOT are also performed by increasing *tree depth* to 100 and *per-step planning time* to 120 s. There is still no successful trails for either of the initial belief setups. The failure of DESPOT may be caused by using a single simulation episode to estimate the value at a belief node. To elaborate, a naive application of DESPOT to systems with continuous measurement causes the degeneracy of the belief tree. All nodes in the tree, other than the root, are only visited once, i.e., when the node is constructed. Therefore, only one simulation episode is available in estimating the value at the nodes, which may be far from adequate in providing an accurate estimate. The inaccurate estimation at the nodes, eventually, leads to the wrong selection of optimal policy. In the experiments of Section VI-A, simulation episodes

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

TABLE II
COMPARISON OF RHC, POMCPOW, AND POMCP++ IN A HALLWAY ENVIRONMENT

initial belief	method	success/collision/wrong stop/timeout[1][%]	total actions[2]	total distance[2][m]	distance to target[2][m]	final entropy[2]
single mode	RHC	0.66/0.32/0.02/**0.00**	**66.48**	**29.13**	0.45	2.92
	POMCPOW	0.78/0.08/0.10/0.04	74.36	29.64	0.41	2.56
	POMCP++	**0.84**/**0.00**/**0.00**/0.16	75.69	29.51	**0.31**	**2.53**
two modes	RHC	0.58/0.38/0.04/**0.00**	**66.62**	**29.07**	0.44	2.94
	POMCPOW	0.34/0.36/0.30/**0.00**	74.18	29.66	0.39	**2.40**
	POMCP++	**0.90**/**0.00**/**0.02**/0.08	73.38	29.28	**0.26**	2.46

[1]In Table II and III, the bold entities refer to the best performance of all methods in comparison.
[2]There are three causes of a failure trial, collision, wrong stop (stops more than 0.5 m away from the target), or timeout (uses over 100 steps).
[3]These data are averaged over successful trials only.

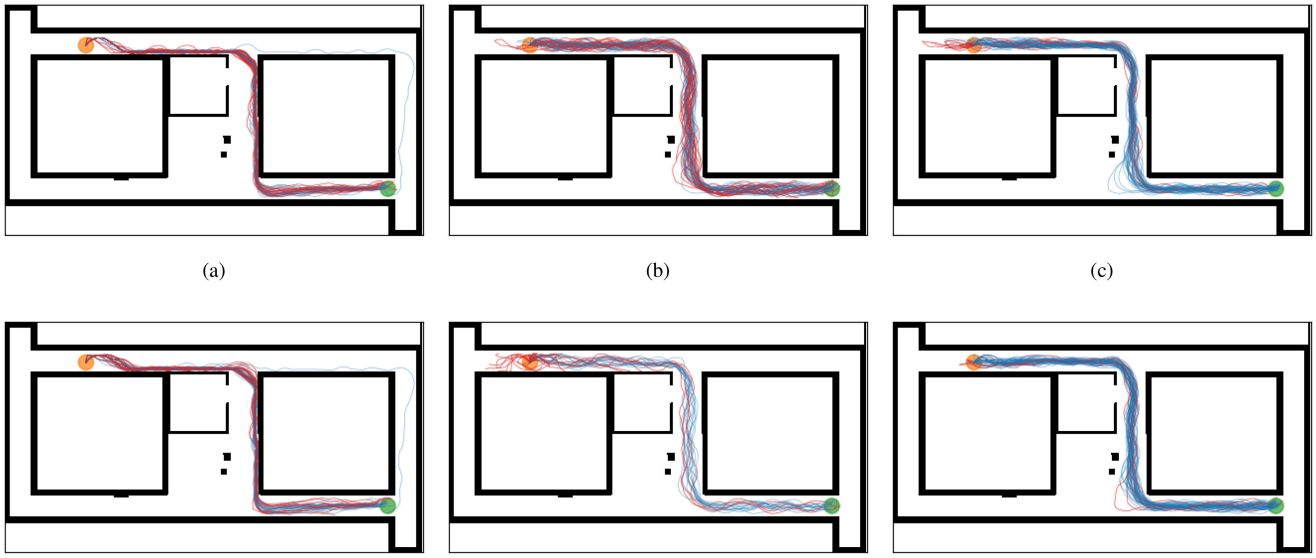


Fig. 5. (a)–(c) Paths of RHC, POMCPOW, and POMCP++ with single mode initial belief, whereas (d)–(f) are for the two mode initial belief. The orange and green patches are the start and target locations. Blue and red lines are paths of successful and failure trials. More details of the setup for the two different scenarios can be found in Fig. 1.

are short. It takes around ten steps to drive the robot to the goal. Simulation episodes obtained with the same action sequence may not differ much. Therefore, using only one simulation episode in estimating the value at the nodes may be sufficient and leads to meaningful actions. However, the number of actions required to reach the goal in the hallway experiments is over 60. The effect of degeneracy for DESPOT becomes more apparent.

Table II summarizes the performance of RHC, POMCPOW, and POMCP++. For each algorithm and initial belief setup, we run the experiment 50 times. The paths of all trials are shown in Fig. 5. It is expected that RHC takes fewer actions on average for successful trials, whose planning is based on A*. However, the total traveled distance is similar among the algorithms. This implies that, under POMDP solutions, the robot takes more rotation actions utilizing the local environment to reduce state uncertainty. Fig. 5 also shows that POMDP solutions are able to maintain the robot in the middle of a narrow corridor, which reduces the chance of collision caused by state uncertainty and motion stochasticity. Comparing POMCPOW

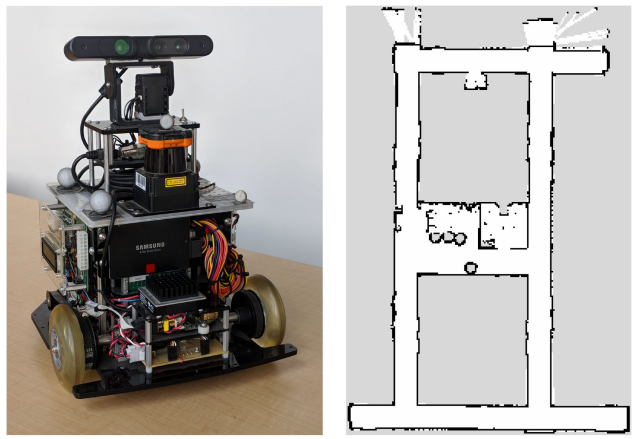


Fig. 6. (a) Differential drive ground robot used in the hardware experiment. Seven beams, spanning $\frac{3}{2}\pi$ with $z_{\max} = 1.5$ m, are used as sensor measurements. (b) Map of the hallway environment, built with gmapping[50]. The dimension of the occupancy grid map, 280×186 with 0.1-m resolution, is slightly different from the one used in the simulation.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

TABLE III
COMPARISON OF POMCPOW AND POMCP++ WITH DIFFERENT PER-STEP PLANNING TIME CONSTRAINTS

per-step time $[s]$	method	success/collision/wrong stop/timeout[1][%]	total actions[1]	total distance[1]$[m]$	distance to target[1]$[m]$	final entropy[1]
1	POMCPOW	0.18/**0.18**/0.62/**0.02**	**80.33**	**30.00**	0.40	2.52
	POMCP++	**0.22**/0.20/**0.52**/0.06	88.82	30.95	**0.28**	**2.39**
5	POMCPOW	0.64/0.14/0.20/**0.02**	**75.56**	30.10	0.38	**2.46**
	POMCP++	**0.74**/**0.00**/**0.04**/0.22	78.60	**29.70**	**0.28**	2.48
10	POMCPOW	0.68/0.14/0.10/**0.08**	75.24	29.91	0.39	**2.54**
	POMCP++	**0.76**/**0.00**/**0.04**/0.20	**74.39**	**29.31**	**0.22**	2.60
15	POMCPOW	0.68/0.12/0.14/**0.06**	74.35	29.70	0.37	**2.54**
	POMCP++	**0.90**/**0.02**/**0.00**/0.08	**73.58**	**29.41**	**0.28**	2.67

[1] See Table II for the notes on the metrics.

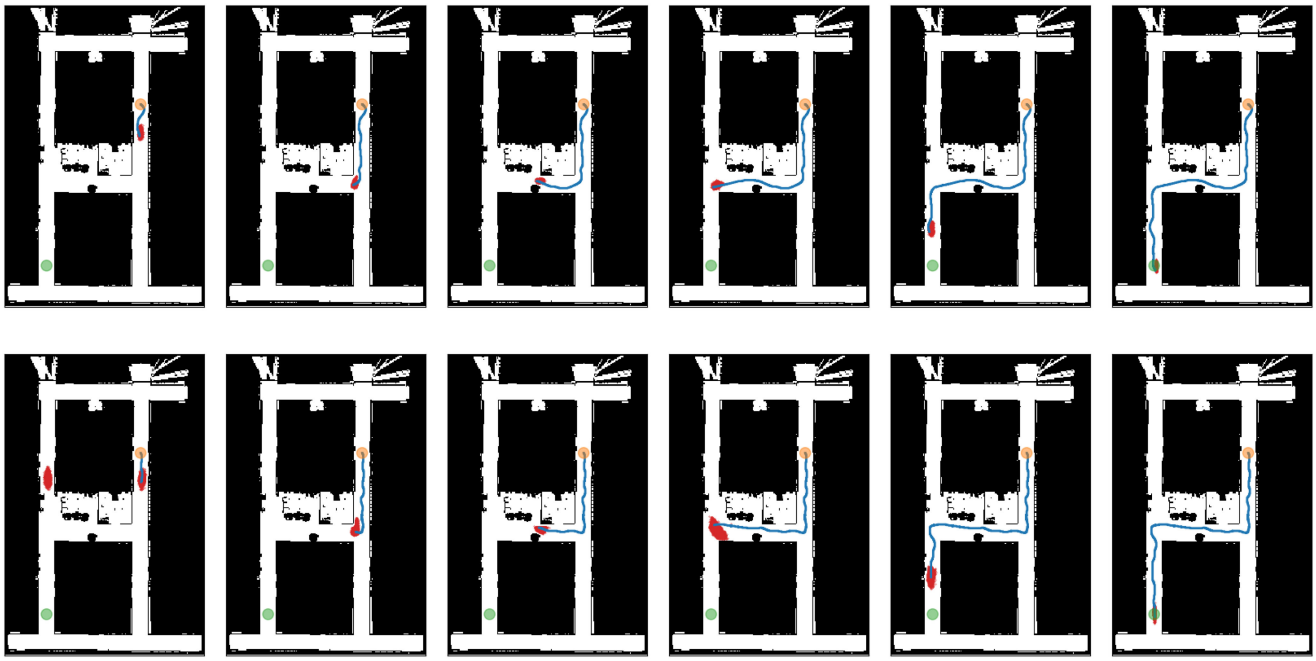


Fig. 7. Change of belief (red particles) of the robot position in the hardware experiments using POMCP++. The first row is the belief snapshots for the single mode initial belief setup, whereas the second row is the one for the two mode initial belief setup. The orange and green patches are the start and target locations. Blue lines are paths estimated with an onboard LIDAR odometry.

and POMCP++, although the performance of the success trails of the two algorithms is similar, POMCP++ is able to achieve higher success rate, especially when the initial belief contains ambiguity. The improvement of POMCP++ is due to the two major modifications of POMCP, namely the new strategy of measurement selection, and the removal of the implicit perfect state assumption at the start of the rollout phase.

We also compare the anytime performance of POMCPOW and POMCP++ in the hallway environment with single mode initial belief setup. Table III summarizes the performance of POMCPOW and POMCP++ under different per-step time constraints. The simulations are conducted on a desktop computer using Intel Core i9-9920X CPU with 12 cores running at 3.5 GHz. The maximum per-step planning time is set to 15 s, where both POMCPOW and POMCP++ are able to finish 3000 episodes (the default N in Table I). It can be observed in Table III that POMCP++ is able to constantly outperform POMCPOW in terms of success rate. Table III also indicates that albeit the anytime feature, sufficient per-step planning time is required for both methods in order to achieve reasonable performance.

C. Hardware Demonstration

To further demonstrate the application of POMCP++, the same experiments in the simulations are carried out with hardware. Fig. 6 shows the setup of the ground robot and the map of the hallway environment similar to the one used in the simulations. Fig. 7 shows paths and the change of belief with the two different initial belief configurations. It should be noted that the hardware experiment serves as a demonstration that POMCP++ can be applied to actual hardware robots and realistic environments in addition to ideal models in the simulations. In terms of efficiency, POMCP++ may not be sufficient for real-time

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

planning purposes, especially with onboard computers. Most of the planning time for each step is spent on constructing rollout policies and forward simulating the system. Using the onboard processor i7-6600U, 2 cores running at 2.6 GHz, the planning time for an initial step can take as long as 2 min, when the robot is far from the goal. Improving the efficiency of the algorithm is a promising direction of future research.

VII. Conclusion

In this article, we formulated a stochastic goal navigation problem as a POMDP, and proposed a new algorithm, POMCP++, an MCTS method based on POMCP for partially observed problems. The algorithm featured two key improvements over prior methods, including a novel measurement selection strategy and the usage of importance sampling to avoid overoptimism in the value estimation of the rollout policy. Compared with the LQG approaches, the proposed algorithm inherited the advantages from general POMDP solvers, such as dealing with multimodel belief and requiring only generative models for the system. Meanwhile, with the introduced improvements, POMCP++ can be applied on systems with continuous measurements, which was often a limitation of general POMDP solvers. More importantly, the proposed POMCP++ was proved to provide unbiased estimation of the policy value with infinite number of samples and simulation episodes, thus was a valid MCTS algorithm.

We demonstrated the effectiveness of the algorithm with comparisons to both existing general purpose online POMDP solvers, and a practical robotic approach based on RHC. The results indicated that POMCP++ outperforms other methods on various metrics. In terms of the metric generally used to quantify POMDP solvers, POMCP++ achieved higher total reward. In terms of reliability in completing the navigation task, POMCP++ achieved higher success rate.

Various future research directions are promising in extending the current work. Problemwise, it should be of broader interest to release the assumption of a known map and investigate the problem of navigation in unknown or uncertain environments. Solutionwise, improving the efficiency of POMCP++ is critical for real-time applications on computationally constrained platforms.

Appendix A

We attempt to answer two questions in this section. First, how often a node in the tree is visited? Second, how many child nodes a belief action node will spawn using the measurement selection strategy in Algorithm 2?

Lemma 1: A belief action node $h\boldsymbol{a}$ will be visited *i.o.* given that its parent belief node h is visited *i.o.* and $0 < \epsilon_a < 1$.

Proof: Define A_n as the event that $h\boldsymbol{a}$ is visited at the nth visit of h. A_ns are independent events following the definition of the action selection function in Algorithm 2. Note $P(A_n)$ is lower bounded by $1/\epsilon_a$ with $0 < \epsilon_a < 1$. Therefore

$$\sum_{n=1}^{\infty} P(A_n) \geq \lim_{n\to\infty} \frac{n}{\epsilon_a} = \infty.$$

Define A as the event that infinitely many A_ns occur, i.e., $A = \limsup_{n\to\infty} A_n$. We have $P(A) = 1$ as a result of Borel–Cantelli lemmas [51]. ■

Lemma 2: A belief node $h\boldsymbol{z}$ will be visited *i.o.* given that its parent belief action node h is visited *i.o.* and $\epsilon_z < 0$.

Proof: Assume $h\boldsymbol{z}$ is the mth spawned child node of h. Define Z_n as the independent events that $h\boldsymbol{a}$ is visited at the nth visit of h after $h\boldsymbol{z}$ is spawned. $P(Z_n)$ with $n = 1, 2, \ldots$ is a nonincreasing sequence since the number of child nodes of h is nondecreasing. Note $P(Z_n)$ is lower bounded by $(1-(m+n)^{\epsilon_z})/(m+n-1)$, which assumes new child node is added at every visit of h. Therefore

$$\begin{aligned}\sum_{n=1}^{\infty} P(Z_n) &\geq \sum_{n=1}^{\infty} \frac{1-(m+n)^{\epsilon_z}}{m+n-1} \\ &= \sum_{n=1}^{\infty} \frac{1}{m+n-1} - \sum_{n=1}^{\infty} \frac{1}{(m+n-1)(m+n)^{-\epsilon_z}} \\ &= \sum_{k=m}^{\infty} \frac{1}{k} - \sum_{k=m}^{\infty} \frac{1}{k(k+1)^{-\epsilon_z}} \geq \sum_{k=m}^{\infty} \frac{1}{k} - \sum_{k=m}^{\infty} \frac{1}{k^{1-\epsilon_z}}.\end{aligned}$$

Given $\epsilon_z < 0$, $\sum_{n=1}^{\infty} P(Z_n) = \infty$ following the property of p-series. We have $P(\limsup_{n\to\infty} Z_n) = 1$ as a result of Borel–Cantelli lemmas. ■

Theorem 1: If the root node is visited *i.o.*, *i.e.,* $N \to \infty$, $0 < \epsilon_a < 1$, and $\epsilon_z < 0$, all nodes will be visited *i.o.* ■

Proof: Based on Lemmas 1 and 2, the *i.o.* visit of a node in the tree depends on the *i.o.* visit of its parent nodes. Therefore, all nodes are visited *i.o.* if the root node is visited *i.o.* ■

Theorem 2: A belief action node h spawns child nodes *i.o.* if $-1 \leq \epsilon_z < 0$.

Proof: Define S_n as the independent events that a new child belief node is spawned by h at its nth visit. The sequence $P(S_n)$, $n = 1, 2, \ldots$ is nonincreasing, with each $P(S_n)$ lower bounded by $(n+1)^{\epsilon_z}$. Therefore

$$\sum_{n=1}^{\infty} P(S_n) \geq \sum_{n=1}^{\infty} (n+1)^{\epsilon_z}.$$

A sufficient condition for $\sum_{n=1}^{\infty} P(S_n) = \infty$ is $-1 \leq \epsilon_z < 0$ ($\epsilon_z < 0$ is enforced to ensure $0 < (n+1)^{\epsilon_z} < 1$), under which $P(\limsup_{n\to\infty} S_n) = 1$, i.e., h spawns new child nodes *i.o.* ■

Intuitively speaking, ϵ_z controls the tradeoff between exploration and exploitation in measurements just as the role of ϵ_a in action selection. Fig. 8 shows distribution of $|C(h\boldsymbol{a})|$ and the discounted reward with different ϵ_z. Although the results of Theorem 2 may not apply, cases with $\epsilon_z < -1$ are also included for comparison.

Appendix B

In this section, we would like to show that POMCP++ is a valid MCTS algorithm, which is, at its base, an asynchronous value iteration algorithm interleaving policy evaluation and policy improvement. Since POMCP++ already share the algorithm structure with MCTS, the validity of POMCP++ depends on

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

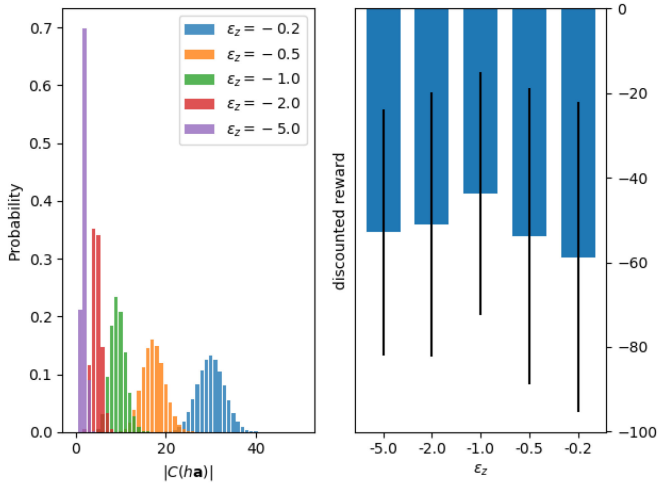


Fig. 8. Left shows the distribution of $|C(h\boldsymbol{a})|$ under different ϵ_z, with $N(h\boldsymbol{a}) = 50$. Right shows average discounted reward for different ϵ_z on random maps of size 20×20 and obstacle density 0.1. For each ϵ_z, the data are averaged over 100 independent trials.

whether the value computed from the simulation episodes serves as unbiased estimates of value at the corresponding belief-action nodes. It seems straightforward, at first glance, that estimation from simulation episodes is unbiased. However, one should be careful in drawing the conclusion since the measurements in simulation episodes are not always obtained by forward simulating the measurement model, but are enforced to be repeated most of the time. To fill in this gap, we start by considering the value at the belief nodes in the tree, which are not being tracked explicitly in Algorithm 2. In the following, we assume $0 < \epsilon_a < 1$ and $-1 \leq \epsilon_z < 0$ to ensure all nodes are visited *i.o.*, and new child nodes are spawned *i.o.* by belief action nodes as proved in Theorems 1 and 2.

Lemma 3: The discounted summation of rewards of the simulation episode is an unbiased estimate of the value at the traversed belief nodes as $N \to \infty$ and $K \to \infty$.

Proof: Represent a simulation episode from belief node $h_t \in \mathcal{T}$ as $S_t := \boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{z}_t, \boldsymbol{x}_{t+1}, \boldsymbol{a}_{t+1}, \boldsymbol{z}_{t+1}, \ldots$, with $\boldsymbol{x}_t$ sampled from $\boldsymbol{b}_t$. For the sake of notation sanity, define $X_t := \boldsymbol{x}_{t:\infty}$, $A_t := \boldsymbol{a}_{t:\infty}$, $Z_t := \boldsymbol{z}_{t:\infty}$, i.e., $S_t = X_t, A_t, Z_t$. Then, the value of the node h_t under the policy π is

$$V_\pi(h_t) = \int_{S_t} R(S_t) p_\pi(S_t) dS_t.$$

Recall $R(S_t)$ is the discounted summation of reward of a simulation episode, $p_\pi(S_t)$ is the pdf of S_t. Here and also in the following context, the subscript π in $p_\pi(\cdot)$ indicates the pdf is related to the policy π. With some application of conditional probability and Bayes' rule, one has

$$\begin{aligned} V_\pi(h_t) &= \int_{S_t} R(S_t) p_\pi(S_t) dS_t \\ &= \int_{S_t} R(S_t) p(X_t|A_t, Z_t) p_\pi(A_t, Z_t) dS_t \\ &= \int_{S_t} p_\pi(A_t, Z_t) R(S_t) \frac{p(Z_t|X_t, A_t) p(X_t|A_t)}{p(Z_t|A_t)} dS_t \\ &= \int_{A_t, Z_t} p_\pi(A_t, Z_t) V(h_t|A_t, Z_t) dA_t dZ_t \end{aligned} \tag{6}$$

where

$$V(h_t|A_t, Z_t) = \int_{X_t} R(S_t) \frac{p(Z_t|X_t, A_t) p(X_t|A_t)}{p(Z_t|A_t)} dX_t.$$

$V(h_t|A_t, Z_t)$ is the value of the node h_t given the future measurement and action sequence. In Algorithm 2, $V(h_t|A_t, Z_t)$ is estimated with

$$\tilde{V}(h_t|A_t, Z_t) = \frac{1}{\eta} \sum_{i=0}^{K-1} R(S_t^i) p(Z_t|X_t^i, A_t) \tag{7}$$

where X_t^i is forward sampled following the given action sequence A_t with the initial state $\boldsymbol{x}_t^i$ sampled from $\boldsymbol{b}_t$. In [52], Geweke proves the validity of estimation with importance sampling under weak assumptions,[4] i.e.,

$$\tilde{V}(h_t|A_t, Z_t) \xrightarrow[\text{as } K\to\infty]{a.s.} V(h_t|A_t, Z_t) \tag{8}$$

where $\xrightarrow{a.s.}$ means *almost surely* convergence. Therefore

$$\int_{A_t, Z_t} p_\pi(A_t, Z_t) \tilde{V}(h_t|A_t, Z_t) dA_t dZ_t \xrightarrow[\text{as } K\to\infty]{a.s.} V_\pi(h_t). \tag{9}$$

If one samples a measurement and action sequence from $p_\pi(A_t, Z_t)$, (9) shows $\tilde{V}(h_t|A_t, Z_t)$ is an unbiased estimate of $V_\pi(h_t)$. However, in following the tree policy, (A_t, Z_t) is sampled from the distribution $p_\mathcal{T}(A_t, Z_t)$ defined implicitly by the tree $\mathcal{T}$, instead of $p_\pi(A_t, Z_t)$. In the following, we show that if new measurements are sampled *i.o.* from each belief-action node in $\mathcal{T}$, $\tilde{V}(h_t|A_t, Z_t)$ still serves as an unbiased estimate of $V_\pi(h_t)$ even though (A_t, Z_t) is sampled based on $p_\mathcal{T}(A_t, Z_t)$.

To start, consider the first action measurement pair $(\boldsymbol{a}_t, \boldsymbol{z}_t)$ in (A_t, Z_t). Define $f(\cdot)$ to be an integrable function *w.r.t.* the corresponding probability measure (a bounded function defined on the sample space should suffice). Then

$$\begin{aligned} &\mathop{\mathbb{E}}_{p_\mathcal{T}(\boldsymbol{a}_t, \boldsymbol{z}_t)} \{f(\boldsymbol{a}_t, \boldsymbol{z}_t)\} = \int_{\boldsymbol{a}_t} \left(\frac{1}{n} \sum_{i=0}^{n-1} f(\boldsymbol{a}_t, \boldsymbol{z}_t^i) \right) p_\pi(\boldsymbol{a}_t) d\boldsymbol{a}_t \\ &\xrightarrow[\text{as } n\to\infty]{a.s.} \int_{\boldsymbol{a}_t} \left(\int_{\boldsymbol{z}_t} f(\boldsymbol{a}_t, \boldsymbol{z}_t) p_\pi(\boldsymbol{z}_t|\boldsymbol{a}_t) d\boldsymbol{z}_t \right) p_\pi(\boldsymbol{a}_t) d\boldsymbol{a}_t \\ &= \int_{\boldsymbol{a}_t, \boldsymbol{z}_t} f(\boldsymbol{a}_t, \boldsymbol{z}_t) p_\pi(\boldsymbol{a}_t, \boldsymbol{z}_t) d\boldsymbol{a}_t d\boldsymbol{z}_t. \end{aligned} \tag{10}$$

The first equality in (10) is based on how actions and measurements are selected at node $h_t \in \mathcal{T}$ in Algorithm 2. Here and in the following context, we assume that n measurements are sampled for each belief-action node in the tree with n sufficiently large. With large n, we can safely ignore the probability of sampling new measurements at the belief-action nodes, i.e., one of the n measurements will be selected. The *a.s.* convergence in (10) follows the strong law of large numbers.

[4]Briefly speaking, the four assumptions in [52] require the proper definition of prior density and the likelihood function, *independent identically distributed* sample sequence, and an integrable *r.v. w.r.t.* posterior density. All of the assumptions are satisfied in our problem setup.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

Mathematical induction lends itself to prove the rest of the action, measurement sequence. Suppose

$$\begin{aligned}&\mathop{\mathbb{E}}_{p_{\mathcal{T}}(\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})}\{f(\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})\}\\ &\xrightarrow{a.s.}\mathop{\mathbb{E}}_{p_{\pi}(\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})}\{f(\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})\}.\end{aligned}\tag{11}$$

Note

$$\begin{aligned}&\mathop{\mathbb{E}}_{p_{\mathcal{T}}(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})}\{f(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})\}\\ &=\mathop{\mathbb{E}}_{p_{\mathcal{T}}(\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})}\left\{\mathop{\mathbb{E}}_{p_{\mathcal{T}}(\boldsymbol{a}_T,\boldsymbol{z}_T)}\{f(\boldsymbol{a}_T,\boldsymbol{z}_T,\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})\}\right\}\\ &\xrightarrow{a.s.}\mathop{\mathbb{E}}_{p_{\pi}(\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})}\left\{\mathop{\mathbb{E}}_{p_{\mathcal{T}}(\boldsymbol{a}_T,\boldsymbol{z}_T)}\{f(\boldsymbol{a}_T,\boldsymbol{z}_T,\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})\}\right\}.\end{aligned}\tag{12}$$

The *a.s.* convergence in (12) follows the assumption in (11). It is worth mentioning that both $\mathbb{E}_{p_{\mathcal{T}}(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})}\{f(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})\}$ and $\mathbb{E}_{p_{\pi}(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})}\{f(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})\}$ are also bounded functions on $(\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})$. Therefore, in order to show

$$\begin{aligned}&\mathop{\mathbb{E}}_{p_{\mathcal{T}}(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})}\{f(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})\}\\ &\xrightarrow{a.s.}\mathop{\mathbb{E}}_{p_{\pi}(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})}\{f(\boldsymbol{a}_{t:T},\boldsymbol{z}_{t:T})\}.\end{aligned}\tag{13}$$

Equation (12) suggests it is suffice to have

$$\begin{aligned}&\mathop{\mathbb{E}}_{p_{\mathcal{T}}(\boldsymbol{a}_T,\boldsymbol{z}_T)}\{f(\boldsymbol{a}_T,\boldsymbol{z}_T,\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})\}\\ &\xrightarrow{a.s.}\mathop{\mathbb{E}}_{p_{\pi}(\boldsymbol{a}_T,\boldsymbol{z}_T)}\{f(\boldsymbol{a}_T,\boldsymbol{z}_T,\boldsymbol{a}_{t:T-1},\boldsymbol{z}_{t:T-1})\}.\end{aligned}\tag{14}$$

The same argument as (10) can be applied again to show the validity of (14). Recall that the belief action nodes in $\mathcal{T}$ are visited *i.o.* and spawn new measurements *i.o.* as $N\to\infty$, proved in Theorem 2. Therefore, $\tilde{V}(h_t|A_t,Z_t)$, a bounded function, is an unbiased estimate of $V_\pi(h_t)$ with (A_t,Z_t) sampled from $p_{\mathcal{T}}(A_t,Z_t)$, i.e.,

$$\begin{aligned}&\mathop{\mathbb{E}}_{p_{\mathcal{T}}(A_t,Z_t)}\left\{\tilde{V}(h_t|A_t,Z_t)\right\}\\ &\xrightarrow[\text{as } N\to\infty]{a.s.}\mathop{\mathbb{E}}_{p_{\pi}(A_t,Z_t)}\left\{\tilde{V}(h_t|A_t,Z_t)\right\},\qquad\text{Eq. (13)}\\ &\xrightarrow[\text{as } K\to\infty]{a.s.}\mathop{\mathbb{E}}_{p_{\pi}(A_t,Z_t)}\{V(h_t|A_t,Z_t)\},\qquad\text{Eq. (8)}\\ &=V_\pi(h_t),\qquad\text{Eq. (6)}\end{aligned}$$

which completes the proof. ■

Next, we extend the conclusion to the belief action nodes in $\mathcal{T}$ as well.

Theorem 3: The discounted summation of rewards of the simulation episode is an unbiased estimate of the value at all traversed nodes as $N\to\infty$ and $K\to\infty$.

Proof: Given the results of Lemma 3, it is left to show that the summation of discounted rewards is also an unbiased estimation at the traversed belief action nodes. Recall (5)

$$\tilde{V}(h_t\boldsymbol{a}_t|A_{t+1},Z_t)=\frac{1}{\eta}\sum_{i=0}^{K-1}p\big(Z_t|X_t^i,A_t\big)R(S_t).\tag{15}$$

Applying the theory of importance sampling again

$$\begin{aligned}&\tilde{V}(h_t\boldsymbol{a}_t|A_{t+1},Z_t)\xrightarrow{a.s.}\int_{X_t}p(X_t|A_t,Z_t)R(S_t)dX_t\\ &=\int_{X_t}p(X_t|A_t,Z_t)(r(\boldsymbol{x}_t,\boldsymbol{a}_t,\boldsymbol{x}_{t+1})+\gamma R(S_{t+1}))dX_t\end{aligned}$$

shows the summation of stage reward and reward-to-go. The stage reward can be simplified to

$$\begin{aligned}&\int_{X_t}p(X_t|A_t,Z_t)r(\boldsymbol{x}_t,\boldsymbol{a}_t,\boldsymbol{x}_{t+1})dX_t\\ &=\int_{\boldsymbol{x}_t,\boldsymbol{x}_{t+1}}r(\boldsymbol{x}_t,\boldsymbol{a}_t,\boldsymbol{x}_{t+1})p(\boldsymbol{x}_t,\boldsymbol{x}_{t+1}|A_t,Z_t)d\boldsymbol{x}_t d\boldsymbol{x}_{t+1}\\ &=r(\boldsymbol{b}_t,\boldsymbol{a}_t,\boldsymbol{b}_{t+1}).\end{aligned}$$

With some abuse of notation, $r(\cdot)$ denotes both the stage reward for $\boldsymbol{x}_t,\boldsymbol{x}_{t+1}$, and its expectation *w.r.t.* $p(\boldsymbol{x}_t,\boldsymbol{x}_{t+1}|A_t,Z_t)$. Meanwhile, the reward-to-go is

$$\int_{X_t}p(X_t|A_t,Z_t)\cdot\gamma R(S_{t+1})dX_t=\gamma\tilde{V}(h_t\boldsymbol{a}_t\boldsymbol{z}_t|A_{t+1},Z_{t+1}).$$

Therefore, (15) can be rewritten as

$$\tilde{V}(h_t\boldsymbol{a}_t|A_{t+1},Z_t)=r(\boldsymbol{b}_t,\boldsymbol{a}_t,\boldsymbol{b}_{t+1})+\gamma\tilde{V}(h_t\boldsymbol{a}_t\boldsymbol{z}_t|A_{t+1},Z_{t+1}).$$

One should be reminded that (A_{t+1},Z_t) is sampled based on $p_{\mathcal{T}}(A_{t+1},Z_t)$. Then, the expectation of $\tilde{V}(h_t\boldsymbol{a}_t|A_{t+1},Z_t)$ is

$$\begin{aligned}&\mathop{\mathbb{E}}_{p_{\mathcal{T}}(A_{t+1},Z_t)}\left\{\tilde{V}(h_t\boldsymbol{a}_t|A_{t+1},Z_t)\right\}\\ &=r(\boldsymbol{b}_t,\boldsymbol{a}_t,\boldsymbol{b}_{t+1})+\gamma\sum_{i=0}^{n-1}\mathop{\mathbb{E}}_{p_{\mathcal{T}}(A_{t+1},Z_{t+1})}\\ &\times\left\{\tilde{V}(h_t\boldsymbol{a}_t\boldsymbol{z}_t^i|A_{t+1},Z_{t+1})\right\}\\ &\xrightarrow{a.s.}r(\boldsymbol{b}_t,\boldsymbol{a}_t,\boldsymbol{b}_{t+1})+\gamma\sum_{i=0}^{n-1}V_\pi\big(h_t\boldsymbol{a}_t\boldsymbol{z}_t^i\big)\xrightarrow{a.s.}\\ &r(\boldsymbol{b}_t,\boldsymbol{a}_t,\boldsymbol{b}_{t+1})+\gamma\int_{\boldsymbol{z}_t}V_\pi\big(h_t\boldsymbol{a}_t\boldsymbol{z}_t^i\big)p(\boldsymbol{z}_t|\boldsymbol{a}_t)d\boldsymbol{z}_t=V_\pi(h_t\boldsymbol{a}_t).\end{aligned}$$

The first *a.s.* convergence is based on Lemma 3, whereas the second is, again, from the strong law of large numbers. Therefore, $\tilde{V}(h_t\boldsymbol{a}_t|A_{t+1},Z_t)$ is an unbiased estimate of $V_\pi(h_t\boldsymbol{a}_t)$. ■

Comparing (7) and (15), one might find it striking that $\tilde{V}(h_t\boldsymbol{a}_t|A_{t+1},Z_t)=\tilde{V}(h_t|A_t,Z_t)$. The results should not be overinterpreted leading to the conclusion that $V_\pi(h_t\boldsymbol{a}_t)=V_\pi(h_t)$. The correct statement is $\frac{1}{\eta}\sum_i^{K-1}P(Z_t|X_t^i,A_t)R(S_t^i)$ serves as an unbiased estimate for both $V_\pi(h_t\boldsymbol{a}_t)$ and $V_\pi(h_t)$. The discomfort should be resolved by considering that $V_\pi(h_t)$ is updated (implicitly) when any of its child belief-action nodes is visited while updating $V_\pi(h_t\boldsymbol{a}_t)$ requires $h_t\boldsymbol{a}_t$, one child of h_t, to be visited.

The immediate consequence of Theorem 3 is that Algorithm 2 is a *bona fide* MCTS algorithm. To the best of our knowledge, there is no formal proof yet that MCTS would converge to the optimal action at the root when applied to stochastic systems.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

Kocsis and Szepesvári [53] present conclusions on the convergence of UCT. Unfortunately, the details of the proof are not available. On a related note, it is also commented in [47, Ch. 5] that proving the convergence of Monte Carlo control, a more general algorithm that MCTS bases on, could be a fundamental open question in reinforcement learning.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[3] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 35, no. 7, pp. 814–824, Jul. 1990.

[4] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Belmont, MA, USA: Athena Scientific, 2017, no. 2.

[5] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 1617–1624.

[6] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Proc. Robot.: Sci. Syst.*, pp. 233–241, 2007.

[7] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1038–1052, 2010.

[8] A. Martinelli, "Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 44–60, Feb. 2012.

[9] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *Int. J. Robot. Res.*, vol. 28, no. 11/12, pp. 1448–1465, 2009.

[10] J. van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1263–1278, 2012.

[11] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2164–2172.

[12] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1772–1780.

[13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1/2, pp. 99–134, 1998.

[14] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc. 18th Int. Joint Conf. Artif. Intell.*, 2003, vol. 3, pp. 1025–1032.

[15] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 895–913, 2011.

[16] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Auton. Agents Multi-Agent Syst.*, vol. 27, no. 1, pp. 1–51, 2013.

[17] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDPs," *J. Artif. Intell. Res.*, vol. 32, pp. 663–704, 2008.

[18] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," *J. Artif. Intell. Res.*, vol. 58, pp. 231–266, 2017.

[19] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proc. Robot.: Sci. Syst.*, Zaragoza, Spain, Jun. 2010. [Online]: Available: http://www.roboticsproceedings.org/rss06/p37.html

[20] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 101–115, Feb. 2012.

[21] M. P. Vitus and C. J. Tomlin, "Closed-loop belief space planning for linear, Gaussian systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2152–2159.

[22] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *Int. J. Robot. Res.*, vol. 33, no. 2, pp. 268–304, 2014.

[23] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 849–882, 2015.

[24] W. Sun, J. van den Berg, and R. Alterovitz, "Stochastic extended LQR for optimization-based motion planning under uncertainty," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 437–447, Apr. 2016.

[25] M. Rafieisakhaei, S. Chakravorty, and P. Kumar, "T-LQG: Closed-loop belief space planning via trajectory-optimized LQG," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 649–656.

[26] A. Censi, D. Calisi, A. De Luca, and G. Oriolo, "A Bayesian framework for optimal motion planning with uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 1798–1805.

[27] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 723–730.

[28] A.-A. Agha-Mohammadi, S. Agarwal, S. Chakravorty, and N. M. Amato, "SLAP: Simultaneous localization and planning for physical mobile robots via enabling dynamic replanning in belief space," *IEEE Trans. Robot.*, vol. 34, no. 5, pp. 1195–1214, Oct. 2018.

[29] T. Shan and B. Englot, "Belief roadmap search: Advances in optimal and efficient planning under uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5318–5325.

[30] E. J. Sondik, "The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs," *Operations Research*, INFORMS, vol. 26, no. 2, pp. 282–304, 1978.

[31] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, 2004, pp. 520–527.

[32] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Proc. Robot.: Sci. Syst.*, Zurich, Switzerland, 2008, vol. 2008.

[33] S. Paquet, L. Tobin, and B. Chaib-Draa, "Real-time decision making for large POMDPs," in *Proc. Conf. Can. Soc. Comput. Stud. Intell.*, 2005, pp. 450–455.

[34] S. Ross and B. Chaib-Draa, "AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 2592–2598.

[35] J. Hoey and P. Poupart, "Solving POMDPs with continuous or large discrete observation spaces," in *Proc. Int. Joint Conf. Artif. Intell.*, 2005, pp. 1332–1338.

[36] J. van den Berg, S. Patil, and R. Alterovitz, "Efficient approximate value iteration for continuous Gaussian POMDPs," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1832–1838.

[37] P. Chaudhari, S. Karaman, D. Hsu, and E. Frazzoli, "Sampling-based algorithms for continuous-time POMDPs," in *Proc. Amer. Control Conf.*, 2013, pp. 4604–4610.

[38] Z. N. Sunberg and M. J. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *Proc. 28th Int. Conf. Autom. Plan. Scheduling*, 2018. [Online]: Available: https://www.aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17734

[39] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *Proc. Int. Conf. Learn. Intell. Optim.*, 2011, pp. 433–445.

[40] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee, "Motion planning under uncertainty for robotic tasks with long time horizons," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 308–323, 2011.

[41] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robot. Auton. Syst.*, vol. 83, pp. 15–31, 2016.

[42] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[43] D. Fox, "KLD-sampling: Adaptive particle filters," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 713–720.

[44] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *Proc. Int. Conf. Comput. Games*, 2006, pp. 72–83.

[45] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 2. Belmont, MA, USA: Athena Scientific, 2017, no. 2.

[46] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2/3, pp. 235–256, 2002.

[47] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT press, 2018.

[48] V. Kuleshov and D. Precup, "Algorithms for the multi-armed bandit problem," *J. Machine Learn. Research*, Citeseer, 2014.

[49] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 767–774.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.

[50] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.
[51] G. Grimmett and D. Stirzaker, *Probability and Random Processes*. London, U.K.: Oxford Univ. Press, 2020.
[52] J. Geweke, "Bayesian inference in econometric models using Monte Carlo integration," *Econometrica: J. Econometric Soc.*, vol. 57, pp. 1317–1339, 1989.
[53] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proc. Eur. Conf. Mach. Learn.*, 2006, pp. 282–293.



Ke Sun (Student Member, IEEE) received the B.E. degree in electronics and information engineering from The Hong Kong Polytechnic University, Hong Kong, in 2013, and the M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2015. He is currently working toward the Ph.D. degree in electrical and systems engineering with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA.

His research interests include state estimation and stochastic motion planning with applications to autonomous vehicles.



Brent Schlotfeldt (Student Member, IEEE) received the B.Sc. degree in electrical engineering and computer science from the University of Maryland, College Park, MD, USA, in 2016, and the M.Sc. degree in robotics in 2017 from the University of Pennsylvania, Philadelphia, PA, USA, where he is currently working toward the Ph.D. degree in electrical and systems engineering.

His research interests include planning, control, and state estimation for robotic systems with applications to autonomous driving and active sensing.



George J. Pappas (Fellow, IEEE) received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1998.

He is currently the Joseph Moore Professor and Chair of the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA. He also holds a secondary appointment with the Department of Computer and Information Sciences and Department of Mechanical Engineering and Applied Mechanics. He is a Member of the GRASP Lab and the PRECISE Center. He was the Deputy Dean for Research with the School of Engineering and Applied Science. His research interests include control theory and, in particular, hybrid systems, embedded systems, cyber-physical systems, and hierarchical and distributed control systems, with applications to unmanned aerial vehicles, distributed robotics, green buildings, and bimolecular networks.

Dr. Pappas was the recipient of various awards, such as the Antonio Ruberti Young Researcher Prize, George S. Axelby Award, Hugo Schuck Best Paper Award, George H. Heilmeier Award, National Science Foundation PECASE Award, and numerous Best Student Papers Awards at the American Control Conference, IEEE Conference on Decision and Control, and ACM/IEEE International Conference on Cyber-Physical Systems.



Vijay Kumar (Fellow, IEEE) received the Ph.D. degree in mechanical engineering from The Ohio State University, Columbus, OH, USA, in 1987.

He is currently the Nemirovsky Family Dean of Penn Engineering (with appointments) with the Department of Mechanical Engineering and Applied Mechanics, Department of Computer and Information Science, and Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA. He served as the Assistant Director of robotics and cyber-physical systems with the White House Office of Science and Technology Policy from 2012 to 2014. He is the Founder of Exyn Technologies, a company that develops solutions for autonomous flight.

Dr. Kumar became a Fellow of the American Society of Mechanical Engineers in 2003 and a member of the National Academy of Engineering in 2013. He was the recipient of the 2012 World Technology Network Award, 2013 Popular Science Breakthrough Award, 2014 Engelberger Robotics Award, 2017 IEEE Robotics and Automation Society George Saridis Leadership Award in Robotics and Automation, and 2017 ASME Robert E. Abbott Award. He was also the recipient of Best Paper Awards at the 2002 International Symposium on Distributed Autonomous Robotic Systems (DARS), 2004 IEEE International Conference on Robotics and Automation (ICRA), ICRA 2011, 2011 Robotics: Science and Systems Conference (RSS), RSS 2013, and 2015 EAI International Conference on Bio-Inspired Information and Communications Technologies. He has advised doctoral students, who have received best student paper awards at ICRA 2008, RSS 2009, and DARS 2010.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:04:59 UTC from IEEE Xplore. Restrictions apply.