Monte-Carlo Robot Path Planning

Tuan Dam[1], Georgia Chalvatzaki[1], Jan Peters[1] and Joni Pajarinen[1,2]

Abstract—Path planning is a crucial algorithmic approach for designing robot behaviors. Sampling-based approaches, like rapidly exploring random trees (RRTs) or probabilistic roadmaps, are prominent algorithmic solutions for path planning problems. Despite its exponential convergence rate, RRT can only find suboptimal paths. On the other hand, RRT*, a widely-used extension to RRT, guarantees probabilistic completeness for finding optimal paths but suffers in practice from slow convergence in complex environments. Furthermore, real-world robotic environments are often partially observable or with poorly described dynamics, casting the application of RRT* in complex tasks suboptimal. This paper studies a novel algorithmic formulation of the popular Monte-Carlo tree search (MCTS) algorithm for robot path planning. Notably, we study Monte-Carlo Path Planning (MCPP) by analyzing and proving, on the one part, its exponential convergence rate to the optimal path in fully observable Markov decision processes (MDPs), and on the other part, its probabilistic completeness for finding feasible paths in partially observable MDPs (POMDPs) assuming limited distance observability (proof sketch). Our algorithmic contribution allows us to employ recently proposed variants of MCTS with different exploration strategies for robot path planning. Our experimental evaluations in simulated 2D and 3D environments with a 7 degrees of freedom (DOF) manipulator, as well as in a real-world robot path planning task, demonstrate the superiority of MCPP in POMDP tasks.

Index Terms—Planning under Uncertainty, Motion and Path Planning, Planning, Scheduling and Coordination

I. INTRODUCTION

ROBOT path planning refers to the process of finding a sequence of configurations that lead a robot system from a starting configuration to a goal configuration without violating task constraints. Path planning is a crucial component in robotics [1], autonomous driving [2] and other domains such as surgical planning, computational biology, and molecules [3]. In robotics, path planning is an integral tool for manipulation tasks with robotic manipulator arms [4]–[6] and mobile robots [7]–[9].

Due to the redundancy of robotic arms and the complexity and constraints of real-world tasks, sampling-based approaches yielded significant results [10], [11]. Among the different algorithmic contributions [1], [12]–[14], RRT* [15] is a widely used method that ensures finding the optimal path with probabilistic completeness guarantees [16]. While RRT* is effective

[1]Department of Computer Science, Technische Universität Darmstadt, Germany

[2]Department of Electrical Engineering and Automation, Aalto University, Finland
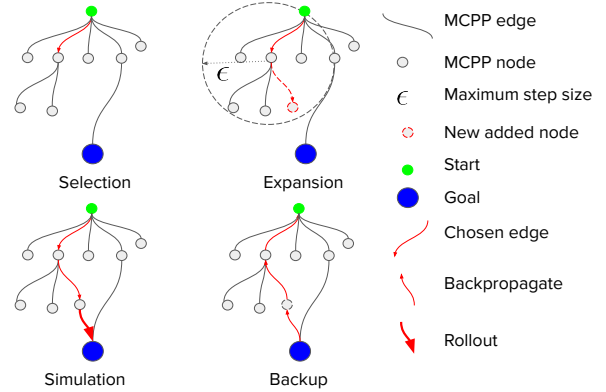
{tuan,georgia,jan,joni}@robot-learning.de

Fig. 1: Four stages of MCPP planner to traverse from the initial position (in green color) to the goal position (in blue color).
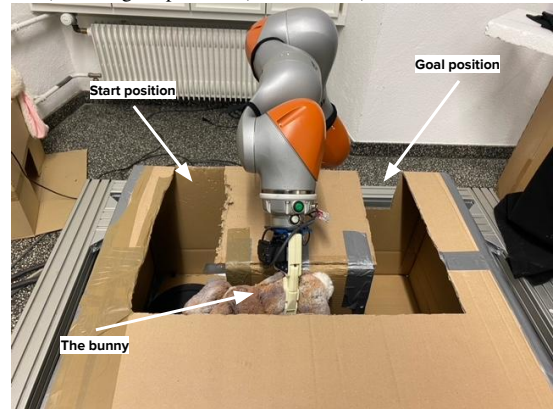

Fig. 2: Demonstration of path planning using MCPP in a robotic disentangling task. A 7-DOF robotic KUKA arm tries to extract an object from the cardboard box through the hole in the back of the box. The robot does not use any sensors except for proprioception, making the task partially observable. Therefore, the task requires advanced MCPP-based path planning that takes information gathering about the environment into account. We put a limit to prevent the robot arm to move the hand up, therefore, the robot arm has to find the path from the start position on the left side to the goal position on the right side inside the box.

in solving path planning tasks in fully observable MDPs, real-world robotics applications are characterized by partial information, casting their settings into POMDP problems. In the real world, robots should make decisions based on information from laser sensors [17], camera images [18], and sensory feedback [19], which generally contains noise, and subsequently makes it hard for planners such as RRT*. Therefore, robot path planning under uncertainty [20]–[23] has become one of the critical topics in the robotics community and remains an open research challenge.

This work proposes an algorithmic formulation to path planning problems based on the popular MCTS algorithm. We argue that the exploration-exploitation properties of MCTS algorithms are essential for robotic path planning in POMDPs, and they can outperform sampling-based planners like RRT* that greedily explore the state-space. To this end, we formulate an MCPP algorithmic framework that we analyze theoretically

and provide proofs of convergence for the MDP and POMPD settings. In particular, when applying the upper confidence bounds for Trees (UCT) algorithm [24], we can guarantee the exponential convergence of MCPP to optimal paths in MDP problems. Crucially, we extend our theoretical analysis to prove the probabilistic completeness of MCPP in POMDP problems assuming limited distance observability. To the best of our knowledge, this is the first work to provide the theoretical analysis for MCTS in both MDP and POMDP robot path plannings. We continue by proposing different exploration strategies in MCPP for robotic path planning. In particular, we build on top of our prior work on power-mean UCT (Power-UCT) [25] and convex regularization with Tsallis Entropy Monte-Carlo Planning (TENTS) [26], integrating them in MCPP. We provide various experimental evaluations of MCPP, initially in MDP environments for completeness and thereafter in challenging POMDP tasks in 2D and 3D while planning with a 7-DOF robot arm. Moreover, we evaluate the different variants of MCPP against RRT* in a real-world POMDP experiment (see Fig. 2), where the robot can only observe collisions in the box while planning to take out a bunny-toy. Our experimental results confirm that MCPP has a higher probability of solving POMDP path planning tasks with less planning time and requiring fewer samples than the baseline methods. We believe that our theoretical findings and empirical results will shed new light on robotic path planning in complex, partially observable tasks. To summarize, our *contribution* is threefold:

- we prove that MCPP enjoys exponential convergence in choosing the optimal path in MDP problems and has convergence guarantees to find a feasible path in POMDP environments with limited distance observability (for the POMDP case, we provide the proof sketch);
- using our theoretical insights, we propose an MCTS-based path planning framework that can incorporate different exploration strategies, such as our state-of-the-art methods, Power-UCT, and TENTS, into POMDP path planning problems;
- we provide empirical evaluations in simulation and real-world experiments that confirm our theoretical findings for the MCPP algorithmic framework to be a promising solution for planning in POMDP environments.

II. RELATED WORK

Probabilistic RoadMaps (PRMs) [13] and RRTs [15] are fundamental approaches for sampling-based motion planning. RRT* improves over RRT by applying the rewiring technique to shorten the unnecessary traversing path. Moreover, RRT* has proven to guarantee probabilistic completeness for choosing the optimal path in MDP problems, but no convergence rate of RRT* has been studied so far.

There are several heuristic improvements over the state-of-the-art RRT and RRT*. For example, A* is a sufficient heuristic path planning-based method for finding an optimal path given the graph representation of the environment. A*-RRT [27] integrates the benefit of the heuristic A* in RRT by sampling a new tree node using an A* path, and therefore

improving the performance in terms of sample efficiency and cost compared to RRT. A*-RRT* [27] combines A* with RRT* to improve the sample efficiency over RRT*. Theta*-RRT [28] considers Theta*, an any-angle discrete search method combined with RRT. Palmieri et al. [28] prove that Theta*-RRT enjoys the probabilistic completeness of RRT and RRT*, while finding shorter trajectories and plans significantly faster than baseline planners (RRT, A*-RRT, RRT*, A*-RRT*). Informed-RRT* [29] focuses the search on the ellipsoidal informed subset of the state-space of the initial running solution found by RRT*.

Regarding applications of MCTS in path planning, Kim et al. [30] proposes the use of Voronoi diagrams to discretize the action space and provides a regret-bound analysis for the sample efficiency, but the authors do not provide a convergence rate for goal reaching in the robot path planning setting. Sun et al. [31] propose POMCP++, an improvement over POMCP [32] to solve continuous observation problems. First, the authors propose using multiple particle samples from the current initial belief instead of a single particle sample of POMCP. Second, the authors handle the continuous observation space by proposing a new measurement sampling method. At each Q-node in the tree, POMCP++ either samples a new observation or chooses existing observations with some probability. Experiments show that POMCP++ yields a significantly higher success rate and total reward. However, there is no actual convergence rate analysis in the robot path planning settings. Sunberg et al. [33] integrated the progressive widening technique in MCTS to discretize the continuous action and observation cases in POMDP settings and derived POMCPOW and POMCP-DPW (with double progressive widening). The authors further combined a weighted particle filter with progressive widening and showed the benefits over the baseline algorithm Determinized Sparse Partially Observable Tree (DESPOT) [34].

Our work uses a simple uniform discretization of the action space for the MCTS algorithm in the context of robotic path planning. While our approach can apply the Voronoi diagram discretization of [30], in this paper, we focus on the theoretical justification of our method and its comparison to sampling-based planners. We provide proofs of convergence for planning the optimal path to the goal in MDPs and a feasible path in POMDPs (with a proof sketch), which is not provided in [31], [33], but can also apply to them. Notably, we propose MCPP as a general MCTS-based framework for robotic path planning. MCPP can incorporate different exploration strategies [25], [26] to continuous actions, adapting, subsequently, the convergence rates for MCPP.

III. BACKGROUND

Markov Decision Process. A finite-horizon MDP can be defined as a 5-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the finite action-space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi \in \Pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a probability distribution of the event of executing an action a in a state s. Most sampling-based

algorithms consider the environment as an MDP. Notably, in robot path planning problems, we know the obstacle space so that when we sample a new vertex, we can determine if the new sampled point lies in the free space or not and then calculate the cost function.

Partially Observable MDP. We consider a finite-horizon POMDP as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{P}_s, \mathcal{P}_o, \gamma \rangle$, where $\mathcal{S}$ is the state-space, $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the finite action-space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P}_s : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition kernel, $\mathcal{P}_o : \mathcal{O} \times \mathcal{A} \to \mathcal{S}$ is the observation dynamics, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi \in \Pi : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is a probability distribution of the event of executing an action a in an observation o. In POMDP settings, the agent does not observe the full information of the state of the environment, and the decisions are based only on observations. In general, the decision process can be made based either on the history of all past actions and observations $h_t = \{a_0, o_0, a_1, o_1, ..., a_t, o_t\}$, or through the belief of the agent over the state-space [32].

Monte-Carlo Tree Search. MCTS [35] combines tree search with Monte-Carlo sampling in order to build a tree, where states and actions are modeled as nodes and edges, respectively, to compute optimal decisions. The MCTS algorithm consists of a loop of four steps: *Selection*: start from the root node, interleave action selection and sample the next state (tree node) until a leaf node is reached; *Expansion*: expand the tree by adding a new edge (action) to the leaf node and sampling the next state (new leaf node); *Simulation*: rollout from the reached state to the end of the episode using random actions or a heuristic; *Backup*: update the nodes backward along the trajectory starting from the end of the episode until the root node according to the rewards collected.

UCT [24], [36] is an extension of the well-known UCB1 [37] multi-armed bandit algorithm. UCB1 chooses the arm (action a) using

$$a = \operatorname*{arg\,max}_{i \in \{1...K\}} \overline{Q}_{i, T_i(n-1)} + C \sqrt{\frac{\log n}{T_i(n-1)}}. \quad (1)$$

where $T_i(n) = \sum_{t=1}^{n} \mathbf{1}\{t = i\}$ is the number of times arm i is played up to time n, and $\overline{Q}_{i, T_i(n-1)}$ is the average reward of arm i up to time $n-1$ and $C = \sqrt{2}$ is an exploration constant. In UCT, the value of each node is backed up recursively from the leaf node to the root node as averaging over the child nodes. At each action selection step in MCTS, each arm in the tree is chosen as the maximum value of nodes in the current non-stationary multi-armed bandit setup, as in (1). UCT ensures the asymptotic convergence of choosing the optimal arm at the root node [36].

Power-UCT [25], an improvement over UCT, solves the problem of the underestimation of the average mean and the max-backup operators in MCTS by proposing the use of power mean as the backup operator. Power-UCT has a polynomial convergence rate for choosing the optimal action at the root node. TENTS [26] is derived as a result of Tsallis entropy regularization in MCTS. TENTS has an exponential convergence rate at the root node, which is faster than Power-UCT and UCT. TENTS has a lower value error and smaller regret bound at the root node compared to other regularization approaches.

IV. PROBLEM FORMULATION

Let us define the robot path planning problem, both for MDPs and POMDPs. Let $\mathcal{X} = (a, b)^d$ be the configuration space of the robot, where $a, b \in \mathcal{R}$ are joint limits in configuration space, with $a < b$, and $d \in \mathcal{N}, d > 0$ denoted the robot's DOF. Let's define $\mathcal{X}_{\text{OBS}}$ as the obstacles region and $\mathcal{X} \backslash \mathcal{X}_{\text{OBS}}$ the open set, and the obstacle-free space as $\mathcal{X}_{\text{FREE}} = cl(\mathcal{X} \backslash \mathcal{X}_{\text{OBS}})$, where $cl(\cdot)$ denotes the closure of a set. The initial condition, or start region, $\mathbf{x}_{\text{INIT}}$ is an element of $\mathcal{X}_{\text{FREE}}$, and the goal region $\mathbf{x}_{\text{GOAL}}$ is an open subset of $\mathcal{X}_{\text{FREE}}$. A path planning problem is defined by the triplet $(\mathcal{X}_{\text{FREE}}, \mathbf{x}_{\text{INIT}}, \mathbf{x}_{\text{GOAL}})$. A trajectory is defined as the map $\tau : [0, T] \to \mathcal{X}_{\text{FREE}}$, where $\tau(0) = \mathbf{x}_{\text{INIT}}, \tau(T) = \mathbf{x}_{\text{GOAL}}$. Let's define a function $\sigma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as the cost function for moving the robot from the configuration point $\mathbf{x}_i$ to $\mathbf{x}_j$, where $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$. A solution to such a problem is a trajectory that moves the robot from the initial region to the goal region, while avoiding collisions with obstacles and having minimum cost.

Fully observable problem. Here, we assume that we know the state of the environment, i.e., we know the $\mathcal{X}_{\text{OBS}}$ space and $\mathcal{X}_{\text{FREE}}$ regions. Whenever a new point is sampled in the configuration space $\mathcal{X} = (a, b)^d$, we can measure the cost and determine if the point is inside the free space or not.

Partially observable problem. In this setting, we assume that the environment is partially observable, i.e., we only know the start position and the goal position. We do not observe the full state but only observations of the environment and progressively build a belief about the environment's state from observations.

V. MONTE-CARLO PATH PLANNING

We wish to transform MCTS into a sampling-based method for solving robot path planning problems when applicable. We build our proposed MCPP approach starting from the UCT algorithm. MCPP and UCT share similar ways of selecting nodes to traverse and back up the value of nodes in the tree. However, we need to make several algorithmic choices to do path planning with UCT. First, we draw an ϵ-ball to limit the maximum distance that the robot can move from the current configuration point. Second, we perform uniform sampling of the configuration points inside the ϵ-ball to discretize the continuous actions in the MDP. Third, we investigate different exploration strategies for MCPP, like in the PowerUCT and TENTS algorithms. We provide a proof of the exponential convergence rate of finding the optimal path in MDPs. Moreover, we connect this result to Power-UCT and TENTS and derive their respective convergence rates for path planning. In POMDPs, we provide a probabilistic completeness guarantee for finding the feasible path to the goal with limited distance observations.

A. Fully observable environment

In an MDP, the agent knows the full state of the environment. Let us define the start position as $\mathbf{x}_{\text{INIT}}$ and the goal position as $\mathbf{x}_{\text{GOAL}}$. The cost function is the Euclidean distance $d(\mathbf{x}, \mathbf{y})$ between two points $\mathbf{x}$ and $\mathbf{y}$. We want to minimize the
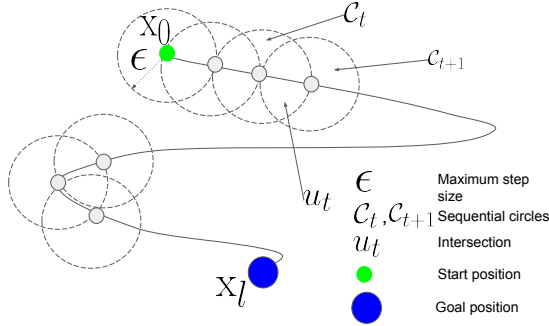
Fig. 3: 2-D sketch of the proof for exponential convergence of MCPP to the optimal path in MDPs. The MDP proof relies on showing that MCPP convergences exponentially to a path starting from x_0 and ending at x_l while the agent stays inside a tube composed of a sequence of spheres with a radius of ϵ.

total cost, that is, the total distance traveled from the start to the goal position using MCPP. As shown in Fig. 1, at each node of the tree, starting from the root node, actions are generated by uniformly sampling random points in the ϵ-ball distance from the current node.

The Algorithm 1 provides the pseudocode of the MCPP method in the MDP case. The `MainLoop` procedure is the main loop of the algorithm. The algorithm stops when the x_{GOAL} position is reached. The algorithm follows the four basic steps of a regular MCTS method. First, at the _Selection_ step, we determine the next node to traverse in the tree by selecting the action as in the `SelectAction` procedure. Here, an action is selected based on the UCB algorithm. Note that when we implement Power-UCT [25] we also use UCB, while TENTS [26] uses stochastic Tsallis entropy regularization for the action sampling. Second, at the _Expansion_ step, $|A|$ number of actions are generated by uniformly sampling inside the circle $\mathcal{C}(s, \epsilon)$ as shown in the `Expand` procedure. When we reach the leaf node, a new node is created and is added to the MCTS tree. Third, at the _Simulation_ step, as shown in the `Rollout` procedure, the value function of the current node s is calculated as the distance from that node s to the goal position. Finally, at the _Backup_ step, the return value is backpropagated in the two procedures `SimulateV`, `SimulateQ`.

B. Partially observable environment

Under partial observablity, the agent does not observe the full state of the environment, but has only access to possibly noisy observations. The MCPP planner makes decisions based on the current belief of the agent over the state of the environment. Therefore, our approach in POMDP will be the same as in MDP, except for the fact that we do the planning in the belief space. The other choice is that MCPP planner can make decisions over the history of actions and observation as if it has some sufficient statistic [38], [39].

C. Theoretical analysis

In this section, we prove that MCPP ensures an exponential convergence rate for finding the optimal path from the start position to the goal position in an MDP environment. In a POMDP setting, we prove that there is a high probability that MCPP can find the path to the goal position.

1) MDP: First, we make the following assumption.

Assumption 1. _There exists an optimal path from the start position x_{INIT} to the goal position x_{GOAL} with δ clearance (minimum distance to an obstacle)._

Based on this assumption, we derive a theorem for the convergence rate of finding the optimal path using MCPP:

Theorem 1. _The probability that MCPP fails to find the optimal path from x_{INIT} to x_{GOAL} after n simulations is at most $ae^{-bf(n)n}$, for some constants $a, b \in R_{>0}$._

Proof. Let us consider all feasible paths from the start position (x_{INIT}) to the goal position (x_{GOAL}). We will prove that MCPP ensures probabilistic completeness of finding the shortest path from (x_{INIT}) to (x_{GOAL}). We will further prove that the failure probability of finding the shortest path decays exponentially for an infinite number of samples.

We choose a ball with radius $\epsilon = \delta$, where δ is the clearance of the shortest path τ^*. Along the path τ^*, we define a set of $l + 1$ circles with the radius ϵ and the center $x_t (t = 0...l) \in \mathcal{X}_{\text{FREE}}$. Here $x_0 = x_{\text{INIT}}$ and $x_l = x_{\text{GOAL}}$, as shown in Fig. 3. We define each circle $\mathcal{C}_t = (x_t, \epsilon), t = 0, 1, ...l$. We define the intersection set $u_t = \mathcal{C}_t \cap \mathcal{C}_{t+1}$. Let p be the probability that MCPP can move from $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$. Consider starting from planning node x_t, which is the center of the circle $\mathcal{C}_t$. If the next planning node x_{t+1} lies in the circle $\mathcal{C}_{t+1}$, it has to lie inside the intersection u_t, and we can see that $p < 1/2$. For the robot to travel from x_0 to x_l, it has to use at least l MCPP vertices. Let the probability that MCPP chooses the best action (action with smallest cost) be $f(n)$. Therefore, the probability of MCPP of taking an optimal action that also lies inside u_t is $f(n)p$. The failure probability that MCPP cannot find the shortest path τ^* from x_0 to x_l is $\mathbf{Pr}(X_n < l)$ where X_n is the number of circles $\mathcal{C}_t, t \in 1, 2, ...l$ which are connected by vertices, that is, for an optimal path, all circles need to be connected by vertices. To calculate X_n, the initial value of X_n is zero. We will incrementally increase X_n by one when a new circle along the optimal path is connected with a new vertex. When X_n is equal to or greater than l, we, then, have found the optimal path. Let us upper bound the failure probability $\mathbf{Pr}(X_n < l)$ by first upper bounding $\mathbf{Pr}(X_n = h)$, as

$$\mathbf{Pr}(X_n = h) \le \binom{n}{h}(f_0 p)^h (1 - f_0 p)^{n-h} \quad (2)$$

$$\le \binom{n}{h}(f(n)p)^h (1 - f_0 p)^{n-h},$$

where $(f(n)p)^h$ is the upper bound probability of having h circles connected by vertices and $\binom{n}{h}$ makes sure there is at least one consecutive sequence of connected circles. f_0 is the initial probability of choosing the optimal action. (2) can be explained as $H(x) = x^h (1 - x)^{n-h}$ is a decreasing function. This yields an upper bound for $\mathbf{Pr}(X_n = h)$:

$$\mathbf{Pr}(X_n = h) \le \binom{n}{h}(f(n)p)^h (1 - f_0 p)^{n-h}$$

$$\le \binom{n}{h}(f(n)p)^h (1 - \alpha f(n)p)^{n-h}$$

where $f_0 = \alpha$ is a constant and $f(n) \le 1$ so that $1 - f_0 p \le 1 - \alpha f(n)p$. The probability of failing to find the optimal path is then

```
1   |A|: number of actions.
2   N(s): number of simulations of V_Node. Default is 0.
3   n(s, a): number of simulations of Q_Node. Default is 0.
4   r, r(s, a): intermediate rewards defined as the distance between two
      nodes.
5   V(s): Value of V_Node at state s. Default is 0.
6   Q(s, a): Value of Q_Node. Default is 0.
7   γ: discount factor. Default is 1.

8   R = Rollout(s)
9       R = Distance from the current node s to the goal position.
10      return R

11  a = SelectAction(s)
12      return arg max_a Q(s, a) + C √(log N(s) / n(s,a))

13  a = Search(s)
14      while Time remaining do
15          SimulateV(s)
16      end
17      return arg max_a Q(s, a)

18  R = Expand(s)
19      Generate |A| actions for the current node s by randomly
          sampling |A| via-points inside the circle C(s, ε)
20      discountedReward = Rollout(s)
21      return discountedReward

22  SimulateV(s)
23      a = SelectAction(s)
24      SimulateQ(s, a)
25      N(s) = N(s) + 1
26      V(s) = ( Σ_a (n(s,a)/N(s)) Q(s, a) )

27  SimulateQ(s, a)
28      (s', r) ~ τ(s, a)
29      if V(s') not expanded then
30          r = r + γ.Expand(s')
31      else
32          SimulateV(s')
33      end
34      r(s, a) = r(s, a) + r
35      n(s, a) = n(s, a) + 1
36      Q(s, a) = (Σ_a r_{s,a} + γ. Σ_{s'} N(s').V(s')) / n(s,a)
37      where V(s') is the value function of the next node by action a
          from the current Q(s, a) node

38  MainLoop
39      while Xgoal is reached do
40          a = Search(s)
41          Execute(a) in real Robot
42      end
```

Algorithm 1: Pseudocode of MCPP.

$$\mathbf{Pr}(X_n < l) = \sum_{h=0}^{l-1} Pr(X_n = h)$$

$$\leq \sum_{h=0}^{l-1} \binom{n}{h} (f(n)p)^h (1 - \alpha f(n)p)^{n-h}$$

$$\leq \sum_{h=0}^{l-1} \binom{n}{l-1} (f(n)p)^h (1 - \alpha f(n)p)^{n-h} \text{ (as } l << n)$$

$$\leq \binom{n}{l-1} \sum_{h=0}^{l-1} (1 - \alpha f(n)p)^n$$

(as $f(n)p < 1/2$ so that $f(n)p < 1 - f(n)p < 1 - \alpha f(n)p$)

$$\leq \binom{n}{l-1} \sum_{h=0}^{l-1} (e^{-\alpha f(n)p})^n = \binom{n}{l-1} l e^{-\alpha f(n)pn}$$

(as $1 - \alpha f(n)p <= e^{-\alpha f(n)p}$)

$$= \frac{\prod_{i=n-l}^{n} i}{(n-1)!} l e^{-\alpha f(n)pn} \leq \frac{l}{(l-1)!} n^l e^{-\alpha f(n)pn} \leq a e^{-b f(n)pn}$$

The provided convergence rate proves that the MCPP algorithm is probabilistically complete and converges to the optimal path exponentially. □

Let us define $g(t)$ as the failure probability of finding the optimal path from $\mathbf{x}_{INIT}$ to $\mathbf{x}_{GOAL}$ after t time steps. We derive the following corollaries:
With Power-UCT, $f_{\text{Power-UCT}} = 1 - (\frac{1}{t})^\alpha)t$. The probability that the MCPP using Power-UCT fails to find the path from x_0 to x_l is as follows.

Corollary 1. *Power-UCT:*
$g_{Power\text{-}UCT}(t) = ae^{-b(1-(\frac{1}{t})^\alpha)t}, where \ 0 < \alpha < 1, a, b \in \mathbb{R}_{>0}$

With TENTS $f_{\text{TENTS}} = 1 - ct \exp\{-\frac{t}{\hat{c}(\log t^3)t}\}$. The probability that MCPP using TENTS fails to find the path from x_0 to x_l is
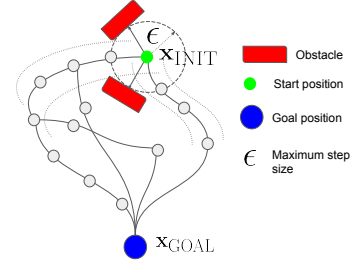

Fig. 4: Sketch of how to generate paths for MCPP algorithm from $\mathbf{x}_{INIT}$ to $\mathbf{x}_{GOAL}$ positions with minimum number of via-points in POMDP environments.

Corollary 2. *TENTS:*
$g_{TENTS}(t) = ae^{-b(1-ct \exp\{-\frac{t}{\hat{c}(\log t^3)t}\}}, where \ a, b, c, \hat{c} \in \mathbb{R}_{>0}$

The results show that MCPP-TENTS robot path planning converges faster compared to MCPP-Power-UCT.

2) POMDP: First, we make the following assumption.

Assumption 2. *The agent observes the environment only up to γ distance.*

This assumption is reasonable in many robotic settings, e.g., for mobile robotics. Based on this assumption and Assumption 1, we derive a theorem to show that with high probability, the MCPP algorithm can find the feasible path to the goal position in a POMDP environment:

Theorem 2. *In POMDP environments with limited distance observability, MCPP will find a path from the start position $\mathbf{x}_{INIT}$ to the goal position $\mathbf{x}_{GOAL}$ with high probability.*

Proof. We assume that there is a finite number of feasible paths $(\tau_1, \tau_2, ...\tau_K)$ to go from the start position $\mathbf{x}_{INIT}$ (or $\mathbf{x}_0$) to the goal position $\mathbf{x}_{GOAL}$. Each feasible path τ_i

has at least δ_i clearance from obstacles. We choose $\epsilon = \min\{\delta_1, \delta_2, ... \delta_K, \gamma\}$. γ is the observation distance defined in Assumption 2.

Along each path τ_i, let us define a set of circles $\mathcal{C}_i = (\mathbf{x}_i, \epsilon), i = 0, 1, ... l_{\tau_i}$. as shown in Fig. 3. Let us define $\mathbf{C}$ as the set of all circles (along all the feasible paths that we define). We assume that if the agent collides, the agent moves back to the last planning point and will not go to the direction of the obstacle again with high probability. We define that the probability $p_{\text{collision}} \to 0$ when the time step $t \to \infty$. We prove that with high probability, the agent can find the path from the start position $\mathbf{x}_{\text{INIT}}$ to the goal position $\mathbf{x}_{\text{GOAL}}$.

The proof is derived by induction. From the start position $\mathbf{x}_0$, there is a finite number of circles $\mathcal{C}_i \in \mathbf{C}$ as the next feasible region that the MCPP planner can sample as the next node in the tree (MCPP samples the next planning point inside the ϵ-ball distance). Because the probability of colliding again is $p_{\text{collision}} \to 0$ when the time step $t \to \infty$, and the MCPP objective is to minimize the cost to go to the goal position. When we increase the number of samples, the next planning node will lie inside the circle that contains the optimal path. Therefore, with high probability $1 - p_{\text{collision}}$ the next MCPP node will be inside one of the circles $\mathcal{C}_i$. Assume now that the agent is inside the circle $\mathcal{C}_i$. Using the same induction, there is high probability $1 - p_{\text{collision}}$ that the next MCPP node will be inside one of the next circle $\mathcal{C}_{i+1}$. $\mathcal{C}_{i+1} \in \mathbf{C}$. Since the number of circles is finite, the agent will get to the goal region after a certain number of time steps with high probability, concluding the proof. □

VI. EXPERIMENTS

In this section, we evaluate the performance MCPP in challenging POMDP environments. In MCPP, we apply the two recent advanced improvement techniques in MCTS, Power-UCT and TENTS, along with the baseline MCTS method, UCT. We compare our new robot path planning methods against the baseline sampling-based method RRT*, and a state-of-the-art continuous action POMDP solver POMCP-DPW [33]. In simulation, we also compared against two different simple heuristic methods. The first method puts a ball around the agent to sample the next point. We use the same step size (the ball's diameter) and the same number of samples as MCPP and RRT*. In the second heuristic, we use an ϵ-greedy probability (1%) to sample the goal position or otherwise the next node, similarly to random node sampling in RRT*. We do not put any restrictions on the step size to sample the next node.

The POMDP setting of our experiment is the same as in [40]. Similarly to the path planning definition in Sec. IV, the state space $\mathcal{X}$ consists of configuration space coordinates such as robot joint angles. The action space $\mathcal{A}$ is identical to the state space, consisting of target configuration space coordinates defining where to move the robot. We use linear interpolation to move the robot from the current configuration to the next one. The observation is a configuration in collision with an obstacle in obstacle space $\mathcal{X}_{\text{OBS}}$, or the goal configuration $\mathbf{x}_{\text{GOAL}}$, when the robot reaches the goal. We assume

static obstacles and deterministic transition and observation probabilities $\mathcal{P}_s$ and $\mathcal{P}_o$. We define the reward function as a success pseudo-probability along the path from one configuration ($\mathbf{x}_1$) to another ($\mathbf{x}_2$): $\mathcal{R}(\mathbf{x}_1, \mathbf{x}_2) = P_{\text{SUCCESS}}(\mathbf{x}_1, \mathbf{x}_2)$ where $P_{\text{SUCCESS}}(\mathbf{x}_1, \mathbf{x}_2)$ is defined in [40]. We set the discount factor $\gamma = 1$ and limit the planning horizon. As in [40], to approximate the belief over states, based on prior collisions, we compute a probabilistic map that assigns a probability of colliding to any given position in the environment. The belief distribution is able to represent multi-modal and asymmetric belief distributions (see Fig. 2 in [40]). Initially, we assume a non-zero probability of colliding at any location on the map. After each collision, we update the map by assigning a failure probability that takes into account the collision coordinates and the movement direction (see Fig. 2 in [40]).

Following the previous POMDP definition, we evaluate the methods in simulation in two 7-DOF configuration space POMDP tasks with 2D and 3D task spaces. Finally, we compare MCPP to RRT* in a real robot POMDP disentangling application, similar to the one described in [40].

A. Experimental evaluation in simulation

We provide three simulation settings in 2D and 3D state spaces to demonstrate that the MCPP planner is more explorative and can easily solve POMDP path planning problems compared to baselines. First, in a 2D U-Shape problem (Fig. 5), the start position is in green color while the goal position is in blue. We compare UCT, Power-UCT, and TENTS compared to RRT* and POMCP-DPW. As shown in Table I, over 100 random seeds with the same number of samples (500), UCT and Power-UCT obtain 93% and 95% success rate, respectively, with approximately the same number of collisions. TENTS is less explorative, with 31% success rate and 18.3 collisions. POMCP-DPW gets 46% success rate and 22.7 collisions while RRT* gets 76% success rate with 14.6 collisions. The benefits of MCPP over RRT* can be explained as, even using a similar representation with the updating belief (probabilistic map), MCPP makes decisions based on the value function of the POMDP (by building a multistep look-ahead forward tree search), which is more explorative towards the goal. In contrast, each step decision of RRT* will be more greedy in choosing the smallest cost. Meanwhile, MCPP shows the benefit of uniformly sampling the actions inside the ϵ-ball compared to POMCP-DPW, which restricts the number of actions by using the progressive widening technique. Both of the two random heuristic baselines fail to solve this task. We demonstrate one more 2D POMDP experiment with an L shape obstacle (Fig. 6. Over 20 random seeds with 500 samples, RRT* fails to solve the problem, while UCT and Power-UCT obtain 100% success rate. TENTS is less explorative with 85% success rate. POMCP-DPW obtains 85% success rate. The two heuristic methods fail to solve this task.

Third, we build a High-wall 3D POMDP (Fig. 7). In this problem, the start position (green color) and the goal position (blue color) are very close, while there is a high wall standing between. The agent is not aware of the existence of the wall.
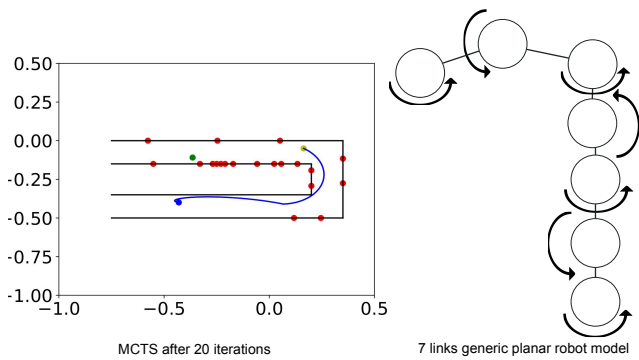
Fig. 5: U-Shape 2D POMDP. Green point is the start position. Blue point is the goal position. Red points are the collisions. The figure shows a success case using MCPP, where the blue line depicts the 2D trajectory of the end effector. Note that in all the 2D experiments we plan in the configuration space using a 7-DOF planar robot arm model illustrated on the right.
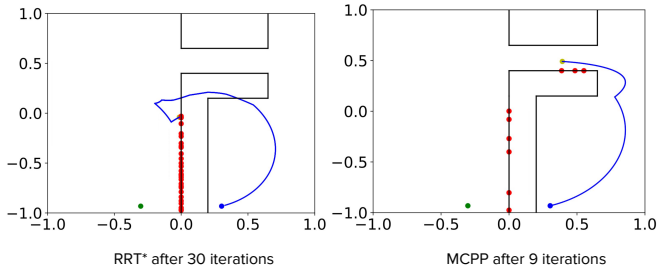

Fig. 6: L-Shape 2D POMDP. Green point is the start position. Blue point is the goal position. Red points are the collisions. The blue lines are the planning path. The figure shows a failure case of RRT* and a success case for MCPP, which shows that it is more explorative. Over 20 random seeds, RRT* failure to solve the problem with 0% success, while MCPP obtain 100% success with UCT and Power-UCT. TENTS gets 85%.
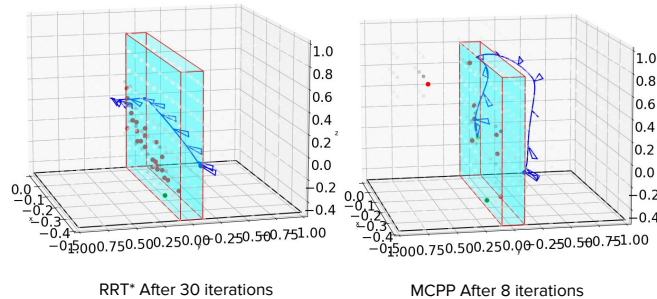

Fig. 7: High Wall Environment in 3D. Grey point is the start position. Blue point is the goal position. Red points are collisions. Over 20 random seeds, RRT* can only success with 35%, UCT obtains 55% success rate. Power-UCT gets 70% success rate while TENTS gets 45% success rate.

As we can see in Table II, for 20 random seeds with the same number of samples (500), UCT obtains 55% success rate with 16.4 collisions on average. Power-UCT gets a higher success rate with 70% and 15.7 collisions on average. TENTS is less explorative with 45% success rate and 23.0 collisions on average. POMCP-DPW gets 70% success rate and 15.9 collisions. On the other hand, RRT* can only obtain 10% success rate and 26.0 collisions. Finally, the first baseline heuristic method fails to solve this task, while the second one achieves 15% success rate.

B. Real robot object disentangling task

We compare MCPP against RRT* in the real-robot disentangling POMDP problem, as in [40]. We use a 7-DOF KUKA LBR robot arm equipped with a SAKE gripper. Fig. 2

TABLE I: Comparison for the U-Shape 2D POMDP

Methods	Time(second)	Collisions	Success Rate
RRT*	1555±229	14.6±1.5	76%
UCT	141.7±16.3	15.9±1.7	93.0%
Power-UCT	146±17	15.8±1.7	95.0%
TENTS	179.5±16	18.3±1.7	31%
POMCP-DPW	322±30.8	22.7±1.6	46%

TABLE II: Comparison for the High-Wall 3D POMDP

Methods	Time(second)	Collisions	Success Rate
RRT*(bias=1)	2854.6±4.1	26.0±0.4	10%
RRT*(bias=100)	2080.3±2.6	19.4±1.7	35%
RRT*(bias=200)	2548.1±2.7	22.2±1.	55%
UCT	178.8±13.0	16.4±1.0	55%
Power-UCT	208±18.5	15.7±1.2	70.0%
TENTS	267.6±17.1	23.0±1.2	45.0%
POMCP-DPW	215.56±23.9	15.9±1.4	70.0%

TABLE III: Comparison for the real robot object disentangling

Methods	Time(second)	Collisions	Success Rate
RRT*	1099 ±356	10.25 ±3	40%
UCT	346.2 ±64	20.2 ±3	70%
Power-UCT	436.5 ±177	22.5 ±6	40.0%
TENTS	428.5 ±133	25 ±6	20.0%

illustrates the intermediate scenario of the robot arm trying to reach the goal position in the unknown box environment, while trying to disentangle the toy-bunny that was lying inside the box (the grasp part was pre-programmed). The robot does not know the shape of the box.

To evaluate the performance of our MCPP variants, we run both UCT, Power-UCT, and TENTS. We run 10 random seeds, each random seed with 500 number of samples, and perform 30 iterations to determine if the planners can reach the goal position or not. After each iteration, if the robot hits a collision, the robot moves back a bit from the last collision position and performs the planning again with the new start position. The detailed results are shown in Table III. While RRT* can get 40% success rate over ten random seeds, UCT achieves 70% which shows the benefits of MCPP in a real-world POMDP. Power-UCT achieves 40% success rate, and TENTS can only succeed 20% of the times. In terms of time, the average time in all success cases of UCT, Power-UCT, and TENTS are 364.2 seconds, 436.5 seconds, and 428.5, respectively, which are much faster compared to RRT* with an average of 1099 seconds. This can be explained as RRT* spends more time in performing the sorting to find the nearest node, as it is more biased to grow towards large unsearched areas.

VII. CONCLUSIONS

This paper addressed the major challenges of planning robot paths under partial observability from a theoretical perspective, deriving a new framework for applying MCTS planning in continuous action spaces for robot path planning. We theoretically analyzed our proposed Monte-Carlo Path Planning (MCPP) approach and proved an exponential convergence rate for MCPP for choosing the optimal path in fully observable MDPs and probabilistic completeness for

finding a feasible path in POMDPs. Moreover, MCPP allows us to integrate different established exploration strategies in MCTS literature to improve exploration for path planning. We empirically analyzed our MCPP variants in benchmarks for POMDP path planning problems, showing superiority in terms of performance and computational time compared to RRT*, and POMCP-DPW. We further applied our new method to a real robot POMDP problem using a KUKA 7-DOF robot arm to disentangle objects from other objects, without any sensory information, except for the observation of collisions. Future development involves the application of MCPP to more complicated robotic tasks and studying heuristics to accelerate the planning process with MCPP.

REFERENCES

[1] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2016.

[3] J.-C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *The International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[4] G. Sahar and J. M. Hollerbach, "Planning of minimum-time trajectories for robot arms," *The International journal of robotics research*, vol. 5, no. 3, pp. 90–100, 1986.

[5] B. K. Kim and K. G. Shin, "Minimum-time path planning for robot arms and their dynamics," *IEEE transactions on systems, man, and cybernetics*, no. 2, pp. 213–223, 1985.

[6] T. Kunz, U. Reiser, M. Stilman, and A. Verl, "Real-time path planning for a robot arm in changing environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 5906–5911.

[7] A. Zelinsky, R. A. Jarvis, J. Byrne, S. Yuta *et al.*, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proceedings of international conference on advanced robotics*, vol. 13. Citeseer, 1993, pp. 533–538.

[8] C. Alexopoulos and P. M. Griffin, "Path planning for a mobile robot," *IEEE Transactions on systems, man, and cybernetics*, vol. 22, no. 2, pp. 318–322, 1992.

[9] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, "Path planning for the mobile robot: A review," *Symmetry*, vol. 10, no. 10, p. 450, 2018.

[10] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[11] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: a survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[12] S. M. Persson and I. Sharf, "Sampling-based a* algorithm for robot path-planning," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 2014.

[13] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, S. Sorkin *et al.*, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: the algorithmic perspective: 1998 workshop on the algorithmic foundations of robotics*, 1998, pp. 141–154.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[15] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[16] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Robotics and Automation Letters (RAL)*, vol. 4, no. 2, pp. x–xvi, 2018.

[17] M. Ivanov, L. Lindner, O. Sergiyenko, J. C. Rodríguez-Quiñonez, W. Flores-Fuentes, and M. Rivas-Lopez, "Mobile robot path planning using continuous laser scanning," in *Optoelectronics in machine vision-based theories and applications*. IGI Global, 2019, pp. 338–372.

[18] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE transactions on robotics and automation*, vol. 18, no. 4, pp. 534–549, 2002.

[19] V. J. Lumelsky, "Dynamic path planning for a planar articulated robot arm moving amidst unknown obstacles," *Automatica*, vol. 23, no. 5, pp. 551–570, 1987.

[20] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[21] N. Dadkhah and B. Mettler, "Survey of motion planning literature in the presence of uncertainty: Considerations for uav guidance," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 233–246, 2012.

[22] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion- and uncertainty-aware path planning for micro aerial vehicles," *Journal of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

[23] M. Kazemi, K. Gupta, and M. Mehrandezh, "Path-planning for visual servoing: A review and issues," *Visual Servoing via Advanced Numerical Methods*, pp. 189–207, 2010.

[24] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[25] T. Dam, P. Klink, C. D'Eramo, J. Peters, and J. Pajarinen, "Generalized mean estimation in monte-carlo tree search," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020, pp. 2397–2404. [Online]. Available: https://doi.org/10.24963/ijcai.2020/332

[26] T. Q. Dam, C. D'Eramo, J. Peters, and J. Pajarinen, "Convex regularization in monte-carlo tree search," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2365–2375.

[27] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 5539–5544.

[28] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2775–2781.

[29] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[30] B. Kim, K. Lee, S. Lim, L. Kaelbling, and T. Lozano-Pérez, "Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 9916–9924.

[31] K. Sun, B. Schlotfeldt, G. J. Pappas, and V. Kumar, "Stochastic motion planning under partial observability for mobile robots with continuous range measurements," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 979–995, 2020.

[32] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," *Advances in neural information processing systems*, vol. 23, 2010.

[33] Z. N. Sunberg and M. J. Kochenderfer, "Online algorithms for pomdps with continuous state, action, and observation spaces," in *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

[34] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *Advances in neural information processing systems*, vol. 26, 2013.

[35] H. Baier and P. D. Drake, "The power of forgetting: Improving the last-good-reply policy in monte carlo go," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 4, pp. 303–309, 2010.

[36] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved monte-carlo search," *Univ. Tartu, Estonia, Tech. Rep*, vol. 1, 2006.

[37] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[38] T. Smith and R. G. Simmons, "Point-based pomdp algorithms: Improved analysis and implementation," in *UAI*, 2005.

[39] D. Braziunas, "Pomdp solution methods," *University of Toronto*, 2003.

[40] J. Pajarinen, O. Arenz, J. Peters, and G. Neumann, "Probabilistic approach to physical object disentangling," *IEEE Robotics and Automation Letters (RAL)*, vol. 5, no. 4, pp. 5510–5517, 2020.