Vectorized Online POMDP Planning with Neural POMDP Models

Marcus Hoerger*, Rishikesh Joshi*, Rahul Shome*, Ian Manchester†, and Hanna Kurniawati*
*Australian National University, Canberra, ACT, Australia.
†The University of Sydney, Sydney, NSW, Australia.

***Abstract*—Partially Observable Markov Decision Processes (POMDPs) provide a principled framework for decision making under uncertainty, but existing solution approaches either rely on hand-crafted models or require large amounts of task-specific training data. Recent methods have combined planning and learning by integrating neural POMDP models into search-based decision making. However, these approaches often rely on sequential or CPU-bound planning algorithms, limiting efficient utilization of neural POMDP models, whose evaluation naturally maps to large batched tensor operations on modern GPUs. In this paper, we present *Vectorized Online planning wIth Learned diffusion model for POMDP Agents* (VOiLA), a framework that combines learned neural POMDP models with massively parallel online planning. Central to VOiLA is *Vectorized Online POMDP Planner* (VOPP), a fully GPU-native online POMDP solver that represents belief trees as tensors and performs planning through tens of thousands of parallel simulations. Building on this architecture, VOiLA learns task-agnostic neural transition, observation, and likelihood models from offline simulation data and integrates them directly into planning and belief estimation. Experimental results show that VOPP achieves planning performance comparable to state-of-the-art online POMDP solvers while using up to $1000\times$ smaller planning budgets. Furthermore, VOiLA achieves competitive performance while requiring less than 10% of the training data used by recurrent reinforcement learning, stronger generalization to unseen environments, and successful sim-to-real transfer.**

I. Introduction

In the real world, robots must operate reliably despite imprecise actuators, noisy sensors, perception errors, and incomplete knowledge of their environments. Partially Observable Markov Decision Processes (POMDPs) provide a principled mathematical framework for reasoning under various types of uncertainty by maintaining a belief, i.e., a probability distribution over possible states, enabling robots to act effectively even when the outcomes of their actions, their own internal state, and the state of the world are not fully known.

Although solving POMDPs exactly is computationally intractable [13], approximate methods have largely evolved along two directions: online POMDP planning [8], which requires POMDP models to be available a priori, and reinforcement learning approaches, starting with [4], which relax this requirement, but often demand substantial training data.

Methods have been proposed to combine the strengths of planning and learning. Some [2, 10, 11, 16] integrate search tree structures with learned neural POMDP models, heuristics, or value estimates. However, these methods often rely on single-threaded or CPU-bound planning components, limiting efficient utilization of learned neural POMDP models, whose evaluation naturally maps to large batched tensor operations on modern GPUs. In addition, learning task-specific components such as value functions or heuristics can hinder generalization to unseen environments.

In this paper, we adopt this integrated planning-and-learning paradigm and propose an approximate POMDP solving framework, called *Vectorized Online planning wIth Learned diffusion model for POMDP Agents* (VOiLA). In contrast to existing work, VOiLA exploits the structure of POMDPs to learn reusable neural POMDP model components rather than task-specific policies, value functions, or heuristics. VOiLA integrates these model components with a massively parallel online POMDP planner to generate policies online. This decoupling of reusable POMDP model learning from policy generation significantly reduces the data requirements of VOiLA, while improving generalization to unseen environments.

Specifically, VOiLA learns neural parameterizations of the transition, observation, and likelihood models from offline simulation data and integrates them directly into online planning and belief estimation. To capture complex and potentially multimodal transition and observation functions, we first learn diffusion-based transition and observation models and subsequently distill them into efficient feedforward samplers suitable for online planning. A contrastive likelihood model is learned to support particle-based belief updates and handling continuous observations during planning.

To make this integration computationally practical, VOiLA uses a novel online POMDP solver called *Vectorized Online POMDP Planner* (VOPP). Similar to many online POMDP solvers, VOPP performs belief-space sampling to construct a representative belief tree and evaluate different action sequences. Unlike existing approaches, VOPP represents the entire belief tree as a collection of tensors and implements all planning operations as fully vectorized computations over this representation. The result is a fully GPU-native online POMDP solver that performs planning through tens of thousands of parallel simulations, enabling the learned neural POMDP models to be evaluated as large batched neural-network inference operations on the GPU.

We evaluate VOiLA in three settings. First, we compare VOPP against state-of-the-art online POMDP solvers on standard planning benchmarks and demonstrate that it frequently achieves comparable or superior performance using substantially smaller (up to $1000\times$) planning budgets. Second, we

evaluate VOiLA on simulated planning under uncertainty problems, including quadruped navigation tasks, and show that it achieves performance comparable to recurrent SAC while using only a fraction of the training data and exhibiting stronger generalization to previously unseen environments. Finally, we deploy VOiLA on a physical Unitree Go2 quadruped and demonstrate successful sim-to-real transfer, with POMDP models learned entirely in simulation enabling successful task completion in all real-world evaluation runs.

II. Background & Related Work

A. Partially Observable Markov Decision Process (POMDP)

A POMDP provides a general mathematical framework for sequential decision-making under uncertainty. Formally, a POMDP is an 8-tuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, b_0, \gamma \rangle$. Initially, the robot is in a hidden state $s_0 \in \mathcal{S}$. This uncertainty is represented by an initial belief $b_0 \in \mathcal{B}$, a probability distribution on the state space $\mathcal{S}$, where $\mathcal{B}$ is the set of all possible beliefs. At each step $t \geq 0$, the robot executes an action $a_t \in \mathcal{A}$ according to some policy π. Due to stochastic effects of executing actions, it transitions from the current state $s_t \in \mathcal{S}$ to a next state $s_{t+1} \in \mathcal{S}$ according to the transition model $T(s_t, a_t, s_{t+1}) = p(s_{t+1} \mid s_t, a_t)$, which is a conditional probability function. The robot does not know the state s_{t+1} exactly, but perceives an observation $o_t \in \mathcal{O}$ from the environment according to the observation model $Z(s_{t+1}, a_t, o_t) = p(o_t \mid s_{t+1}, a_t)$. In addition, the robot receives an immediate reward $r_t = R(s_t, a_t) \in \mathbb{R}$. The goal is to find a policy π that maximizes the expected total discounted reward or the *policy value*

$$\mathcal{V}_\pi(b_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, b_0, \pi\right], \tag{1}$$

where the discount factor $0 < \gamma < 1$ ensures that $\mathcal{V}_\pi(b)$ is finite and well-defined.

The robot's decision space is the set Π of policies, defined as mappings from beliefs to actions. In this paper, we consider *stochastic policies*, i.e., π is a belief-dependent distribution over the action space. The POMDP solution is then the optimal policy, denoted as π^* and given by

$$\pi^* = \arg\max_{\pi \in \Pi} \mathcal{V}_\pi(b), \tag{2}$$

for each belief $b \in \mathcal{B}$. A more elaborate explanation is available in [5].

B. Online Planning and Learning under Partial Observability

Online POMDP planning computes actions by performing forward search from the current belief during execution. Early methods such as POMCP [17] combine Monte Carlo Tree Search with particle-based belief representations to scale planning to large state spaces. Subsequent approaches including DESPOT [19], ABT [9], and HyP-DESPOT [1] improve planning efficiency through scenario-based search, belief-tree reuse, and parallel simulation. More recent extensions such as POMCPOW and PFT-DPW [18] extend online planning to continuous observation spaces through progressive widening. While these methods have demonstrated strong performance across a wide range of planning under uncertainty problems, they typically rely on hand-crafted POMDP models. Furthermore, most existing online POMDP solvers were developed around serial CPU execution and do not fully exploit the large-scale parallelism offered by modern GPUs.

An alternative is to learn environment models and use them for planning. Methods such as MuZero [16], Dreamer [3], and Visual Tree Search (VTS) [2] learn dynamics and observation models that support planning, policy optimization, or belief updates.

Recurrent reinforcement learning methods [4, 6, 12, 14] learn policies directly by augmenting them with memory modules, such as recurrent neural networks or transformers, allowing them to summarize action-observation histories without explicitly maintaining belief distributions.

Despite their success, these approaches are typically trained using task-specific objectives and environment interactions. As a result, the learned models or policies often become closely coupled to the tasks and environments used during training, which can limit generalization to unseen environments and require large amounts of task-specific training data.

Motivated by these limitations, we investigate whether task-agnostic POMDP models can be learned from offline simulation data and integrated into a scalable online POMDP planner. Efficiently exploiting such learned models requires evaluating transition, observation, and likelihood models thousands of times during planning, motivating the vectorized planning architecture introduced in the next section.

III. Method

We propose *Vectorized Online planning wIth Learned diffusion model for POMDP Agents* (VOiLA), a framework that combines massively parallel online POMDP planning with learned neural-network parameterized POMDP models. Since learned models achieve their highest computational efficiency when evaluated on large batches, exploiting them during online planning requires a planning architecture capable of performing thousands of model evaluations in parallel.

To address this challenge, we first introduce *Vectorized Online POMDP Planner* (VOPP), a fully GPU-native tree-search-based online POMDP solver that performs planning through thousands of parallel forward-simulations using batched tensor operations. Unlike traditional online POMDP solvers, which often rely on sequential tree expansions and synchronization between simulations, VOPP executes planning entirely through batched tensor computations. As a result, VOPP enables large-scale evaluation of learned neural POMDP models during online planning.

Building on this foundation, VOiLA learns task-agnostic neural POMDP models purely from offline-generated simulation data and integrates them directly into VOPP. In particular, VOiLA learns three model components: (a) a transition sampler $f_T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ that approximately samples successor states according to the transition function $T(s, a, s')$, (b) an

observation sampler $f_Z : \mathcal{S} \times \mathcal{A} \to \mathcal{O}$ that approximately samples observations according to $Z(s', a, o)$, where high-dimensional observations are modeled in a learned latent observation space, and (c) a likelihood model $\ell : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to \mathbb{R}$ that approximates the unnormalized observation density.

The learned transition, observation, and likelihood models are used both for online belief-space planning and particle-based belief updates. During planning, vectorized forward simulations are generated directly using the learned transition and observation samplers instead of hand-crafted generative models. Because VOPP already propagates thousands of simulations simultaneously, all model evaluations can be performed as large batched neural-network inference operations on the GPU. This enables learned neural POMDP models to be exploited at a scale that would be difficult to achieve with conventional online POMDP planning algorithms.

A. Vectorized Online POMDP Planner

VOPP is a massively parallel online POMDP solver that represents the entire planning process using batched tensor computations on the GPU. VOPP builds on the recently proposed Partially Observable Reference Policy Programming (PORPP) framework [7]. PORPP introduces an *analytical* value function, which is the original POMDP value function, penalized by the Kullback-Leibler (KL) divergence between the maximizing policy and a user-defined *reference policy* π_0. For a belief $b \in \mathcal{B}$, this value function can be compactly written as

$$\mathcal{V}(b) = \frac{1}{\eta} \log \Big[\sum_{a \in \mathcal{A}} \exp[\eta \Psi(b, a)] \Big] := [\mathcal{L}_\eta \Psi](b), \qquad (3)$$

where $\eta > 0$ is a temperature parameter, balancing reward maximization with deviation from the reference policy, and $\mathcal{L}_\eta$ is the log-sum-exp operator. The expression Ψ denotes *preferences* over belief-action pairs:

$$\Psi(b, a) = \frac{1}{\eta} \log(\pi_0(a \mid b)) + \mathcal{R}(b, a) + \gamma \sum_{o \in \mathcal{O}} p(o \mid b, a)[\mathcal{L}_\eta \Psi](\tau(b, a, o)), \qquad (4)$$

where $\mathcal{R}(b, a)$ is a Monte Carlo estimate of $\int_{s \in \mathcal{S}} R(s, a) b(s) \mathrm{d}s$ and τ is the belief update operator. Belief-action preferences induce a policy

$$\pi(b, a) = \frac{\exp[\eta \Psi(b, a)]}{\sum_{a' \in \mathcal{A}} \exp[\eta \Psi(b, a')]}. \qquad (5)$$

This reformulation eliminates much of the synchronization and coordination overhead typically associated with parallel tree search: During planning, actions are sampled from the preference-value induced policy, instead of selected, and eq. (3) can be estimated analytically, without having to maximize over the action space.

VOPP exploits these new avenues for massive parallelization. The key idea is to represent the belief tree as a collection of tensors, storing belief nodes, action nodes, and action preferences. Subsequently, VOPP performs all planning operations as batched tensor manipulations on this belief tree data structure. In particular, VOPP alternates between two fully vectorized operations during planning:

1) **Vectorized forward search:** Thousands of forward-simulations are executed in parallel by sampling actions from the current preference value induced policies to expand the belief tree and propagating them through a vectorized generative model.
2) **Vectorized preference backup:** The action preferences in eq. (4) and value functions in eq. (3) are updated simultaneously for all belief nodes at a given depth using batched tensor operations, starting from the leaf nodes.

To support continuous observation spaces, VOPP further incorporates a vectorized implementation of Progressive Widening (PW) [18]. PW incrementally limits the number of observation branches associated with each action node, allowing planning to remain tractable even when observations are continuous or high-dimensional. Importantly, PW is implemented using the same tensorized data structures and batched operations used throughout VOPP, thereby preserving the planner's massively parallel execution model.

Because all planning operations are implemented entirely as GPU tensor computations, VOPP can perform tens of thousands of parallel simulations without explicit synchronization between simulations.

B. Vectorized Online Planning with Learned POMDP Models

We now describe VOiLA and its integration of learned POMDP models into VOPP for online belief-space planning.

1) Offline Data Generation: Unlike many learning-based approaches that directly optimize task-specific policies, VOiLA learns POMDP models from offline-generated data. Training data is generated entirely in simulation by executing a uniform exploration policy and recording trajectories of states, actions, and observations.

Formally, we construct an offline dataset $\mathcal{D} = \{\xi^{(n)}\}_{n=1}^{N}$, where each trajectory $\xi^{(n)} = (s_i^{(n)}, a_i^{(n)}, o_i^{(n)})_{i=1}^{K}$ consists of a sequence of states, actions, and observations generated through interaction with the simulator. Here, N denotes the number of trajectories and K their maximum length.

Because the data collection policy is independent of the downstream task and reward structure, the resulting dataset captures environment dynamics and observation structure rather than task-specific behaviors. Task-specific behavior is subsequently generated through online planning. This helps with generalization to unseen environments and sim-to-real transfer.

2) Learning POMDP Models: Based on $\mathcal{D}$, VOiLA learns three reusable and efficient POMDP model components for online planning:

- **Transition sampler** $f_T(s, a, \omega_T)$, which approximates the transition distribution $T(s, a, s')$ and generates stochastic successor states, where $\omega_T \sim \mathcal{N}(0, I)$ is a latent noise variable used to model transition uncertainty.

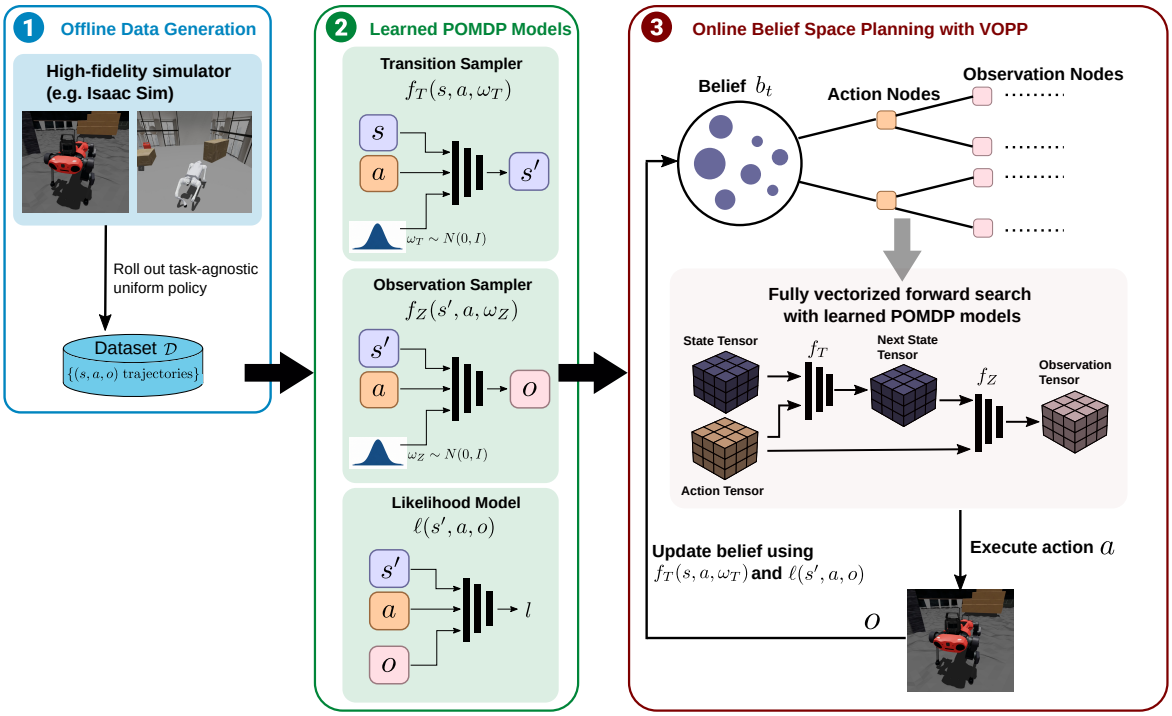


Fig. 1. Overview of VOiLA. Offline trajectories generated in a high-fidelity simulator using a task-agnostic policy are used to learn a transition sampler f_T, an observation sampler f_Z, and a likelihood model ℓ. During online planning, these models are integrated into the vectorized POMDP planner VOPP, which performs belief-space planning through thousands of parallel simulations and belief updates.

- **Observation sampler** $f_Z(s', a, \omega_Z)$, which approximates the observation distribution $Z(s', a, o)$ and generates observations conditioned on successor states and actions, where $\omega_Z \sim \mathcal{N}(0, I)$ is a latent noise variable capturing observation stochasticity.
- **Likelihood model** $\ell(s', a, o)$, which approximates the observation likelihood and is used for particle-based belief updates.

Real-world robotic systems often exhibit highly nonlinear and multimodal dynamics due to contact dynamics, actuation errors, and robot–environment interactions, while observations may consist of high-dimensional sensor inputs such as images or lidar scans. To capture such complex distributions, VOiLA first trains conditional denoising diffusion models on transition tuples $(s, a, s') \sim \mathcal{D}$ and observation tuples $(s', a, o) \sim \mathcal{D}$. Following recent advances in diffusion modeling, the diffusion models are trained using a v-prediction objective, which has been shown to provide stable training and high-quality generative samples. For high-dimensional observations such as images, observations are first mapped into a lower-dimensional latent space using an autoencoder, and diffusion models are learned in the resulting latent observation space.

While diffusion models can accurately model complex multimodal distributions, their iterative sampling procedure is computationally expensive and therefore unsuitable for integration with VOPP, where thousands of forward simulations are generated during planning. To address this challenge, VOiLA first constructs ODE-based diffusion samplers and subsequently distills them into compact feedforward generators. In particular, the transition and observation samplers f_T and f_Z are trained to imitate samples generated by the corresponding diffusion samplers, allowing them to directly generate transition and observation samples from states, actions, and latent noise variables. The resulting samplers retain much of the expressive power of the original diffusion models while reducing sampling cost by up to $283\times$, making them suitable for large-scale batched inference during online planning.

In addition, VOiLA learns an unnormalized likelihood model $\ell(s', a, o)$ using a contrastive objective on matching and mismatched state-action-observation tuples from $\mathcal{D}$. During online execution, exponentiated likelihood values are used as relative particle weights for particle-based belief updates.

3) Online Belief-Space Planning with VOiLA: The learned models described above are used directly during the vectorized forward-search procedure of VOPP. During vectorized forward search, VOPP propagates thousands of simulated trajectories in parallel. Instead of querying a hand-crafted generative model, VOiLA performs batched neural-network inference using the learned transition, observation, and likelihood models.

The learned likelihood model is used both during planning and belief updates. After executing an action and receiving an observation, beliefs are updated using a particle filter. The transition sampler generates proposal particles, while the likelihood model computes relative particle weights before resampling. Consequently, the same learned POMDP models support both online planning and belief estimation within a unified framework.

Because observations generated by the observation sampler may lie in continuous observation spaces, VOiLA uses the vectorized progressive widening mechanism provided by VOPP to control the growth of observation branches during planning.

IV. Experiments

We tested VOiLA on five planning under uncertainty problems detailed below.

A. Experimental Scenarios

Navigation in a partially known map (Navigation) [1]: This benchmark, shown in Figure 3(a), evaluates online planning under map uncertainty. A robot must navigate through a partially known 13×13 grid world to reach a goal region while avoiding obstacles. The state space consists of the robot position and occupancy of unknown cells, resulting in a state space of size $|\mathcal{S}| = 169 \times 2^{124}$. The robot receives noisy observations of neighboring cell occupancies ($|\mathcal{O}| = 8$) and can move to one of its eight neighboring cells or remain stationary ($|\mathcal{A}| = 9$). Reaching the goal yields a reward of 20, collisions incur a penalty of -1, standing still incurs a penalty of -0.2, and each step incurs a penalty of -0.1. The discount factor is $\gamma = 0.983$. Details can be found in [1].

Multi-Agent Rocksample (MARS) [1]: MARS(n, m), shown in Figure 3(b), is a cooperative multi-agent extension of the Rocksample benchmark. Two agents operate in an $n \times n$ environment containing m rocks that are either `GOOD` or `BAD`, resulting in a state space of size $|\mathcal{S}| = n^4 \times 2^m$. The agents receive noisy observations about rock quality ($|\mathcal{O}| = 9$) and may move or inspect rocks, resulting in an action space of size $|\mathcal{A}| = (5+m)^2$. Sampling a good rock yields a reward of 10, sampling a bad rock incurs a penalty of -10, and successfully exiting the map yields a reward of 10. The discount factor is $\gamma = 0.983$. Further details are provided in [1].

FloorPositioning [2]: This benchmark evaluates planning under state uncertainty in a small indoor environment. A robot operates in a 2.0×1.0m environment divided into two floors, each associated with a different goal region. The continuous state space is $\mathcal{S} = [0, 2] \times [0, 1]$. Observations consist of noisy distances to surrounding walls ($\mathcal{O} = \mathbb{R}^4$), and the robot can move in the eight compass directions or remain stationary ($|\mathcal{A}| = 9$). Reaching the correct goal yields a reward of 100, reaching the incorrect goal incurs a penalty of -100, and each non-terminal action incurs a penalty of -1. The discount factor is $\gamma = 0.99$. Additional details can be found in [2].

LeggedNavigation: This is a partially observable navigation task involving an ANYmal-C quadruped equipped with an onboard RGB camera operating in a cluttered 20×20m environment with rough terrain, obstacles, and two internal walls. Starting on the left-hand side of the scene, the robot must navigate to a goal region centered at $(7.5, -7.5)$m (green spheres in Figure 2). The two walls can occupy one of five predefined configurations shown in Figure 2, resulting in different traversable routes. Both the robot's initial position and wall configuration are unknown and must be inferred from RGB observations.

The state comprises the robot position, orientation (represented using a continuous 6D representation [20]), linear and angular velocities, joint positions and velocities, and the positions of the two internal walls, resulting in $\mathcal{S} = \mathbb{R}^{39} \times \mathtt{ENV}_1, \ldots, \mathtt{ENV}_5$. Observations are RGB images of resolution 32×32, yielding $\mathcal{O} = 0, \ldots, 255^{3\times32\times32}$. The action space consists of six discrete body-velocity commands (forward, slow forward, turn left/right, and forward arcs left/right), which are executed using a pre-trained locomotion controller.

The initial position is sampled uniformly from a 2.3×5.25m region on the left-hand side of the environment, while the wall configuration is sampled uniformly from the five layouts in Figure 2. The initial orientation and joint configuration are assumed known. The objective is to reach a circular goal region of radius 2m centered at $(7.5, -7.5)$m while avoiding collisions. Reaching the goal yields a reward of $+500$, collisions incur a penalty of -500, and each timestep incurs a penalty of -1. Episodes terminate upon reaching the goal or collision. The discount factor is $\gamma = 0.99$.

TargetFinding: The TargetFinding problem, shown in Figure 4, is a partially observable object-search task designed to evaluate sim-to-real transfer. A Unitree Go2 quadruped must locate and reach a target asset (a jerry can) using only lidar observations. The environment consists of a 4.9×3.6m room containing static obstacles and two cardboard boxes with uncertain positions. The target asset is hidden behind one of the boxes and can occupy one of two possible locations. As a result, the robot must actively gather information about both the asset and obstacle locations while planning a collision-free path to the target.

The state comprises the robot pose, velocities, joint states, asset position, and box positions, resulting in $\mathcal{S} = \mathbb{R}^{45}$. Observations consist of compact lidar scans represented by 36 range measurements, yielding $\mathcal{O} = \mathbb{R}^{36}$. The action space consists of six discrete body-velocity commands executed using a pre-trained locomotion controller.

The robot, asset, and box positions are initialized randomly at the start of each episode. Reaching the asset yields a reward of $+100$, collisions incur a penalty of -500, and each step incurs a penalty of -1. Episodes terminate when the robot reaches the asset or collides with an obstacle. The discount factor is $\gamma = 0.99$. Unlike LeggedNavigation, all POMDP models are trained entirely in simulation and deployed on the physical robot without additional real-world training data.

B. Experimental Setup

The purpose of our experiments is three-fold. First, we evaluate whether VOPP provides an effective and computationally efficient alternative to existing online POMDP solvers on problems with hard-coded POMDP models. The results of these experiments are presented in Section IV-C. Second, in Section IV-D we investigate whether the learned POMDP models of VOiLA support effective online belief-space planning and generalize to unseen environments. Third, we evaluate whether

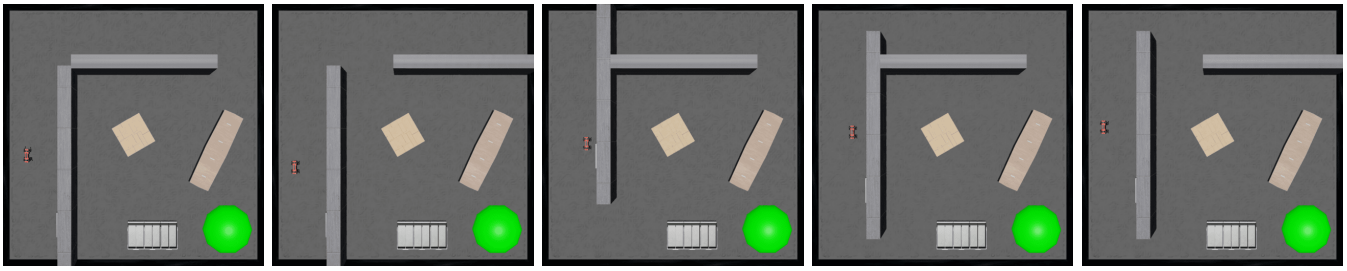


Fig. 2. Wall configurations used for the LEGGEDNAVIGATION environment. Different wall configurations induce different traversable routes, which must be inferred from onboard RGB observations. The green region represents the goal.

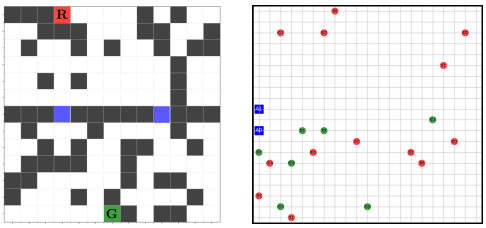
POMDP models learned entirely in simulation transfer to real-world robotic planning tasks, as evaluated in Section IV-E.

For the NAVIGATION and MARS benchmarks, VOPP is compared against HyP-DESPOT [1], DESPOT [19], and POMCP [17]. In these experiments, all solvers use the ground-truth hand-crafted POMDP models of the environments. For VOPP, the temperature parameter in eq. (3) was set to $\eta = 2.0$ for both problems, while the number of parallel simulations was set to $50,000$ and $60,000$ for NAVIGATION and MARS, respectively. HyP-DESPOT and DESPOT use the authors' implementation and recommended parameter settings, while POMCP parameters were tuned for each benchmark.

For the FLOORPOSITIONING, LEGGEDNAVIGATION, and TARGETFINDING problems, VOiLA learns transition, observation, and likelihood models from offline datasets generated using a random exploration policy. For each problem, an offline dataset consisting of 50K transitions is collected in simulation. In LEGGEDNAVIGATION, training data is collected only in environments 1 and 2, while all five environments are used during evaluation to assess generalization to unseen wall configurations. For LEGGEDNAVIGATION and TARGETFINDING, actions correspond to desired body velocities. A locomotion controller trained in Isaac Lab converts these commands to joint actions and is shared across all methods. VOiLA and VTS use 1s planning budgets (1.5s in TARGETFINDING), while recurrent SAC performs policy inference only.

For VOiLA, the transition & observation samplers and likelihood models are implemented as MLPs. In LEGGEDNAVIGATION, image observations are encoded into a shared 64-dimensional latent space used by VOiLA, VTS, and recurrent SAC. Recurrent SAC is trained on 750K environment interactions, whereas VOiLA and VTS are trained using 50K offline transitions.

VOiLA was implemented in Python using PyTorch [15]. For recurrent SAC and VTS, we use the official implementations provided by the authors. The VOPP experiments on NAVIGATION and MARS were conducted on a laptop with an Intel Core i7-13850HX CPU, 32GB RAM, and an NVIDIA RTX 3500 Ada Laptop GPU with 12GB VRAM. The VOiLA experiments on FLOORPOSITIONING, LEGGEDNAVIGATION, and TARGETFINDING were conducted on a laptop with an Intel Core Ultra 9 285HX CPU, 32GB RAM, and an NVIDIA RTX PRO 4000 Blackwell Laptop GPU with 16GB VRAM.



(a) Navigation (b) MARS

Fig. 3. The problem scenarios used to evaluate VOPP.

TABLE I
AVERAGE TOTAL DISCOUNTED REWARDS AND 95% CONFIDENCE INTERVALS OF ALL TESTED SOLVERS ON THE NAVIGATION AND MARS PROBLEMS. RESULTS ARE AVERAGED OVER 200 TRIALS PER SOLVER, PROBLEM, AND PLANNING TIME PER STEP.

	Planning time per step (s)			
Navigation	0.01	0.05	0.1	1.0
VOPP (Ours)	$\mathbf{8.8 \pm 1.0}$	$\mathbf{10.7 \pm 0.8}$	$\mathbf{11.7 \pm 0.7}$	$\mathbf{11.9 \pm 0.6}$
HyP-DESPOT	3.1 ± 1.4	5.2 ± 1.5	5.7 ± 1.6	9.3 ± 1.3
DESPOT	0.3 ± 1.9	0.7 ± 1.8	2.2 ± 1.8	8.7 ± 1.3
POMCP	-2.1 ± 0.9	-2.0 ± 0.8	-1.9 ± 1.1	-0.5 ± 1.0
MARS(20,20)	0.01	0.05	0.1	1.0
VOPP (Ours)	$\mathbf{31.1 \pm 2.6}$	$\mathbf{50.0 \pm 1.9}$	$\mathbf{53.3 \pm 2.1}$	$\mathbf{58.8 \pm 2.1}$
HyP-DESPOT	14.3 ± 1.5	19.2 ± 2.0	22.3 ± 2.4	47.9 ± 1.6
DESPOT	10.4 ± 1.5	17.8 ± 2.1	19.1 ± 2.3	20.9 ± 2.0
POMCP	-588.3 ± 73.1	-9.7 ± 3.5	-5.9 ± 3.0	6.7 ± 1.0

MARS(50,50) (3025 actions, 1.0s planning time/step):
VOPP achieved an average total discounted reward of $\mathbf{45.1 \pm 2.0}$ (200 trials).
HyP-DESPOT, DESPOT and POMCP crashed on MARS(50,50).

C. Evaluation of VOPP

Table I summarizes the average discounted rewards obtained by all tested solvers. Across both benchmark problems and all planning budgets, VOPP consistently achieves the highest performance. In the NAVIGATION problem, VOPP achieves an average discounted reward of 11.7 ± 0.7 using only 0.1s planning time per step, outperforming HyP-DESPOT, DESPOT, and POMCP even when they are given substantially larger planning budgets.

A similar trend is observed in MARS(20,20). For a planning budget of only 0.01s per step, VOPP achieves an average discounted reward of 31.1 ± 2.6, corresponding to approximately 64% of the reward achieved by HyP-DESPOT with a planning budget of 1s per step. Increasing the planning budget to 0.05s yields an average discounted reward of 50.0 ± 1.9, already exceeding the performance of HyP-DESPOT with 1s per step (47.9 ± 1.6). With a planning budget of 0.1s, VOPP further improves to 53.3 ± 2.1, substantially outperforming all

TABLE II
RESULTS ON THE LEGGEDNAVIGATION PROBLEM. ENVIRONMENTS 1 AND 2 WERE USED DURING TRAINING; ENVIRONMENTS 3–5 WERE UNSEEN DURING TRAINING.

Environment	Method	Success (%)	Avg. Reward
1	VOiLA	**92.0**	279.6±62.8
	Recurrent SAC	**92.0**	**342.7±36.9**
	VTS	32.5	-162.6±68.7
2	VOiLA	**92.0**	**315.3±65.4**
	Recurrent SAC	80.0	299.8±55.1
	VTS	36.0	-38.4±89.5
3	VOiLA	**74.0**	**196.8±102.8**
	Recurrent SAC	12.0	-22.0±33.0
	VTS	40.0	11.3±91.8
4	VOiLA	**80.0**	222.3±87.0
	Recurrent SAC	78.0	**232.4±78.1**
	VTS	16.0	-143.6±73.2
5	VOiLA	**72.0**	**161.9±100.9**
	Recurrent SAC	10.0	-102.0±58.2
	VTS	20.0	-161.5±82.2

TABLE III
TRAINING AND COMPUTATIONAL REQUIREMENTS FOR THE LEGGEDNAVIGATION PROBLEM.

Metric	VOiLA	Recurrent SAC	VTS
Num. transitions for training	**50K**	750K	**50K**
Data Collection Time (s)	**1647.6**	2760.8	**1647.6**
Training Time (s)	**6194.0**	8144.2	6922.0

competing methods. We additionally evaluated DESPOT and POMCP using a planning budget of 10s per step. DESPOT achieved an average discounted reward of 27.9 ± 4.8, while POMCP achieved 10.3 ± 4.4 (averaged over 20 trials). Thus, even with $1000\times$ larger planning budgets, both sequential solvers remained substantially less effective than VOPP with only 0.01s per step.

To further evaluate scalability, we additionally tested VOPP on MARS(50,50), a significantly larger variant with $3,025$ actions. With a planning budget of 1s per step, VOPP achieved an average discounted reward of 45.1 ± 2.0. In contrast, HyP-DESPOT, DESPOT, and POMCP were unable to solve this problem, as their implementations crashed for problem instances of this size. This result highlights an important advantage of VOPP: unlike many existing online POMDP solvers, VOPP does not require exhaustive action enumeration and therefore scales naturally to large action spaces.

These results indicate that the vectorized planning architecture of VOPP enables highly efficient online belief-space planning. In particular, VOPP frequently achieves comparable or superior performance using planning budgets that are one to three orders of magnitude smaller than those required by existing online POMDP solvers.

D. Learning Reusable POMDP Models

On FLOORPOSITIONING, VOiLA achieved a 100% success rate across 100 runs, matching recurrent SAC and outperforming VTS (91%), while using only 50K training transitions compared to 2.6M and 32M transitions for recurrent SAC and VTS, respectively.

The results on the more challenging LEGGEDNAVIGATION problem are shown in Tables II and III. During training, data was collected only in environment configurations 1 and 2. Nevertheless, VOiLA generalizes effectively to the unseen configurations 3–5. Across all five environments, VOiLA achieves success rates between 72% and 92%, consistently outperforming VTS and substantially outperforming recurrent SAC in several unseen environments, despite using substantially less training data.

The strongest evidence of generalization is observed in environments 3 and 5, where VOiLA substantially outperforms recurrent SAC despite being trained on the same environments 1 and 2. This suggests that the learned POMDP models capture aspects of the environment dynamics and observation process that transfer to unseen configurations. Additionally, on the LEGGEDNAVIGATION problem, the distilled transition and observation samplers achieved approximately 5.9×10^4 samples per second, compared to 2.1×10^2 and 7.2×10^2 samples per second for the diffusion models, achieving up to $283\times$ higher sampling throughput.

E. Results on TARGETFINDING (Sim-to-Real Transfer)

We deploy VOiLA on the TARGETFINDING problem using a Unitree Go2 quadruped operating in a real indoor environment. Across 10 real-world test runs, including five runs for each possible target location, VOiLA successfully completed the task in every trial. Figure 4 illustrates an example execution & the corresponding belief evolution. As observations are incorporated, the belief progressively concentrates around the true robot pose, obstacle locations, and target location.

To introduce variability, the positions of the cardboard obstacles were manually changed between runs. Despite this variation, the learned POMDP models transferred successfully from simulation to the physical robot without retraining. These results demonstrate that the combination of learned POMDP models and online belief-space planning can support robust closed-loop decision making on real robotic systems.

V. CONCLUSION

Learned neural POMDP models achieve their highest computational efficiency when evaluated on large batches, yet existing online POMDP planning algorithms are not naturally designed to exploit this capability. In this paper, we presented VOiLA, an integrated framework for decision making under uncertainty that combines learned task-agnostic POMDP models with massively parallel online planning. Central to this framework is VOPP, a fully GPU-native online POMDP solver that represents belief trees as tensors and performs planning through tens of thousands of parallel simulations, allowing learned transition, observation, and likelihood models to be evaluated efficiently as large batched neural-network inference operations. Experimental results show that our method achieves planning performance comparable to state-of-the-art online POMDP solvers while using up to $1000\times$

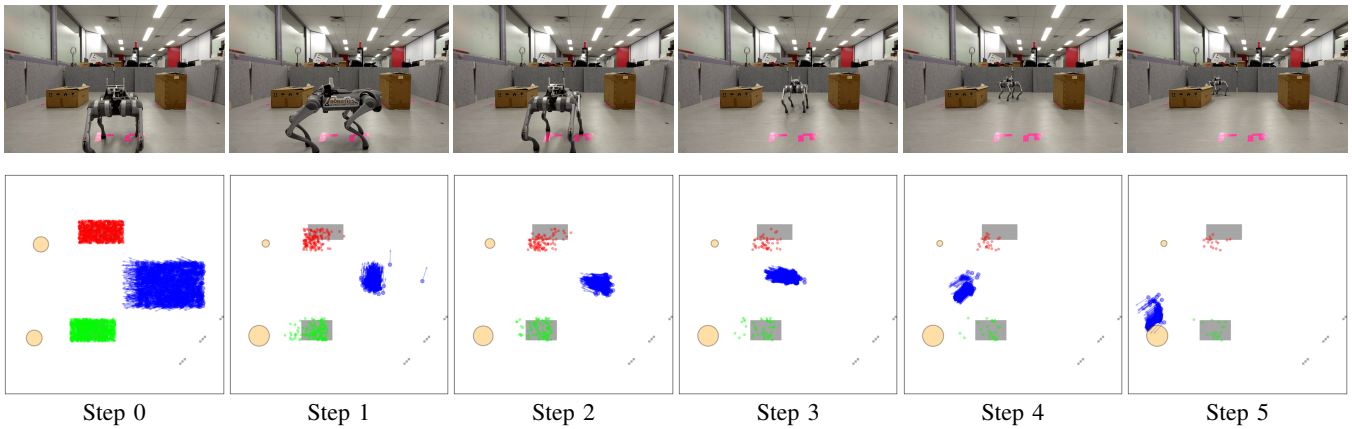


Fig. 4. Real-world execution of the Unitree Go2 quadruped and corresponding belief evolution. The top row shows images captured during execution. The bottom row shows beliefs over robot pose (blue particles), asset location (orange circles), and movable obstacle locations (green and red particles). As observations are incorporated, the belief progressively concentrates around the true environment state.

smaller planning budgets. Furthermore, the results demonstrate that combining vectorized online planning with learned task-agnostic POMDP models enables data-efficient learning, strong generalization to unseen environments, and successful sim-to-real transfer without additional training.

Acknowledgments

This work was supported by the ARC Research Hub in Intelligent Robotic Systems for Real-Time Asset Management (IH210100030), in collaboration with the University of Sydney and Nexxis Technology.

References

[1] Panpan Cai, Yuanfu Luo, David Hsu, and Wee Sun Lee. Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty. *IJRR*, 40(2-3):558–573, 2021.

[2] Sampada Deglurkar, Michael H Lim, Johnathan Tucker, Zachary N Sunberg, Aleksandra Faust, and Claire Tomlin. Compositional learning-based planning for vision POMDPs. In *L4DC*, pages 469–482. PMLR, 2023.

[3] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

[4] Matthew J Hausknecht and Peter Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI fall symposia*, volume 45, page 141, 2015.

[5] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[6] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *ICLR*, 2018.

[7] Edward Kim, Yohan Karunanayake, and Hanna Kurniawati. Reference-based POMDPs. *NeurIPS*, 36:40659–40675, 2023.

[8] H. Kurniawati. Partially Observable Markov Decision Processes and Robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1), 2022.

[9] Hanna Kurniawati and Vinay Yadav. An online POMDP solver for uncertainty planning in dynamic environment. In *ISRR*, pages 611–629. Springer, 2016.

[10] Yiyuan Lee, Panpan Cai, and David Hsu. Magic: Learning macro-actions for online POMDP planning. *arXiv preprint arXiv:2011.03813*, 2020.

[11] Robert J. Moss, Anthony Corso, Jef Caers, and Mykel J. Kochenderfer. BetaZero: Belief-State Planning for Long-Horizon POMDPs using Learned Approximations. In *RLC*, 2024.

[12] Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free RL can be a strong baseline for many POMDPs. In *ICML*, pages 16691–16723, 2022.

[13] Christos H Papadimitriou and John N Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[14] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *ICML*, pages 7487–7498, 2020.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019.

[16] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[17] David Silver and Joel Veness. Monte-carlo planning in large pomdps. *NeurIPS*, 23, 2010.

[18] Zachary Sunberg and Mykel Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *ICAPS*, pages 259–263, 2018.

[19] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee.

DESPOT: Online POMDP planning with regularization. *JAIR*, 58:231–266, 2017.

[20] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, pages 5745–5753, 2019.