VOiLA: Vectorized Online Planning with Learned Diffusion Models for POMDP Agents

Marcus Hoerger[1], Rishikesh Joshi[1], Rahul Shome[1], Ian Manchester[2], and Hanna Kurniawati[1]

[1] Australian National University, Canberra, ACT, Australia,
{marcus.hoerger, rishikesh.joshi, rahul.shome,
hanna.kurniawati}@anu.edu.au,
[2] The University of Sydney, Sydney, NSW, Australia.
ian.manchester@sydney.edu.au

Abstract. Planning under uncertainty is an essential capability for autonomous robots. The Partially Observable Markov Decision Process (POMDP) provides a powerful framework for such a capability. Although POMDP-based planning has advanced significantly, its application to real-world problems is often limited by the difficulty of obtaining faithful POMDP models. We present *Vectorized Online planning wIth Learned diffusion model for POMDP Agents* (VOiLA), a framework that learns task-agnostic POMDP world models for online planning under uncertainty. Specifically, VOiLA learns transition and observation samplers using conditional diffusion models and observation-likelihood models for particle-based belief updates. To enable efficient online planning, the diffusion samplers are distilled into compact feedforward generators and integrated with Vectorized Online POMDP Planner (VOPP), an online POMDP planner designed to leverage GPU parallelization. Experimental results indicate the distillation strategy reduces sampling cost by up to nearly three orders of magnitude, making learned generative POMDP models practical for online planning. Evaluation of VOiLA on three benchmark problems indicates that VOiLA achieves equal or better performance than current reinforcement learning (RL) methods for POMDPs, such as Recurrent Soft Actor Critic and DreamerV3, while using less than 10% of the training data, and generalizes much better to unseen environment configurations. In a physical robot experiment, VOiLA accomplishes the task in 10 of 10 runs while using models learned purely from simulated data for online planning.

Keywords: Planning under Uncertainty, Integrated Planning and Learning, POMDPs, Partial Observability, Generalizable Learning, Sim2Real, World Models

1 Introduction

In the real world, robots must operate reliably despite imprecise actuators, noisy sensors, perception errors, and incomplete knowledge of their surroundings. Partially Observable Markov Decision Processes (POMDPs) provide a principled and unified framework for reasoning under uncertainty, 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 [21], substantial progress has been made in computing approximate solutions. These methods have largely evolved along two separate but complementary directions. The first is approximate POMDP solving [12], which today is often performed online and therefore avoids substantial offline pre-computation, but requires POMDP models to be available a priori. The second is reinforcement-learning-based approaches, starting with [6], which relax the need for explicit models but generally require large amounts of data and significant training time.

Methods have been proposed to combine the strengths of planning and learning. Some [4, 16, 18, 25] use search tree structures typical of planning and integrate them with learning of certain components of the POMDP models, heuristics, or estimates of the value functions. However, the planning components in these combinations are single-threaded. We adopt this integrated planning-and-learning paradigm, but build on recent learning techniques and a recent online POMDP planner compatible with modern learning platforms.

Specifically, we propose an approximate POMDP solving framework *Vectorized Online planning wIth Learned diffusion model for POMDP Agents* (VOiLA). In POMDP planning, the required world models include the transition models, which capture the uncertainty of action outcomes as a conditional probability distribution function over possible next states, and observation models, which capture the likelihood of observations under different pairs of states and actions. VOiLA models the POMDP transition and observation functions using expressive conditional diffusion models. To enable fast sampling, these models are distilled into compact feedforward generators. In addition, VOiLA learns a contrastive likelihood model for belief updates and online planning. All models are learned offline purely from simulated data.

Once learned, the models are used by VOPP [8] to compute near-optimal POMDP policies online. VOPP is a recent online POMDP planner that vectorizes planning computations to exploit the massive parallelism of GPUs. For VOiLA, we extend VOPP to handle continuous observation spaces. By learning only the transition and observation models and then using them for planning, VOiLA can significantly reduce data requirements while improving generalization across tasks and scenarios.

We evaluate VOiLA on three domains: a simple POMDP benchmark introduced in [4], a simulated quadruped locomotion task in uncertain and previously unseen environments, and a physical robot experiment on target finding in an uncertain environment using a quadruped. The simulation results indicate that VOiLA achieves comparable performance to current RL-based POMDP methods, DreamerV3 [5] and Recurrent Soft Actor Critic for Discrete Actions [19], in trained scenarios while using only 10% of their training data, and exhibits stronger generalization to unseen scenarios. In a physical target finding problem involving a quadruped robot, VOiLA uses models learned purely from simulated data for online planning and successfully accomplishes the task in 10 of 10 runs. This indicates the potential that models learned solely from simulation data

can be directly used to generate good real-world strategies under the POMDP framework.

2 Background & Related Work

2.1 Partially Observable Markov Decision Process (POMDPs)

A POMDP problem is specified as a 7-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \gamma \rangle$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the state, action, and observation spaces. In this paper, $\mathcal{S}$ and $\mathcal{O}$ may be discrete, continuous, or hybrid, while $\mathcal{A}$ is discrete; for simplicity, we describe the background assuming all spaces are discrete. The transition function T is a conditional distribution, with $T(s, a, s') = P(s'|s, a)$ denoting the probability of next state s' after performing action a in state s. The observation function Z is defined by $Z(s', a, o) = P(o|s', a)$, and the reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is bounded, and $\gamma \in (0, 1)$ is a discount factor.

Since the agent does not directly observe the true state, it maintains a belief $b \in \mathcal{B}$, i.e., a probability distribution over states, where $\mathcal{B}$ is the set of all possible beliefs. In almost all scalable approximate POMDP solvers, the agent starts from a given initial belief $b_0 \in \mathcal{B}$. Given belief b, action a, and observation o, the belief is updated according to $b'(s') = \tau(b, a, o) \propto Z(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s')b(s)$, where $\tau(\cdot)$ is the belief update operator. The expected immediate reward is $R(b, a) = \sum_{s \in \mathcal{S}} R(s, a)b(s)$.

The actions a POMDP agent takes at each time step is governed by a policy $\pi : \mathcal{B} \to \mathcal{A}$, which maps beliefs to actions. In this paper, we assume a deterministic policy, where a belief is mapped to an action, rather than a distribution over the action space. Since beliefs are sufficient statistics of action-observation histories, this is equivalent to mapping histories to actions. Each policy π has value $V^\pi : \mathcal{B} \to \mathbb{R}$, defined as the expected discounted return $V^\pi(b) := E[\sum_{t=0}^{\infty} \gamma^t R(b, a)]$, where the expectation is taken over trajectories induced by the policy, transition function, and observation function. The POMDP solution is an optimal policy π^* with value $V^{\pi^*}(b) = \sup_{\forall \pi} V^\pi(b)$ for all $b \in \mathcal{B}$.

2.2 Online POMDP Planning

Online POMDP planning interleave policy computation and policy execution by maintaining a belief and performing forward search from the current belief. Earlier approaches such as POMCP [26] use Monte Carlo Tree Search combined with particle beliefs to scale to large state spaces. Subsequent methods, including DESPOT [30] and ABT [13], improve efficiency through scenario-based planning and belief-tree reuse. More recent extensions such as POMCPOW and PFT-DPW [28] address continuous observation spaces using progressive widening. While these methods have demonstrated strong planning performance, they were primarily designed around serial CPU-based simulation and do not fully exploit modern parallel hardware such as GPUs. Although learned transition and observation models can be integrated into these solvers, their architectures do not efficiently leverage batched neural-network inference during online planning.

Vectorized Online POMDP Planner (VOPP) [8] is a recently proposed massively parallel online method based on the Partially Observable Reference Policy Programming (PORPP) framework [11], which partially solves the POMDP value function analytically, leaving the numerical computation to be only estimation of expected total reward: $V(b) = \frac{1}{\eta} \log \left(\sum_{a \in \mathcal{A}} \exp(\eta \Psi(b, a)) \right) := [\mathcal{L}_\eta \Psi](b)$, 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 [2]. The expression Ψ denotes *preferences* over belief-action pairs

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

where $R(b, a)$ is a Monte Carlo estimate of $\int_{s \in \mathcal{S}} R(s, a) b(s) \mathrm{d}s$. By removing the need to interleave numerical optimisation and estimation of expectation, the scheduling to parallelise POMDP solving can be substantially reduced.

Building on this formulation, VOPP represents the entire belief tree using a collection of tensors storing belief nodes, action nodes, and action preferences. It incrementally constructs the belief tree by interleaving fully vectorized forward search using tens of thousands of parallel simulations with fully vectorized preference backup operations. Both operations are implemented entirely as tensor computations running on the GPU, enabling VOPP to achieve state-of-the-art performance in online POMDP planning.

2.3 Learning-Based Approaches in POMDPs

Learning approaches have been proposed to compute POMDP policies. One typical approach is Recurrent model-free Reinforcement Learning. They augment policies with memory mechanisms such as recurrent neural networks [6, 9, 19] or transformers [22], allowing them to summarize action-observation histories without explicitly maintaining a belief state. Model-based Reinforcement Learning methods [15] and end-to-end learning [3, 10] for approximate POMDP solving have been proposed too. While effective on the training distribution, such approaches typically require large amounts of task-specific training data.

An alternative is to learn models for planning. Methods such as MuZero [25], Dreamer [5], and Visual Tree Search (VTS) [4] learn dynamics and/or observation models that are used for planning, policy optimization, or belief updates. However, these methods primarily target task-specific learning and are therefore tightly coupled to the training objectives and environments used during learning.

In contrast, VOiLA learns task-agnostic transition and observation models designed specifically for integration with a scalable online POMDP planner. The learned models can therefore be reused across tasks sharing the same environment dynamics while supporting both online planning and particle-based belief updates. The highly scalable POMDP planner of VOiLA enables good policies to be inferred despite model errors, further reducing the data requirements.

3 Overview of VOiLA

We propose VOiLA, a sequential decision-making framework that combines the recently proposed massively parallel online POMDP planning VOPP with

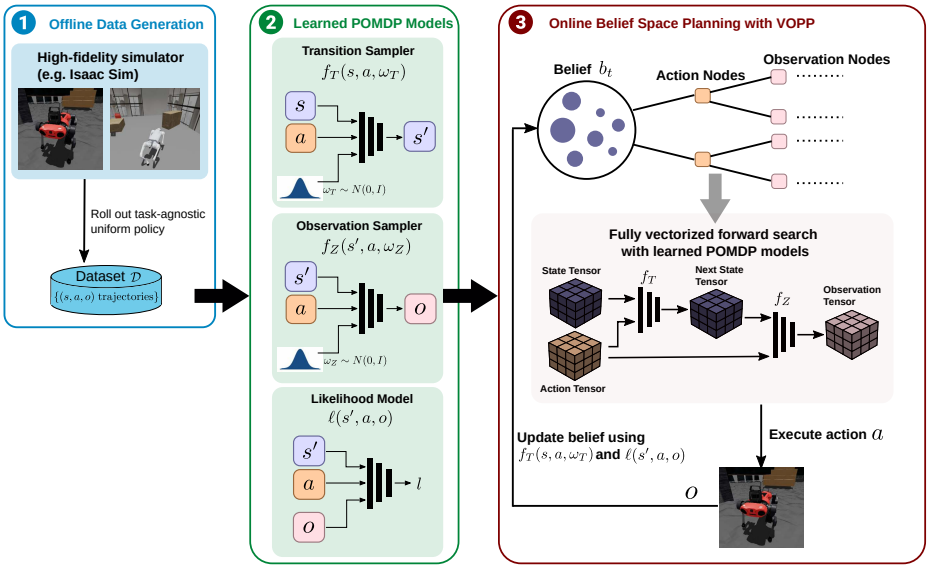


Fig. 1. Overview of VOiLA. Offline trajectories generated in a high-fidelity simulator using a task-agnostic policy are used to learn a transition sampler f_T, an observation sampler f_Z, and a likelihood model ℓ. During online planning, these models are integrated into the vectorized POMDP planner VOPP,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

learned neural-network parameterized POMDP models for scalable planning under uncertainty. VOiLA learns task-agnostic POMDP models purely from offline-generated simulation data and uses them with VOPP to enable efficient online belief-space planning. An overview of VOiLA is shown in Figure 1.

In particular, our method learns three POMDP model components: (a) a transition sampler $f_T : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ to approximately sample next states according to the transition function $T(s, a, s')$, (b) an observation sampler $f_Z : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{O}$ 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 $\ell : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \mapsto \mathbb{R}$ which approximates the (unnormalized) POMDP observation density Z.

A core requirement of the learned transition and observation samplers is that they must be sufficiently expressive to capture complex stochastic robot dynamics (e.g., legged locomotion) and high-dimensional observation distributions (e.g., images), while remaining efficient enough to serve as generative models during online POMDP planning. We achieve this by training both the transition and observation samplers in two stages. First, expressive conditional diffusion models are trained to model the underlying transition and observation distributions. Second, the diffusion samplers are distilled into compact feedforward generators that closely approximate the original diffusion models while reducing sampling cost by up to nearly three orders of magnitude. Details on our model learning approach are provided in Section 4.2.

The learned transition, observation, and likelihood models are integrated into VOPP for both online belief-space planning and particle-based belief updates. During planning, vectorized forward simulations are generated directly using the offline learned transition and observation samplers instead of relying on handcrafted models. To support continuous observation spaces, we additionally extend VOPP with a fully vectorized implementation of progressive widening [28].

After computing an action for the current belief, VOiLA executes the action in the environment and updates the belief based on the perceived observation. Beliefs are represented as sets of particles and updated using a Sequential Importance Resampling (SIR) particle filter [1]. The transition sampler f_T is used to sample proposal particles, while the likelihood model ℓ is used to compute particle weights from observations before resampling. Details on how the learned models are integrated into VOPP are provided in Section 4.3.

4 Details of VOiLA

4.1 Offline Data Generation

Training data for the learned POMDP models is generated entirely offline using high-fidelity robot simulators such as Isaac Sim [17]. To generate training data, we execute a task-independent data collection policy (a uniform policy in all experiments) in simulated environments and record trajectories consisting of POMDP 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 general dynamics and observation structure rather than task-specific behaviors. The dataset $\mathcal{D}$ is then used to train the transition sampler, observation sampler, and likelihood model.

4.2 POMDP Model Learning Approach

Learning POMDP models for real-world robotic planning under uncertainty is challenging due to complex stochastic dynamics, high-dimensional observations, and the computational requirements of online belief-space planning. The learned models must not only accurately represent the underlying transition and observation distributions, but also support efficient sampling during online planning.

VOiLA learns three efficient model components for online planning: a transition sampler f_T that approximately samples successor states according to the transition distribution $T(s, a, s')$, an observation sampler f_Z that approximately samples observations according to the observation distribution $Z(s', a, o)$, and a likelihood model $\ell(s', a, o)$ that approximates the (unnormalized) observation density Z. Details on how these model components are learned, including training objectives and derivations, are presented in the following subsections.

Transition Sampler The transition sampler is modeled as

$$s' = f_T(s, a, \omega_T), \tag{2}$$

where $\omega_T \sim \mathcal{N}(0, I)$ is a latent noise variable that captures transition stochasticity. Real-world transition dynamics are often highly nonlinear and multimodal due to contact dynamics, actuation errors, and robot–environment interactions. Capturing such dynamics therefore requires expressive generative models.

Motivated by their strong performance in modeling complex multimodal distributions, we first train a conditional denoising diffusion model [7] on transition tuples $(s, a, s') \sim \mathcal{D}$ using a v-prediction objective [24]. In particular, we construct a noise-corrupted next state $x_\tau = \sqrt{\bar{\alpha}_\tau} s' + \sqrt{1 - \bar{\alpha}_\tau}\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, $\tau \in (0, 1)$ denotes the diffusion time, and $\bar{\alpha}_\tau \in (0, 1)$ is the cumulative noise schedule (we use a cosine schedule) evaluated at τ. A conditional denoiser $v_T(s, a, x_\tau, \tau)$ is then trained to predict the v-target $v = \sqrt{\bar{\alpha}_\tau}\epsilon - \sqrt{1 - \bar{\alpha}_\tau} s'$.

While diffusion models are highly expressive, their iterative sampling process is computationally too expensive for online POMDP planning, where a large number of forward simulations must be generated.

To enable efficient online planning, we distill an ODE-based diffusion sampler [27] into the compact model f_T, which directly maps (s, a, ω_T) to a next-state sample s', such that s' is approximately distributed according to $T(s, a, s')$. We train f_T to imitate the next-state samples generated by the ODE-based diffusion sampler. Specifically, we minimize the distillation loss

$$\mathcal{L}_{\text{dist}} = \mathbb{E}_{(s,a,s') \sim \mathcal{D},\, \omega_T \sim \mathcal{N}(0,I)} \left[\left\| \hat{s'} - f_T(s, a, \omega_T) \right\|_2^2 \right], \tag{3}$$

where $\hat{s'}$ denotes the next-state sample generated by the ODE-based diffusion sampler using (s, a, ω_T) and denoiser v_T.

The transition sampler f_T closely approximates the ODE-based sampler, while reducing sampling cost by up to nearly three orders of magnitude.

Observation Sampler The observation sampler f_Z is learned analogously to the transition sampler described in the previous paragraph. In particular, f_Z is a compact distilled sampler that samples observations conditioned on (s', a) for efficient use during online planning. For low-dimensional observations, the observation sampler generates observations in the original observation space $\mathcal{O}$ directly. In this case, the same diffusion and distillation learning procedure described for the transition sampler is used to train f_Z.

However, for high-dimensional observations such as images, sampling observations from the original observation space $\mathcal{O}$ can be computationally inefficient during planning. In such cases, we instead introduce a lower-dimensional latent observation space $\mathcal{Z}$ with $\dim(\mathcal{Z}) \ll \dim(\mathcal{O})$ using a conditional observation autoencoder consisting of an encoder E and decoder D, such that $z = E(s', a, o)$ and $o = D(s', a, z)$. The encoder and decoder are trained jointly on tuples $(s', a, o) \sim \mathcal{D}$ by minimizing the reconstruction objective

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{(s',a,o) \sim \mathcal{D}} \left[\| o - D(s', a, E(s', a, o)) \|_2^2 \right]. \tag{4}$$

Different encoder and decoder architectures can be used depending on the observation modality, including vector-valued, image-based, or mixed observations. Once trained, the autoencoder weights are frozen and the encoder E is used to map observations to the latent observation space $\mathcal{Z}$.

We then train a conditional diffusion model in the latent observation space using tuples (s', a, z), where $z = E(s', a, o)$. Following the same diffusion and distillation procedure used for the transition sampler, we finally distill the diffusion sampler into the compact feedforward observation sampler

$$z = f_Z(s', a, \omega_Z), \tag{5}$$

where $\omega_Z \sim \mathcal{N}(0, I)$ is latent noise. This enables efficient latent observation sampling during online planning.

Likelihood Model Online belief updates and planning with continuous observation spaces require evaluating the observation density $Z(s', a, o)$ for candidate next states. However, learning normalized observation densities is challenging in practice due to the intractability of the normalization constants [14]. Fortunately, particle-based belief updates only require relative particle weights, making unnormalized observation densities sufficient for online POMDP planning.

We therefore learn an unnormalized likelihood model $\ell(s', a, o)$ whose exponentiated outputs approximate the observation density $Z(s', a, o)$ up to a normalizing constant. The model ℓ is trained contrastively using an InfoNCE-style objective [20]: for a mini-batch of size B, i.e., $\{(s'_i, a_i, o_i)\}_{i=1}^{B} \sim \mathcal{D}$, the model is evaluated on all $B \times B$ state-action–observation combinations, where matching tuples (s'_i, a_i, o_i) are treated as positive pairs and mismatched tuples (s'_i, a_i, o_j) with $i \neq j$ serve as in-batch negatives. We then minimize the InfoNCE loss

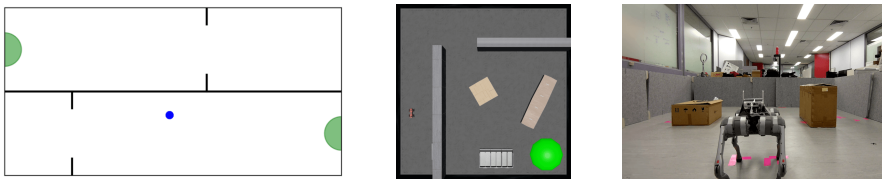
$$\mathcal{L}_{\text{likelihood}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\ell(s'_i, a_i, o_i))}{\sum_{j=1}^{B} \exp(\ell(s'_i, a_i, o_j))}. \tag{6}$$

This objective encourages the likelihood model to assign higher values to matching (s', a, o) tuples from the dataset than to mismatched tuples.

The resulting model ℓ does not represent a normalized observation density. Instead, exponentiated likelihood values $\exp(\ell(s', a, o)/\lambda)$ are used as relative particle weights during belief updates, which is sufficient for particle-based online POMDP planning. The parameter $\lambda > 0$ is a problem-dependent temperature parameter which controls the sharpness of the resulting particle weight distribution, with smaller values producing more concentrated particle weights and larger values yielding more uniform particle weights, thereby mitigating particle impoverishment during belief updates. In our experiments in Section 5, we use $\lambda \in \{1.0, 1.25\}$, which we found to perform well.

4.3 Online belief-space planning with VOiLA

During online planning, VOiLA follows VOPP's vectorized belief-space planning procedure. In the original VOPP algorithm, the belief tree $\mathcal{T}$ is expanded using



(a) FloorPositioning (b) LeggedNavigation (c) TargetFinding

Fig. 2. Problem scenarios used to evaluate VOiLA.

a hand-crafted vectorized generative model $G(s, a)$ that samples successor states and observations during forward search. In VOiLA, we instead replace this generative model with the learned transition and observation samplers introduced in Section 4.2. Given batched states and actions, the transition sampler $f_T(s, a)$ generates successor states s', while the observation sampler $f_Z(s', a)$ samples corresponding observations o.

To handle continuous observation spaces, we extend VOPP with a vectorized implementation of Progressive Widening (PW) [28]. PW incrementally limits the number of observation branches associated with each action node in $\mathcal{T}$. In particular, an action node a is expanded with a sampled observation whenever $|C(a)| < k_o N(a)^{\alpha_o}$, where $|C(a)|$ denotes the number of child observation branches associated with action node a, $N(a)$ is the action-node visitation count, and $k_o, \alpha_o > 0$ are widening hyperparameters. When the widening criterion is satisfied at action node a, a newly sampled observation is added as a new observation branch by extending the observation nodes tensor data structure. Otherwise, the sampled successor state is associated with a previously existing observation branch of the same action node. The likelihood model $\ell(s', a, o)$ is then used to weight successor states according to the observation stored at the selected branch, followed by resampling.

Importantly, parallel simulations are grouped by their parent action nodes, after which the above PW operations are performed for all action-node groups in parallel. This preserves VOPP's massively parallel planning formulation, even in continuous observation spaces.

5 Experiments & Results

We tested VOiLA on three planning under uncertainty problems, detailed below.

5.1 Problem Scenarios

FloorPositioning is a simple POMDP benchmark introduced in [4]. A robot operates in a rectangular environment with internal walls dividing it into upper and lower floors, each with a different goal. The continuous state is the robot position, and the action space comprises motion by 0.05m in eight compass directions and a `STAY` action ($|\mathcal{A}| = 9$). Actions are deterministic, with collisions leaving the robot stationary. Observations are four-dimensional vectors ($\mathcal{O} = \mathbb{R}^4$) of distances to the nearest wall or boundary in the cardinal directions, corrupted by zero-mean Gaussian noise with standard deviation 0.01.

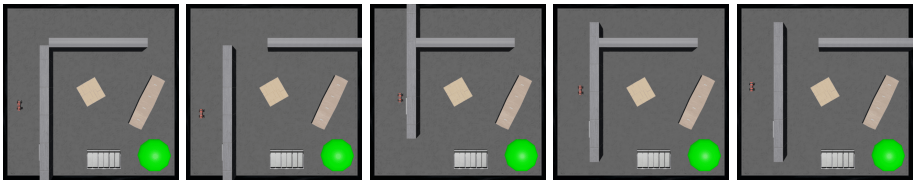


Fig. 3. Wall configurations used for the LeggedNavigation environment. Different wall configurations induce different traversable routes, which must be inferred from onboard RGB observations. The green region represents the goal.

The initial position is sampled such that the robot is uncertain about which floor it occupies. The correct and incorrect goals yield rewards of 100 and -100, respectively, while each non-terminal action incurs -1. Episodes terminate upon reaching the correct goal, with discount factor 0.99.

LeggedNavigation is a partially observable navigation task, where an ANYmal-C quadruped with an onboard RGB camera operates 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 sphere in Figure 2(b)). The two walls can occupy one of five predefined configurations shown in Figure 3, 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 [31]), linear and angular velocities, joint positions and velocities, and the positions of the two internal walls, resulting in $\mathcal{S} = \mathbb{R}^{39} \times \texttt{ENV}_1, \ldots, \texttt{ENV}_5$. 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 3. 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.

TargetFinding TargetFinding is a partially observable object-search task evaluating whether POMDP models learned entirely in simulation transfer to physical robot planning. A Unitree Go2 quadruped must locate and reach a target asset (a jerry can) using lidar observations in a 4.9×3.6m room containing static obstacles and two cardboard boxes with uncertain positions. The asset is hidden behind one of the boxes, requiring the robot to gather information about the asset and box locations while navigating collision-free.

The state comprises the robot pose, linear and angular velocities, joint positions and velocities, and the asset and box positions, yielding $\mathcal{S} = \mathbb{R}^{45}$. Lidar

returns are transformed into the robot frame, restricted to a front-facing field of view, and discretized into 36 azimuth bins containing the minimum range per sector, normalized by the 2.0m sensor range and clipped to $[0, 1]$, yielding $\mathcal{O} = \mathbb{R}^{36}$. The same preprocessing is used in simulation and on the physical robot. The action space consists of six discrete body-velocity commands (forward, slow forward, turn left/right, and forward arcs left/right) executed by a pre-trained locomotion controller.

The initial robot position is sampled uniformly from a 1.5×1.0m region and its heading within $\pm\pi/8$rad of the nominal orientation. The asset location is sampled uniformly from two candidate locations behind the boxes, while box positions vary by up to ± 0.5m longitudinally and ± 0.25m laterally around their nominal positions. Reaching the asset yields a reward of $+100$, collisions yield a penalty of -500, and each step incurs a cost of -1; episodes terminate upon reaching the asset or colliding with a box or environment boundary. All models are trained entirely in simulation and deployed on the physical robot without additional real-world training data.

5.2 Experimental Setup

The purpose of our experiments is two-fold. First, we evaluate whether VOiLA's learned POMDP models support effective online belief-space planning and generalize to unseen environment configurations. We evaluate this in FloorPositioning and LeggedNavigation, where we compare VOiLA against VTS [4], DreamerV3 [5], and RSAC-D [19]. Second, we evaluate whether POMDP models learned from simulation data transfer zero-shot to physical robot planning problems. We investigate this in TargetFinding, where the models learned in simulation are deployed directly for online planning on a physical robot.

For each problem, VOiLA learns its task-agnostic POMDP models from 50K transitions collected offline in simulation using a uniform random policy. In LeggedNavigation, training data is collected only in environments 1 and 2 (Figure 3) for all methods, while evaluation is performed in all five environments to assess generalization to unseen configurations.

We allocate each method approximately the same total wall-clock offline training budget as VOiLA, measured as data collection time plus model or policy training time. Since data collection and training throughput differ between methods and problems, this budget corresponds to different numbers of environment transitions being used by each method. In FloorPositioning, DreamerV3, RSAC-D and VTS use 900K, 1.8M and 32M transitions, respectively; in LeggedNavigation, DreamerV3 and RSAC-D use 600K transitions, and VTS uses 50K transitions. We additionally evaluate DreamerV3 and RSAC-D using 2.6M transitions in both problems, by which point both methods have approximately converged. These variants serve as ablations that assess their performance with substantially more data and computation. Since the official VTS implementation does not support learning transition models, VTS uses the transition sampler learned by VOiLA.

For LeggedNavigation, all methods share the same pre-trained locomotion controller, which converts the six high-level body-velocity actions into low-level

joint commands. Each high-level action is executed for 2s. During execution, VOiLA and VTS use 1s of online planning per step in FloorPositioning and LeggedNavigation; VOiLA uses 1.5s in TargetFinding. DreamerV3 and RSAC-D instead execute their learned policies directly, taking 0.01s/step. We use the R2-Dreamer implementation of DreamerV3[3] and the official implementations of RSAC-D for POMDPs[4] and VTS[5]. VOiLA is implemented in PyTorch [23]. All experiments are carried out on the same machine[6].

5.3 Experimental Results

Table 1. Results on the FloorPositioning problem, averaged over 100 trials. The best results are highlighted in **bold**.

Metric	VOiLA	DreamerV3	RSAC-D	VTS
Success rate (%) ↑	**100**	**100**	**100**	88
Avg. total disc. reward ↑	60.3 ± 1.7	60.7 ± 1.6	$\mathbf{64.4 \pm 1.1}$	23.5 ± 11.9
Training transitions ↓	**50K**	900K	1.8M	32M
Data collection time (s) ↓	**0.71**	19	58	265
Model/policy training time (s) ↓	1,665	1,673	1,623	**1,377**
Total training time	1,666	1,692	1,681	1,642

The results in Tables 1 to 3 show that VOiLA learns useful task-agnostic POMDP models from substantially less training data than the comparators. In FloorPositioning, VOiLA uses at least 18× less training data than the comparators while achieving comparable results. Similarly, in LeggedNavigation, VOiLA uses at least 12× less training data while achieving competitive or higher success rates across tested environments. Additionally, on our evaluation hardware and using a batch size of 1,024, the distilled transition and observation samplers in LeggedNavigation achieved approximately 8.6×10^5 and 1.2×10^6 samples per second, compared to 8.2×10^3 and 1.8×10^4 samples per second for the diffusion models, achieving up to 105× higher sampling throughput, while closely matching the diffusion models. On a held-out dataset, the standardized RMSE between samples generated by the diffusion and distilled models is 0.152 for the transition sampler and 0.274 for the observation sampler.

A key advantage of learning task-agnostic POMDP models is their potential to generalize to unseen environments. This is demonstrated in LeggedNavigation, where training data is collected only in environments 1 and 2. Although DreamerV3 and RSAC-D achieve strong performance in the training environments, their success rates drop substantially in the unseen environments 3–5, with a 0% success rate in environment 3. In the training environments, reaching the goal requires the robot to move north and pass through a gap in the upper-left part of the environment. Environment 3 instead requires the robot to move

[3] https://github.com/NM512/r2dreamer
[4] https://github.com/twni2016/pomdp-baselines/
[5] https://github.com/michaelhlim/VisualTreeSearch
[6] Intel Core i7-13850HX CPU, 32GB RAM, and NVIDIA RTX 3500 Ada Generation Laptop GPU with 12GB VRAM.

Table 2. Results on the LeggedNavigation problem for each environment in Figure 3, averaged over 50 trials. The best results are highlighted in **bold**.

Environment	Method & training transitions	Success rate	Avg. total disc. reward
1 (trained in)	VOiLA (50K)	94	298.3 ± 53.6
	DreamerV3 (600K)	86	229.7 ± 78.0
	DreamerV3 (2.6M)	88	264.8 ± 78.7
	RSAC-D (600K)	94	342.6 ± 56.7
	RSAC-D (2.6M)	**96**	$\mathbf{357.3 \pm 48.2}$
	VTS (50K)	14	-136.9 ± 67.2
2 (trained in)	VOiLA (50K)	**100**	367.5 ± 5.6
	DreamerV3 (600K)	94	321.8 ± 55.8
	DreamerV3 (2.6M)	98	372.5 ± 32.7
	RSAC-D (600K)	94	360.6 ± 60.1
	RSAC-D (2.6M)	98	$\mathbf{388.5 \pm 35.1}$
	VTS (50K)	42	85.1 ± 92.9
3	VOiLA (50K)	**98**	$\mathbf{395.2 \pm 35.4}$
	DreamerV3 (600K)	0	-385.0 ± 38.5
	DreamerV3 (2.6M)	0	-376.1 ± 40.3
	RSAC-D (600K)	0	-132.6 ± 45.0
	RSAC-D (2.6M)	0	-432.6 ± 20.5
	VTS (50K)	74	160.9 ± 97.7
4	VOiLA (50K)	**84**	$\mathbf{259.8 \pm 60.3}$
	DreamerV3 (600K)	46	-108.7 ± 112.2
	DreamerV3 (2.6M)	54	-25.1 ± 117.8
	RSAC-D (600K)	70	131.5 ± 101.9
	RSAC-D (2.6M)	46	-82.5 ± 124.1
	VTS (50K)	32	-102.2 ± 89.7
5	VOiLA (50K)	**88**	$\mathbf{289.8 \pm 62.5}$
	DreamerV3 (600K)	40	-129.6 ± 114.4
	DreamerV3 (2.6M)	44	-104.0 ± 118.9
	RSAC-D (600K)	48	-59.8 ± 112.6
	RSAC-D (2.6M)	38	-144.3 ± 123.5
	VTS (50K)	42	-43.2 ± 92.9

south and pass a lower-left gap, i.e, a qualitatively different navigation strategy from those encountered during training.

Additional training also does not necessarily improve the generalization of task-specific RL policies [29]. While the 2.6M-transition RSAC-D policy performs better in the training environments, it performs worse than the 600K-transition policy in unseen environments 4 and 5. The 2.6M-transition policy turns toward the upper-left passage earlier and cuts closer around the corner. This produces a shorter route in the training environments, but leaves less clearance in environments 4 and 5, where the passage is narrower, resulting in more collisions.

In contrast, VOiLA maintains a success rate of at least 84% across all environments. These results suggest that the learned transition, observation, and likelihood models capture reusable aspects of the environment dynamics and observation process, enabling VOiLA to adapt its online policy to previously unseen environment configurations.

Table 3. Offline training times of all methods in LeggedNavigation.

Method	Training transitions	Data collection time (s)	Model/policy training time (s)	Total training time (s)
VOiLA	**50K**	**2,194**	3,758	5,952
DreamerV3 (600K)	600K	4,764	**1,248**	6,012
DreamerV3 (2.6M)	2.6M	19,319	5,026	24,345
RSAC-D (600K)	600K	2,841	3,613	6,454
RSAC-D (2.6M)	2.6M	12,186	15,930	28,116
VTS	**50K**	**2,194**	4,056	6,250

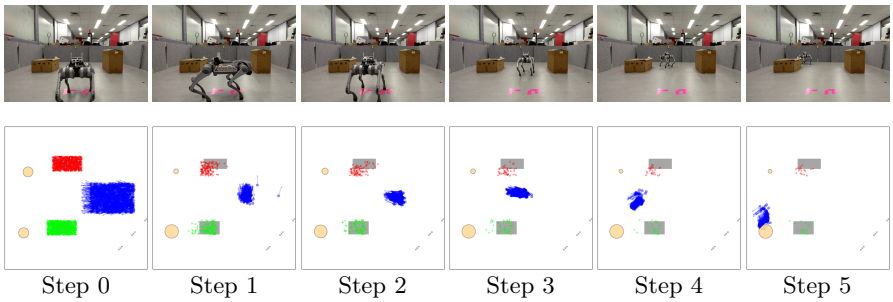


Fig. 4. Real-world execution of the Unitree Go2 quadruped and corresponding belief evolution. The top row shows images caputed during execution. The bottom row shows beliefs over robot pose (blue particles), asset location (orange circles), and movable obstacle locations (green and red particles). As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

Results on TargetFinding To see whether POMDP models learned in simulation transfer to a real robot, we deployed VOiLA on TargetFinding. Similar to the previous benchmarks, the transition, observation, and likelihood models were trained using 50K transitions collected in simulation and were transferred to the real robot without further training. Across 10 real-world test runs, including five runs for each possible target location, VOiLA successfully completed the task in every trial. Figure 4 shows an example run of the problem, while the accompanying video shows a run when the robot initially identifies a wrong target location, explores and finally finds the target. To introduce additional variability, the positions of the cardboard obstacles were manually changed between runs. These results suggest that the learned transition, observation, and likelihood models are sufficiently robust to transfer from simulation to a physical robot.

6 Conclusion

We present VOiLA, a framework for approximate POMDP solving without a priori models. VOiLA consists of two components: a task-agnostic POMDP model learning designed for online planning, and a scalable fully vectorized POMDP planner, VOPP. The model learning component combines diffusion-based generative modeling with model distillation to learn task-agnostic transition and ob-

servation samplers and observation-likelihood models that support both particle-based belief updates and online belief-space planning. VOiLA enables efficient belief-space planning while retaining the ability to represent complex transition and high-dimensional observation functions. Experimental results on three benchmark problems demonstrate that VOiLA can learn efficient POMDP models from relatively small amounts of training data, generalize to unseen environment configurations, and transfer successfully from simulation to real-world planning problems without additional training. These results suggest that combining reusable learned POMDP models with scalable online planning is a promising direction for robotic sequential decision making under uncertainty.

Acknowledgements

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] Arulampalam, M., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing **50**(2), 174–188 (2002). DOI 10.1109/78.978374

[2] Blanchard, P., Higham, D.J., Higham, N.J.: Accurately computing the log-sum-exp and softmax functions. IMA Journal of Numerical Analysis **41**(4), 2311–2330 (2021)

[3] Collins, N., Kurniawati, H.: Locally-connected interrelated network: A forward propagation primitive. IJRR (2022)

[4] Deglurkar, S., Lim, M.H., Tucker, J., Sunberg, Z.N., Faust, A., Tomlin, C.: Compositional learning-based planning for vision POMDPs. In: L4DC, pp. 469–482. PMLR (2023)

[5] Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T.: Mastering diverse control tasks through world models. Nature **640**(8059), 647–653 (2025)

[6] Hausknecht, M.J., Stone, P.: Deep Recurrent Q-Learning for Partially Observable MDPs. In: AAAI fall symposia, vol. 45, p. 141 (2015)

[7] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)

[8] Hoerger, M., Sudrajat, M., Kurniawati, H.: Vectorized online POMDP planning. In: ICRA. IEEE, Vienna, Austria (2026)

[9] Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., Dabney, W.: Recurrent experience replay in distributed reinforcement learning. In: ICLR (2018)

[10] Karkus, P., Hsu, D., Lee, W.S.: Qmdp-net: Deep learning for planning under partial observability. NeurIPS **30** (2017)

[11] Kim, E., Kurniawati, H.: Partially Observable Reference Policy Programming: Approximately Solving POMDPs. In: IJCAI (2025)

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

[13] Kurniawati, H., Yadav, V.: An online POMDP solver for uncertainty planning in dynamic environment. In: ISRR, pp. 611–629. Springer (2016)
[14] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F., et al.: A tutorial on energy-based learning. Predicting structured data **1**(0) (2006)
[15] Lee, A.X., Nagabandi, A., Abbeel, P., Levine, S.: Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. Advances in neural information processing systems **33**, 741–752 (2020)
[16] Lee, Y., Cai, P., Hsu, D.: MAGIC: Learning Macro-Actions for Online POMDP Planning . In: Robotics: Science and Systems. Virtual (2021)
[17] Mittal, M., Roth, P., Tigue, J., Richard, A., Zhang, O., Du, P., Serrano-Munoz, A., Yao, X., Zurbrügg, R., Rudin, N., et al.: Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. arXiv preprint arXiv:2511.04831 (2025)
[18] Moss, R.J., Corso, A., Caers, J., Kochenderfer, M.J.: BetaZero: Belief-State Planning for Long-Horizon POMDPs using Learned Approximations. In: Reinforcement Learning Conference (RLC) (2024)
[19] Ni, T., Eysenbach, B., Salakhutdinov, R.: Recurrent model-free RL can be a strong baseline for many POMDPs. In: ICML, pp. 16,691–16,723 (2022)
[20] Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
[21] Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Mathematics of Operations Research **12**(3), 441–450 (1987)
[22] Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R.L., Clark, A., Noury, S., et al.: Stabilizing transformers for reinforcement learning. In: ICML, pp. 7487–7498 (2020)
[23] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS **32** (2019)
[24] Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
[25] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. Nature **588**(7839), 604–609 (2020)
[26] Silver, D., Veness, J.: Monte Carlo planning in large POMDPs. In: NeurIPS, vol. 2, p. 2164–2172. Red Hook, New York (2010)
[27] Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: ICLR (2021)
[28] Sunberg, Z., Kochenderfer, M.: Online algorithms for POMDPs with continuous state, action, and observation spaces. In: ICAPS, pp. 259–263 (2018)
[29] Witty, S., Lee, J.K., Tosch, E., Atrey, A., Clary, K., Littman, M.L., Jensen, D.: Measuring and characterizing generalization in deep reinforcement learning. Applied AI Letters **2**(4), e45 (2021)
[30] Ye, N., Somani, A., Hsu, D., Lee, W.S.: DESPOT: Online POMDP planning with regularization. JAIR **58**, 231–266 (2017). DOI 10.1613/jair.5328
[31] Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR, pp. 5745–5753 (2019)