Bayesian Residual Policy Optimization:

Scalable Bayesian Reinforcement Learning with Clairvoyant Experts

Gilwoo Lee, Brian Hou, Sanjiban Choudhury, Siddhartha S. Srinivasa

***Abstract*— Informed and robust decision making in the face of uncertainty is critical for robots operating in unstructured environments. We formulate this as Bayesian Reinforcement Learning over latent Markov Decision Processes (MDPs). While Bayes-optimality is theoretically the gold standard, existing algorithms scale poorly to continuous state and action spaces. We build on the following insight: in the absence of uncertainty, each latent MDP is easier to solve. We first obtain an ensemble of experts, one for each latent MDP, and fuse their advice to compute a baseline policy. Next, we train a Bayesian residual policy to improve upon the ensemble's recommendation and learn to reduce uncertainty. Our algorithm, Bayesian Residual Policy Optimization (BRPO), imports the scalability of policy gradient methods and task-specific expert skills. BRPO significantly improves the ensemble of experts and drastically outperforms existing adaptive RL methods, both in simulated and physical robot experiments.**

I. Introduction

Robots that are deployed in the real world must operate in the face of model uncertainty. For example, an autonomous vehicle must safely navigate around pedestrians navigating to latent goals (Figure 1). A robot arm must reason about occluded objects when reaching into a cluttered shelf. This class of problems can be framed as Bayesian reinforcement learning (BRL) where the agent maintains a belief over latent Markov Decision Processes (MDPs). Under model uncertainty, agents do not know which latent MDP they are interacting with, preventing them from acting optimally. At best, they can be *Bayes optimal*, or optimal with respect to their current uncertainty over latent MDPs.

In this work, we focus on continuous control tasks with model uncertainty. This specific Bayesian RL problem is mathematically modeled by independently resampling the latent MDP at the beginning of each episode. Thus, in each episode of the autonomous vehicle example, the agent faces a new set of pedestrians with unknown goals. In these settings, the agent must actively reduce uncertainty while selecting robust actions.

A Bayesian RL problem can be viewed as a large continuous belief MDP, which is computationally infeasible to solve directly [1]. These tasks are challenging even for state-of-the-art belief-space planning and robust RL algorithms, especially when considering continuous action spaces. Existing POMDP algorithms are either limited to discrete action spaces [2] or rely on online planning and samples from the continuous action space [3]. Latent MDPs may require vastly different policies to achieve high reward; robust RL methods [4], [5] often fail to recover that multi-modality.

We build upon a simple yet recurring observation [6], [7]: ignoring uncertainty by solving individual latent MDPs is much more tractable than solving the original belief MDP. If the path for each pedestrian is known, the autonomous vehicle can invoke a motion planner to avoid collisions. We can think of these solutions as *clairvoyant experts*, i.e., experts that think they know the latent MDP and offer advice accordingly. An ensemble policy of these clairvoyant experts can be surprisingly effective, but since each expert is individually confident about which MDP the agent faces, the ensemble never prioritizes uncertainty-reducing or robust actions. Such actions can be critical for solving the original problem with model uncertainty.

Our algorithm, Bayesian Residual Policy Optimization (BRPO), computes a *residual* policy to augment an ensemble of clairvoyant experts (Figure 1). This is computed via policy optimization in a residual belief MDP, induced by the ensemble's actions on the original belief MDP. Because the ensemble is near-optimal when the entropy of the belief distribution is low, BRPO can focus on learning to act safely in regions of high entropy. Moreover, the better initialization provided by the ensemble enables BRPO to learn much faster than methods starting from scratch without experts.

Our key contribution is the following:

- We propose BRPO, a scalable Bayesian RL algorithm for problems with model uncertainty.
- We prove that BRPO monotonically improves upon the expert ensemble, converging to a Bayes-optimal policy.
- We experimentally demonstrate that BRPO outperforms both the ensemble and existing adaptive RL algorithms in simulation, and apply BRPO to a physical robot task.

II. Related Work

a) POMDP methods: Bayesian reinforcement learning formalizes RL where one has a prior distribution over possible MDPs [1]. However, the Bayes-optimal policy is intractable to compute and approximation is necessary [8]. Point-based solvers such as SARSOP [2] and PBVI [9] cannot deal with continous state actions. Online sampling, such as BAMCP [3], POMCP [10], POMCPOW [11], requires a significant amount of online computation. Online exploration is also well-studied in the bandit literature, and techniques such as posterior sampling [12] bound the learner's regret. UP-OSI [13] predicts the most likely MDP and maps that to an action. However, online methods can over-explore unsafe regimes. Another alternative is to treat belief MDPs

All authors are with Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, Washington 98195 {gilwoo, bhou, sanjibac, siddh}@cs.washington.edu.

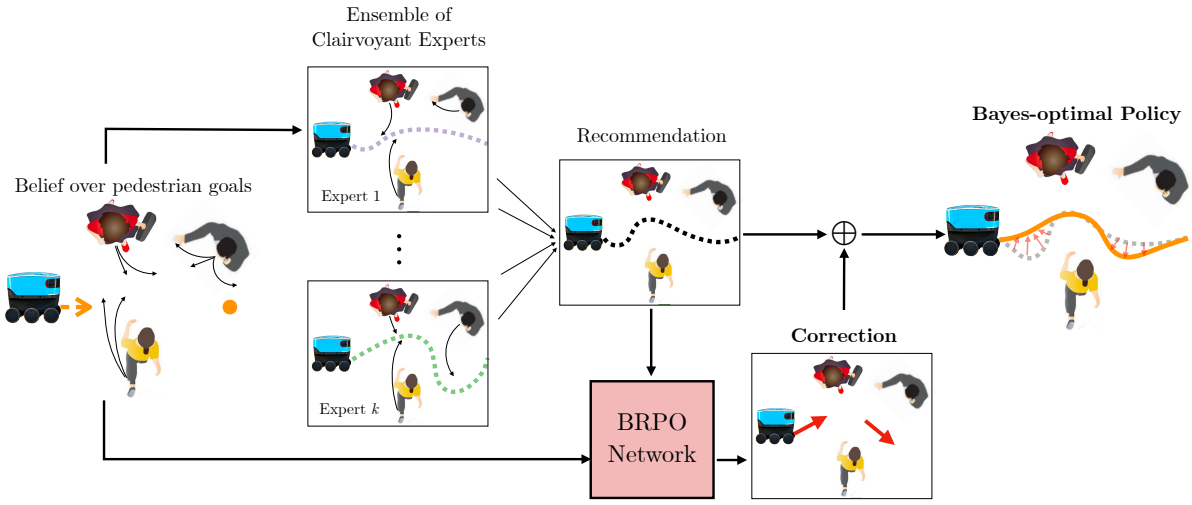


Fig. 1: An overview of Bayesian Residual Policy Optimization. (a) Pedestrian goals are latent and tracked as a belief distribution. (b) Experts propose their solutions for a scenario, which are combined into a mixture of experts. (c) Residual policy takes in the belief and ensemble's proposal and returns a correction to the proposal. (d) The combined BRPO and ensemble policy is Bayes-optimal.

as large state spaces that must be compressed. [14] use Long Short-Term Memory (LSTM) [15], while BPO [16] explicitly utilizes the belief distribution to learn a policy. Unlike BPO, BRPO leverages experts to scale to complex Bayesian tasks.

b) Meta-reinforcement Learning: Meta-reinforcement learning (MRL) approaches train sample-efficient learners by exploiting structure common to a distribution of MDPs. For example, MAML [17] trains gradient-based learners while RL2 [18] trains memory-based learners. While meta-supervised learning has well established Bayesian roots [19], [20], it was only recently that meta-reinforcement learning was strongly tied to Bayesian Reinforcement Learning (BRL) [21], [22]. Our work is more closely related to Bayesian MRL approaches. MAML-HB [23] casts MAML as hierarchical Bayes and improves posterior estimates. BMAML [24] uses non-parametric variational inference to improve posterior estimates. PLATIPUS [25] learns a parameter distribution instead of a fixed parameter. PEARL [26] learns a data-driven Bayes filter across tasks. In contrast to these approaches, we use experts at test time, learning only to optimally correct them.

c) Residual Learning: Residual learning has its foundations in boosting [27], which builds a strong ensemble by sequentially training weak learners that address the failures of their predecessors. Boosting with hand-designed policies or models allows priors to be injected into RL. Prior work has leveraged known but approximate models by learning the residual between the approximate dynamics and the discovered dynamics [28], [29], [30]. There has also been work on learning residual policies over hand-defined ones for solving long horizon [31] and complex control tasks [32]. Similarly, our approach initializes with experts and learns to improve via Bayesian reinforcement learning.

III. Preliminaries: Bayesian RL

As discussed in Section I, we formulate the problem of RL under model uncertainty as model-based Bayesian reinforcement learning (BRL), where the latent model is resampled at the beginning of each episode. Formally, the problem is defined by a tuple $\langle S, \Phi, A, T, R, P_0, \gamma \rangle$, where S is the observable state space of the underlying MDPs, Φ is the latent space, and A is the action space. T and R are the transition and reward functions parameterized by ϕ. The initial distribution over (s, ϕ) is given by $P_0 : S \times \Phi \to \mathbb{R}^+$, and γ is the discount.

Bayesian RL considers the long-term expected reward with respect to the uncertainty over ϕ rather than the true (unknown) value of ϕ. Uncertainty is represented as a *belief distribution* $b \in B$ over latent variables ϕ. The Bayes-optimal action value function is given by the Bellman equation:

$$Q(s,b,a) = R(s,b,a) + \gamma \sum_{s',b'} P(s',b'|s,b,a) \max_{a'} Q(s',b',a') \tag{1}$$

where $R(s,b,a) = \sum_{\phi \in \Phi} b(\phi) R(s,\phi,a)$ and $P(s'|s,b,a) = \sum_{\phi \in \Phi} b(\phi) P(s'|s,\phi,a)$. The posterior update $P(b'|s,b,a)$ is computed recursively: starting from initial belief b_0, $b'(\phi'|s,b,a,s') = \eta \sum_{\phi \in \Phi} b(\phi) T(s,\phi,a,s',\phi')$ where η is the normalizing constant, and the transition function is defined as $T(s,\phi,a,s',\phi') = P(s'|s,\phi,a) P(\phi'|s,\phi,a,s')$.

While some terminology is shared with online RL algorithms (e.g. Posterior Sampling Reinforcement Learning [7]), the online setting makes the different assumption that latent variables are *fixed* across multiple episodes. We refer the reader to Appendix A for further discussion.

IV. Bayesian Residual Policy Optimization

Bayesian Residual Policy Optimization relies on an ensemble of clairvoyant experts where each expert solves a

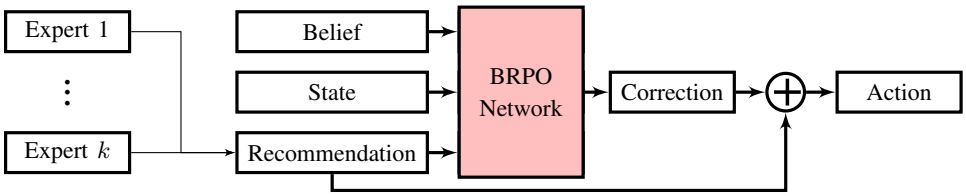


Fig. 2: Bayesian residual policy network architecture.

latent MDP. This is a flexible design parameter with three guidelines. First, the ensemble must be fixed before training begins. This freezes the residual belief MDP, which is necessary for theoretical guarantees (Section IV-C). Next, the ensemble should return its recommendation quickly since it will be queried online at test time. Practically, we have observed that this factor is often more important than the strength of the initial ensemble; even weaker ensembles can provide enough of a head start for residual learning to succeed. Finally, when the belief has collapsed to a single latent MDP, the resulting recommendation must follow the corresponding expert. In general, the ensemble should become more reliable as entropy decreases.

BRPO performs batch policy optimization in the residual belief MDP, producing actions that continuously correct the ensemble recommendations. Intuitively, BRPO enjoys improved data-efficiency because the correction can be small when the ensemble is effective (e.g., when uncertainty is low or when the experts are in agreement). When uncertainty is high, the agent learns to override the ensemble, reducing uncertainty and taking actions robust to model uncertainty.

A. Ensemble of Clairvoyant Experts

The ensemble policy maps the state and belief to a distribution over actions $\pi_e : S \times B \to P(A)$. It combines clairvoyant experts $\pi_1, \cdots, \pi_k$, one for each latent variable ϕ_i. Each expert can be computed via single-MDP RL or optimal control. There are various strategies to produce an ensemble from a set of experts. Following the maximum a posteriori (MAP) expert of the ensemble $\pi_e = \arg\max_{b(\phi)} \pi_\phi$ allows BRPO to solve tasks with infinitely many latent MDPs. The ensemble can also be a weighted sum of expert actions, which is the MAP action for Gaussian policies.

B. Bayesian Residual Policy Learning

Our algorithm is summarized in Algorithm 1. In each training iteration, BRPO collects trajectories by simulating the current policy on several MDPs sampled from the prior distribution. At every timestep of the simulation, the ensemble is queried for an action recommendation (Line 9), which is summed with the correction from the residual policy network (Figure 2) and executed (Line 10-12). The Bayes filter updates the posterior after observing the resulting state (Line 13). The collected trajectories are the input to a policy optimization algorithm, which updates the residual policy network.

The BRPO agent effectively experiences a different MDP. In this new MDP, actions are always shifted by the ensemble recommendation. We formalize this correspondence between the residual and original belief MDPs in the next section, showing that BRPO inherits the monotonic improvement guarantee from existing policy optimization algorithms.

Algorithm 1 Bayesian Residual Policy Optimization

Require: Bayes filter ψ, belief b_0, prior P_0, residual policy π_{r_0}, expert π_e, horizon T, $n_{\text{itr}}, n_{\text{sample}}$

1: **for** $i = 1, 2, \cdots, n_{\text{itr}}$ **do**
2: **for** $n = 1, 2, \cdots, n_{\text{sample}}$ **do**
3: Sample latent MDP $\mathcal{M}$: $(s_0, \phi_0) \sim P_0$
4: $\tau_n \leftarrow \texttt{Simulate}(\pi_{r_{i-1}}, \pi_e, b_0, \psi, \mathcal{M}, T)$
5: $\pi_{r_i} \leftarrow \texttt{BatchPolicyOpt}(\pi_{r_{i-1}}, \{\tau_n\}_{n=1}^{n_{\text{sample}}})$
6: **return** $\pi_{r_{best}}$

7: **procedure** SIMULATE($\pi_r, \pi_e, b_0, \psi, \mathcal{M}, T$)
8: **for** $t = 1, \cdots, T$ **do**
9: $a_{e_t} \sim \pi_e(s_t, b_t)$ // Expert recommendation
10: $a_{r_t} \sim \pi_r(s_t, b_t, a_{e_t})$ // Residual policy
11: $a_t \leftarrow a_{r_t} + a_{e_t}$
12: Execute a_t on $\mathcal{M}$, receive r_{t+1}, observe s_{t+1}
13: $b_{t+1} \leftarrow \psi(s_t, b_t, a_t, s_{t+1})$ // Belief update
14: $\tau \leftarrow (s_0, b_0, a_{r_0}, r_1, s_1, b_1, \cdots, s_T, b_T)$
15: **return** τ

C. BRPO Inherits Motononic Improvement

BRPO guarantees monotonic improvement on the expected return of the mixture between the ensemble policy π_e and the initial residual policy π_{r_0}. First, we observe that π_r operates on its own residual MDP and show that the probability of any state-sequence for π_r in the residual MDP is equal to that of π in the original MDP. Then we observe that the monotonic guarantee from the underlying policy optimization algorithm holds for π_r in the residual MDP. Combining these, we transfer the guarantee for π_r in the residual MDP to π in the original MDP. The following arguments apply to all MDPs, not just belief MDPs; thus, we've omitted the belief from the state for clarity of exposition. We refer the reader to Appendix B for proofs.

Let $\mathcal{M} = \langle S, A, T, R, P_0 \rangle$ be the original MDP. For simplicity, assume that R depends only on states. Every π_e for $\mathcal{M}$ induces a residual MDP $\mathcal{M}_r$ equivalent to $\mathcal{M}$ except for the transition function, T_r. For every residual action a_r, T_r marginalizes over all expert recommendations.

$$T_r(s'|s, a_r) = \sum_{a_e} T(s'|s, a_e + a_r)\pi_e(a_e|s) \qquad (2)$$

Let $\pi_r(a_r|s, a_e)$ be a residual policy. The final policy π executed on $\mathcal{M}$ is a mixture of π_r and π_e.

$$\pi(a|s) = \sum_{a_r} \pi_e(a - a_r|s)\pi_r(a_r|s, a - a_r) \tag{3}$$

First, we note that the probability of observing any sequence of states is equal in both MDPs. Let $\xi = (s_0, s_1, ..., s_{T-1})$ be a sequence of states. Let $\alpha = \{\tau\}$ be the set of all length T trajectories (state-action sequences) in $\mathcal{M}$ with ξ as the states, and $\beta = \{\tau_r\}$ be analogously defined for a set of trajectories in $\mathcal{M}_r$. Note that each state-sequence ξ may have multiple corresponding state-action trajectories $\{\tau\}$.

Lemma 1: The probability of ξ is equal when executing π on $\mathcal{M}$ and π_r on $\mathcal{M}_r$, i.e.,

$$\pi(\xi) = \sum_{\tau \in \alpha} \pi(\tau) = \sum_{\tau_r \in \beta} \pi_r(\tau_r) = \pi_r(\xi)$$

Since reward depends only on the states, $R(\tau) = R(\tau_r) = R(\xi)$ for all $\tau \in \alpha, \tau_r \in \beta$. Hence, Lemma 1 immediately implies that the performance of π_r on the residual MDP $\mathcal{M}_r$ is equivalent to the BRPO agent's performance on the original MDP $\mathcal{M}$.

Theorem 1: A residual policy π_r executed on $\mathcal{M}_r$ has the same expected return as the mixture policy π executed on $\mathcal{M}$.

$$\mathbb{E}_{\tau \sim (\pi, \mathcal{M})}[R(\tau)] = \mathbb{E}_{\tau_r \sim (\pi_r, \mathcal{M}_r)}[R(\tau_r)]$$

Finally, we observe that the residual policy π_r, when executed in $\mathcal{M}_r$, inherits the monotonic improvement guarantee from PPO [33], the underlying policy optimization algorithm.

Lemma 2: BRPO monotonically improves the expected return of π_r in $\mathcal{M}_r$, i.e.,

$$J(\pi_{r_{i+1}}) \geq J(\pi_{r_i})$$

with $J(\pi_r) = \mathbb{E}_{\tau \sim (\pi_r, \mathcal{M}_r)}[R(\tau)]$, where $\tau \sim (\pi_r, \mathcal{M}_r)$ indicates that τ is a trajectory with actions sampled from π_r and executed on $\mathcal{M}_r$.

Combining Theorem 1 with Lemma 2 transfers the monotonic improvement guarantee to $\mathcal{M}$.

Theorem 2: BRPO monotonically improves upon the mixture between ensemble policy π_e and initial residual policy π_{r_0}, eventually converging to a locally optimal policy.

In summary, BRPO tackles RL problems with model uncertainty by building on an ensemble of clairvoyant experts and optimizing a policy on the residual MDP induced by the ensemble. Even suboptimal ensembles often provide a strong baseline, resulting in data-efficient learning and high returns. We empirically evaluate this hypothesis in Section V.

V. Experimental Results

We focus on problems highlighting common challenges for robots with model uncertainty. In these tasks, different latent MDPs require significantly different solutions and costly sensing is needed for disambiguation. Learned policies must balance robust actions in the face of uncertainty with uncertainty-reducing actions.

In all domains that we consider, BRPO improves on the ensemble's recommendation and significantly outperforms adaptive-RL baselines that do not leverage experts (Section V-A.1). Qualitatively, robust Bayes-optimal behavior naturally emerges during training (Section V-A.2). Our ablation studies demonstrate that both the belief and ensemble recommendation are valuable (Appendix C) and that BRPO learns to reduce uncertainty without auxiliary information-gathering reward bonuses (Appendix D). Finally, through physical experiments on the MuSHR racecar platform [34], we demonstrate that BRPO agent significantly improves from a simple expert ensemble and is well-suited for real-robot tasks (Section V-B).

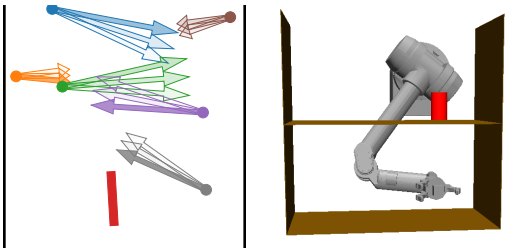
A. Simulated Experiments

a) Crowd Navigation: Inspired by [35], an autonomous agent must quickly navigate past a crowd of people without collisions. Six people cross in front of the agent at fixed speeds, three from each side (Figure 3a). Each person noisily walks toward its latent goal on the other side, which is sampled uniformly from a discrete set of destinations. The agent observes each person's speed and position to estimate the belief distribution for each person's goal. There is a single expert which uses model predictive control: each walker is simulated toward a belief-weighted average goal position, and the expert selects cost-minimizing steering angle and acceleration.

b) Cartpole: In this environment, the agent's goal is to keep the cartpole upright for as long as possible. The latent parameters are cart mass and pole length, uniformly sampled from $[0.5, 2.0]\mathrm{kg} \times [0.5, 2.0]\mathrm{m}$. The agent's estimator is a 3×3 discretization of the 2D continuous latent space, and the resulting belief is a categorical distribution over that grid. Each expert is a Linear-Quadratic Regulator (LQR) for the center of each grid square. The ensemble is the belief-weighted sum of experts.

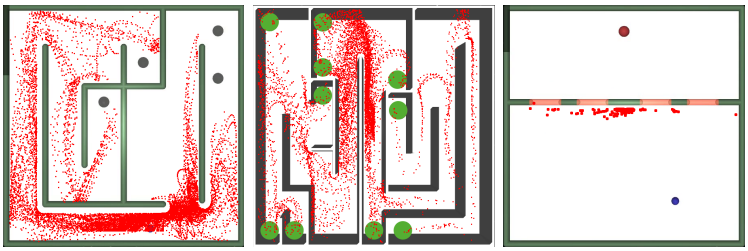
c) Object Localization: In the **`ArmShelf`** environment, the agent must localize an object without colliding with the environment or the object. The continuous latent variable is the object's pose, which is anywhere on either shelf of the pantry (Figure 3b). The agent receives a noisy observation of the object's pose upon sensing, which is less noisy as the end-effector approaches the object. The agent uses an Extended Kalman Filter to track the object's pose. The ensemble is the MAP expert which takes the MAP object pose and proposes a collision-free movement toward the object.

d) Latent Goal Mazes: In the **`Maze4`** and **`Maze10`**, the agent must identify which latent goal is active. At the beginning of each episode, the latent goal is set to one of four or ten goals. The agent is rewarded for reaching the active goal and penalized for reaching an inactive goal. The agent receives a noisy measurement of the distance to the goal, with noise proportional to the true distance. Each expert proposes an action (computed via motion planning) that navigates to the corresponding goal. The ensemble recommends the belief-weighted sum of the experts' suggestions.



(a) CrowdNav (b) ArmShelf

Fig. 3: Setup for CrowdNav and ArmShelf. In CrowdNav, the goal for the agent (red) is to drive upward without colliding with pedestrians (other colors). In ArmShelf, the goal is to reach for the can using noisy sensors.



(a) Maze4 (b) Maze10 (c) Door4

Fig. 4: Sensing locations. In Maze4 and Maze10, sensing is dense around the starting regions (bottom of Maze4, center of Maze10) and where multiple latent goals (gray, green) are nearby and must be disambiguated. In Door4, BRPO only senses when close to the doors, where the sensor is most accurate.

e) Doors: There are four possible doors to the next room of the **Door4** environment. At the beginning of each episode, each door is opened or closed with 0.5 probability. To check the doors, the agent can either sense or crash into them (which costs more than sensing). Sensing returns a noisy binary vector for all four doors with exponentially-decreasing accuracy proportional to the distance to each door. Crashing returns an accurate indicator of the door it crashed into. Each expert navigates directly through the closest open door, and the ensemble recommends the belief-weighted sum of experts.

1) BRPO Improves Ensemble, Outperforms Adaptive Methods: We compare BRPO to adaptive RL algorithms that consider the belief over latent states: BPO [16] and UP-MLE, a modification to [13] introduced by [16] that augments the state with the Bayes filter's maximum likelihood estimate. Neither approach can incorporate experts.

We also compare with the ensemble of experts baselines, which does not take any sensing actions (as discussed in Section IV). For tasks requiring explicit sensing actions (ArmShelf, Maze4, Maze10, Door4), we strengthen the ensemble by sensing with probability 0.5 at each timestep. More sophisticated sensing strategies require more task-specific knowledge to design.

Figure 5 compares the training performance of all algorithms across the six environments. Note that BRPO's initial policy does not exactly match the ensemble: the random initialization for the residual policy network adds zero-mean noise around the ensemble policy, which may result in an initial drop relative to the ensemble (Figure 5c, Figure 5d).

On the wide variety of problems we have considered, BRPO agents perform dramatically better than BPO and UP-MLE agents. BPO and UP-MLE were unable to match the performance of BRPO, except on the simple Cartpole environment. This seems to be due to the complexity of the latent MDPs. In fact, for Maze4 and Maze10, we needed to modify the reward function to encourage information-gathering for BPO and UP-MLE; without such reward bonuses, they were unable to learn any meaningful behavior. We study the effect that such a reward bonus would have on BRPO in Appendix D. For Cartpole, both BPO and UP-MLE learned to perform optimally but required much more training time than BRPO.

		BRPO	Ensemble
Real	Success Rate (%)	**96.6 (29/30)**	36.6 (11/30)
	Navigation Time (s)	$\mathbf{12.4 \pm 0.2}$	18.3 ± 0.8
Simulation	Success Rate (%)	$\mathbf{97.2 \pm 0.04}$	24.0 ± 0.2
	Navigation Time (s)	$\mathbf{6.3 \pm 0.2}$	10.5 ± 0.1

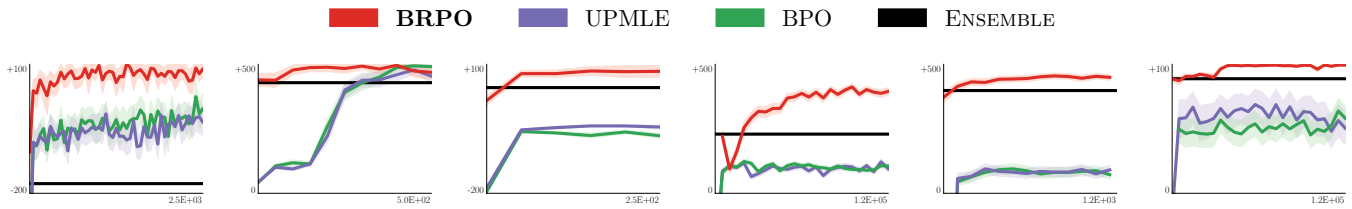
TABLE I: Comparison of BRPO and the expert ensemble on the CarNav environment. In both simulation and on the physical system, BRPO succeeds much more often and requires less time to navigate because it accelerates when safe. Navigation time is only measured for successful trials.

2) BRPO Learns Bayes-Optimal Behavior: For Maze4, Maze10 and Door4, we have visualized where the agent invokes explicit sensing (Figure 4). For Maze4 and Maze10, the BRPO agent learns to sense when goals must be distinguished, e.g. whenever the road diverges. For Door4, it senses only when that is most cost-effective: near the doors, where accuracy is highest. This results in a rather interesting policy in which the agent dashes to the wall, senses only once or twice, and drives through the closest open door. The BRPO agent avoids crashing in almost all scenarios.

B. MuSHR Car Experiment

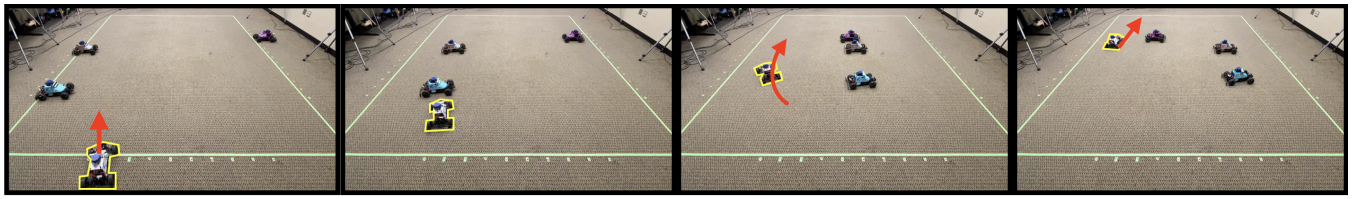
We modify CrowdNav to run an experiment with MuSHR cars [34]. The BRPO agent controls one, while three others represent pedestrians (reduced from CrowdNav due to space constraints). Each car is roughly 30 cm wide and 50 cm long, and is controlled by forward velocity and steering angle. Poses for all cars are tracked using an array of twelve OptiTrack PrimeX 22 cameras. As before, the agent aims to navigate past the "pedestrians" as they noisily move toward their latent goals (Figure 6).

We use a very simple ensemble to represent computational constraints that may be present with a physical robot. There is only one expert in this ensemble, which assumes that



(a) `CrowdNav` (b) `Cartpole` (c) `ArmShelf` (d) `Maze4` (e) `Maze10` (f) `Door4`

Fig. 5: Training curves. BRPO dramatically outperforms agents that do not leverage expert knowledge (BPO, UP-MLE), and significantly improves the ensemble of experts.



(a) Starts moving. (b) Waits for the blue car. (c) Takes slight detour. (d) Accelerates to the goal.

Fig. 6: Rollout on `CarNav`, a modified `CrowdNav` for the physical MuSHR cars. The BRPO agent waits, detours, and accelerates around other cars to reach the goal quickly.

the pedestrians will remain static as it plans forward. It uses model predictive control to avoid these obstacles as it navigates toward its goal, and is restricted to a fixed forward velocity of 0.4 m/s and steering angle between $[-0.2, 0.2]$ radians. We train the agent in simulation and execute directly on the car.

BRPO improves on this very simple baseline, successfully completing the task without collisions in 29 of 30 real-world trials. Table I compares the performance of BRPO with the ensemble in both the real and simulated environments. In both cases, BRPO dramatically improves on the ensemble's success rate. Furthermore, the BRPO agent reduces the navigation time by 36.7% in simulation and 32.2% with the physical robot, indicating that it learns to navigate both safely and quickly.

Qualitatively, the BRPO agent often starts slowly. The pedestrians' latent goals are usually clearer by the time it passes the first car, at which point it can accelerate. Depending on the latent goals, the agent occasionally waits for pedestrians to pass or detours around them. Since the expert moves at a fixed velocity, all deceleration and acceleration emerges naturally from training with BRPO. Figure 6 shows snapshots from one rollout to illustrate some of this behavior; more recorded trials are available in the supplementary video.

VI. Discussion and Future Work

Our algorithm, Bayesian Residual Policy Optimization, builds on an ensemble of experts by operating within the resulting residual belief MDP. We prove that this strategy preserves guarantees, such as monotonic improvement, from the underlying policy optimization algorithm. The scalability of policy gradient methods, combined with task-specific expertise, enables BRPO to quickly solve a wide variety of complex problems, such as navigating through a crowd of pedestrians. BRPO improves on the original ensemble of experts and achieves much higher rewards than existing Bayesian RL algorithms by sensing more efficiently and acting more robustly.

Although out of scope for this work, a few key challenges remain. First is an efficient construction of an ensemble of experts, which becomes important for continuous latent spaces with infinitely many MDPs. Infinitely many MDPs do not necessarily require infinite experts, as many may converge to similar policies. An important future direction is subdividing the latent space and computing a qualitatively diverse set of policies. Another challenge is developing an efficient Bayes filter, which is an active research area. In certain occasions, the dynamics of the latent MDPs may not be accessible, which would require a learned Bayes filter. Combined with a tractable, efficient Bayes filter and an efficiently computed set of experts, we believe that BRPO will provide an even more scalable solution for BRL problems.

Acknowledgment

This work was (partially) funded by the National Science Foundation IIS (#2007011), National Science Foundation DMS (#1839371), the Office of Naval Research, US Army Research Laboratory CCDC, Amazon, and Honda Research Institute USA. Gilwoo Lee is partially supported by Kwanjeong Educational Foundation. Brian Hou is partially supported by a NASA Space Technology Research Fellowship.

REFERENCES

[1] M. Ghavamzadeh, S. Mannor, J. Pineau, A. Tamar, *et al.*, "Bayesian reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 8, no. 5-6, pp. 359–483, 2015.

[2] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces." in *Robotics: Science and Systems*, 2008.

[3] A. Guez, D. Silver, and P. Dayan, "Efficient Bayes-adaptive reinforcement learning using sample-based search," in *Advances in Neural Information Processing Systems*, 2012.

[4] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "EPOpt: Learning robust neural network policies using model ensembles," in *International Conference on Learning Representations*, 2017.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[6] S. Choudhury, M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer, and D. Dey, "Data-driven planning via imitation learning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1632–1672, 2018.

[7] I. Osband, D. Russo, and B. Van Roy, "(more) efficient reinforcement learning via posterior sampling," in *Advances in Neural Information Processing Systems*, 2013.

[8] D. Hsu, W. S. Lee, and N. Rong, "What makes some pomdp problems easy to approximate?" in *Advances in neural information processing systems*, 2008, pp. 689–696.

[9] J. Pineau, G. Gordon, S. Thrun, *et al.*, "Point-based value iteration: An anytime algorithm for POMDPs," in *International Joint Conference on Artificial Intelligence*, 2003.

[10] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems*, 2010.

[11] Z. Sunberg and M. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *International Conference on Automated Planning and Scheduling*, 2018.

[12] I. Osband, B. V. Roy, D. J. Russo, and Z. Wen, "Deep exploration via randomized value functions," *Journal of Machine Learning Research*, vol. 20, no. 124, pp. 1–62, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-339.html

[13] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," in *Robotics: Science and Systems*, 2017.

[14] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *IEEE International Conference on Robotics and Automation*, 2018.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] G. Lee, B. Hou, A. Mandalika, J. Lee, S. Choudhury, and S. S. Srinivasa, "Bayesian policy optimization for model uncertainty," in *International Conference on Learning Representations*, 2019.

[17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[18] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[19] J. Baxter, "Theoretical models of learning to learn," in *Learning to learn*. Springer, 1998, pp. 71–94.

[20] ——, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.

[21] P. A. Ortega, J. X. Wang, M. Rowland, T. Genewein, Z. Kurth-Nelson, R. Pascanu, N. Heess, J. Veness, A. Pritzel, P. Sprechmann, S. M. Jayakumar, T. McGrath, K. Miller, M. G. Azar, I. Osband, N. C. Rabinowitz, A. György, S. Chiappa, S. Osindero, Y. W. Teh, H. van Hasselt, N. de Freitas, M. Botvinick, and S. Legg, "Meta-learning of sequential strategies," *arXiv preprint arXiv:1905.03030*, 2019.

[22] N. C. Rabinowitz, "Meta-learners' learning dynamics are unlike learners'," *arXiv preprint arXiv:1905.01320*, 2019.

[23] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based meta-learning as hierarchical bayes," *arXiv preprint arXiv:1801.08930*, 2018.

[24] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 7332–7342.

[25] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," *arXiv preprint arXiv:1806.02817*, 2018.

[26] K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," *arXiv preprint arXiv:1903.08254*, 2019.

[27] Y. Freund and R. Schapire, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[28] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 4029–4036.

[29] ——, "Conservative to confident: treating uncertainty robustly within learning-based control," in *IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 421–427.

[30] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with gaussian processes," in *2015 European Control Conference (ECC)*. IEEE, 2015, pp. 2496–2501.

[31] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[32] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[34] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Chouhury, C. Mavrogiannis, and F. Sadeghi, "MuSHR: A low-cost, open-source robotic racecar for education and research," *CoRR*, vol. abs/1908.08031, 2019.

[35] P. Cai, Y. Luo, A. Saxena, D. Hsu, and W. S. Lee, "Lets-drive: Driving in a crowd by learning from tree search," *arXiv preprint arXiv:1905.12197*, 2019.

[36] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," *arXiv preprint arXiv:1703.01732*, 2017.

[37] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 16–17.

[38] M. Chen, E. Frazzoli, D. Hsu, and W. S. Lee, "POMDP-lite for robust robot planning under uncertainty," in *IEEE International Conference on Robotics and Automation*, 2016.

APPENDIX

A. Bayesian RL and Posterior Sampling RL

Posterior Sampling Reinforcement Learning (PSRL) [7] is an online RL algorithm that maintains a posterior over latent MDP parameters ϕ. However, the problem setting it considers and how it uses this posterior are quite different than what we consider in this paper.

This work focuses on scenarios where the agent can only interact with the test MDP for a *single episode*; latent parameters are resampled for each episode. The PSRL regret analysis assumes MDPs with finite horizons and *repeated episodes* with the same test MDP, i.e. the latent parameters are fixed for all episodes.

Before each episode, PSRL samples an MDP from its posterior over MDPs, computes the optimal policy for the sampled MDP, and executes it on the fixed test MDP. Its posterior is updated after each episode, concentrating the distribution around the true latent parameters. During this exploration period, it can perform *arbitrarily poorly*. Furthermore, sampling a latent MDP from the posterior determinizes the parameters; as a result, there is no uncertainty in the sampled MDP, and the resulting optimal policies that are executed will never take sensing actions.

B. Proofs

a) Proof of Lemma 1: We prove this by induction. The base case ($T = 0$) holds trivially since $\mathcal{M}$ and $\mathcal{M}_r$ share the same initial state distribution P_0. Assuming that it holds for $T = t$, pick any ξ and let its last element be s. Consider an s'-extended sequence $\xi' = (\xi, s')$. Conditioned on ξ, the probability of ξ' is equal in $(\pi, \mathcal{M})$ and $(\pi_r, \mathcal{M}_r)$, which we can see by marginalizing over all state-action sequences:

$$\sum_{\tau'_r} \pi_r(\tau'_r|\xi) = \sum_{a_r} \pi_r(a_r|s) T_r(s'|s, a_r) \tag{4}$$

$$= \sum_{a_r} \pi_r(a_r|s) \sum_{a} T(s'|s, a) \pi_e(a - a_r|s) \tag{5}$$

$$= \sum_{a} \sum_{a_r} \pi_r(a_r|s) \pi_e(a - a_r|s) T(s'|s, a) \tag{6}$$

$$= \sum_{a} \pi(a|s) T(s'|s, a) \tag{7}$$

$$= \sum_{\tau'} \pi(\tau'|\xi) \tag{8}$$

The transition from (4) to (5) comes from (2) and (6) to (7) comes from (3). It follows that,

$$\pi(\xi') = \pi(\xi) \sum_{\tau'} \pi(\tau'|\xi) = \pi_r(\xi) \sum_{\tau'_r} \pi_r(\tau'_r|\xi) = \pi_r(\xi'),$$

which proves the lemma. Note that this proof directly leads to the proof of Theorem 1.

b) Proof of Lemma 2: BRPO uses PPO for optimization [33]. PPO's clipped surrogate objective approximates the following objective,

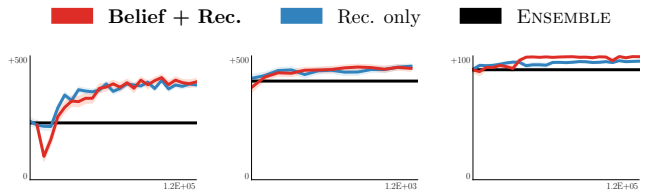
$$\max_{\theta} \hat{\mathbb{E}} \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t - \beta \cdot \text{KL}(\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)) \right],$$

where π_θ is a policy parameterized by θ and $\pi_{\theta_{\text{old}}}$ is the policy in the previous iteration, which correspond to the current and previous residual policies $\pi_{r_i}, \pi_{r_{i-1}}$ in Algorithm 1. $\hat{A}$ is the generalized advantage estimate (GAE) and KL is the Kullback–Leibler divergence between the two policy distributions. PPO proves monotonic improvement for the policy's expected return by bounding the divergence from the previous policy in each update. This guarantee only holds if both policies are applied to the same residual MDP, i.e. the ensemble is fixed.

c) Proof of Theorem 2: From Lemma 2, we have that π_r monotonically improves on the residual MDP $\mathcal{M}_r$. From Theorem 1, monotonic improvement of π_r on $\mathcal{M}_r$ implies monotonic improvement of the mixture policy π on the actual MDP $\mathcal{M}$. If the initial residual policy's actions are small, the expected return of the mixture policy π on $\mathcal{M}$ is close to that of the ensemble π_e.

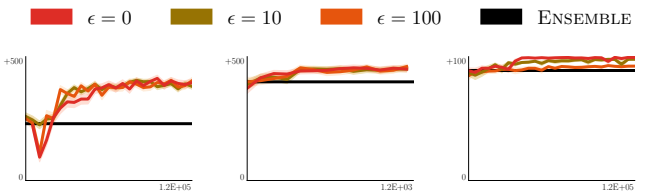
C. Ablation Study: Residual Policy Inputs

The BRPO policy takes the belief distribution, state, and ensemble recommendation as inputs (Figure 2). We considered two versions of BRPO with different inputs: only recommendation (which implicitly encodes belief), and one with both recommendation and belief.



(a) `Maze4` (b) `Maze10` (c) `Door4`

Fig. 7: Ablation study on input features. Including both belief and recommendation as policy inputs results in faster learning in `Door4`.



(a) `Maze4` (b) `Maze10` (c) `Door4`

Fig. 8: Ablation study on information-gathering reward (Equation 9). BRPO is robust to this reward.

Figure 7 shows that providing both belief and recommendation as inputs to the policy is important. Although BRPO with only the recommendation performs comparably to BRPO with both inputs on `Maze4` and `Maze10`, the one with both inputs learns faster on `Door4`.

D. Ablation Study: Information-Gathering Reward Bonuses

Because BRPO maximizes the Bayesian Bellman equation (Equation 1), exploration is incorporated into its long-term objective. As a result, auxiliary rewards to encourage exploration are unnecessary. However, existing work that does not explicitly consider the belief has suggested various auxiliary reward terms to encourage exploration, such as surprisal rewards [36] or intrinsic rewards [37]. To investigate whether such rewards benefit the BRPO agent, we augment the reward function with the auxiliary bonus from [38]:

$$\tilde{r}(s, b, a) = r(s, b, a) + \epsilon \cdot \mathbb{E}_{b'} \left[\|b - b'\|_1 \right] \tag{9}$$

where the latter term rewards belief change in belief.

Figure 8 summarizes the performance of BRPO when training with $\epsilon = 0, 10, 100$. Too much emphasis on information-gathering causes the agent to over-explore and therefore underperform. In `Door4` with $\epsilon = 100$, we qualitatively observe that the agent crashes into the doors more often. Crashing significantly changes the belief for that door; the huge reward bonus outweighs the penalty of crashing from the environment.

We find that BPO and UP-MLE are unable to learn without an exploration bonus on `Maze4`, `Maze10`, and `Door4`. We used $\epsilon = 1$ for `Maze4` and `Door4`, and $\epsilon = 100$ for `Maze10`. Upon qualitative analysis, we found that the bonus helps BPO and UP-MLE learn to sense initially, but the algorithms are unable to make further progress.