PARTICLE BELIEF REPRESENTATIONS FOR REINFORCEMENT LEARNING UNDER PARTIAL OBSERVABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Particle filters flexibly represent complex, multimodal beliefs but expose them as weighted, unordered sets incompatible with conventional fixed-dimensional policies. We study particle-belief encodings as inputs to reinforcement-learning policies while holding belief estimation and policy optimization fixed. We compare statistical summaries, cumulant-generating-function (CGF) features, and permutation-invariant set encoders. Trainable encoders are learned end-to-end or pretrained through belief reconstruction or task-specific prediction. Across eight tasks, pooling-based set encoders and gradient CGF features are reliable end-to-end, while the Set Transformer achieves the highest performance with appropriate pretraining. Task-specific pretraining helps when its target captures decision-relevant belief structure, whereas reconstruction and latent alignment provide more selective benefits. These findings provide practical guidance for selecting and training encoders for particle-filter beliefs in continuous-space partially observable Markov decision processes.

1 INTRODUCTION

Partial observability is a critical challenge in reinforcement learning (RL). Most foundational RL algorithms were developed for Markov decision processes (MDPs), where the current observation contains sufficient information to act optimally. However, an agent rarely has perfect access to the state, and the partially observable Markov decision process (POMDP) is a more realistic model. Unfortunately, in a POMDP, crucial information for choosing an optimal action could be contained in *any* observation in the history, not just the current one. A helpful classical result is that the POMDP *belief*, the state distribution given the observation history, is a sufficient statistic for decision making (Kaelbling et al., 1998b). In mainstream RL research, beliefs are rarely explicitly estimated, with high profile research projects instead using an *implicit* summary of the observation history through recurrent neural networks Vinyals et al. (2019) or more recently tokenization and world-model-based prediction Hafner et al. (2025). In contrast, in fields such as aerospace and robotics it is routine to calculate beliefs over unkown states before applying control actions Lauri et al. (2022).

Though belief estimation has downsides including the potential to track irrelevant information, it has two significant advantages. The first is *composability*. Beliefs provide a natural and unambiguous interface to communicate state estimates and uncertainty. For instance if a policy can interpret beliefs, sensors, such as cameras, lidars, inertial measurement units, or even the dynamics models or belief estimation algorithms themselves can be independently developed and improved without accessing or changing the policy. This composability framework could enable broader re-use and proliferation of AI components provided by a large ecosystem of vendors rather than only monolithic end-to-end trained systems Deglurkar et al. (2023). The second advantage is *interpretability*. In the event of a system failure, investigators can diagnose whether the belief was inconsistent with the true state or the policy made a poor decision due to misspecified objectives or another issue. Moreover, properties of the belief, such as the level uncertainty or the probability of a dangerous state can be directly constrained or penalized in the reward function.

In all but the simplest problems, however, maintaining an exact belief is computationally intractable. While efficient parametric methods such as Kalman filters generally impose a unimodal Gaussian

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

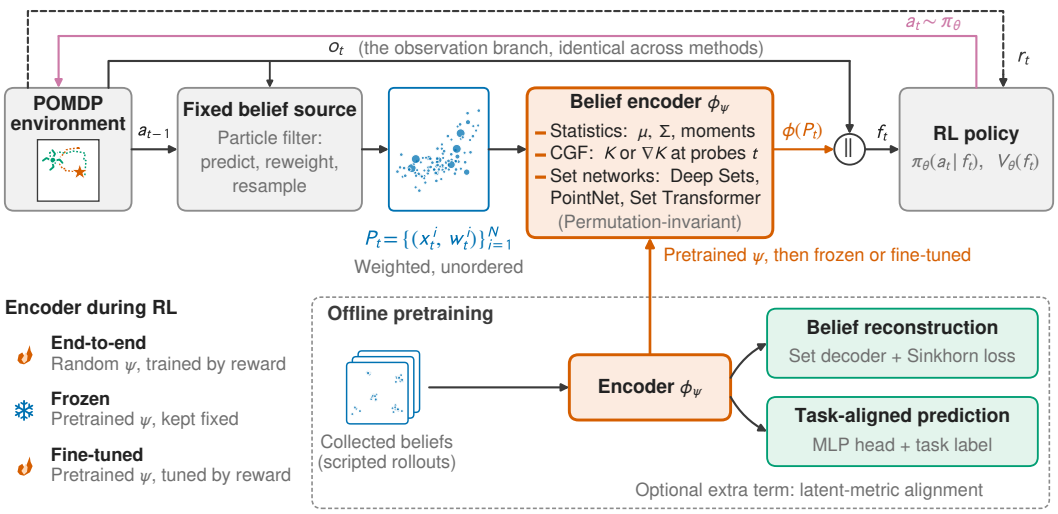


Figure 1: System overview. A fixed particle filter produces a weighted particle belief, which a belief encoder maps to a fixed-dimensional code for the RL policy. We hold the belief source and downstream policy optimization fixed while varying the encoder and, optionally, its offline pretraining and transfer mode.

approximation, particle filters can approximate complex, multimodal belief distributions and easily accommodate nonlinear dynamics. The belief is a weighted collection of state-estimate "particles":

$$b_t^N = \left\{ \left(x_t^i, w_t^i\right) \right\}_{i=1}^N, \qquad \sum_{i=1}^N w_t^i = 1, \qquad w_t^i \geq 0 \tag{1}$$

Particle beliefs also support Monte Carlo estimation of belief-dependent quantities, such as expected rewards, without constructing an explicit discretization of the full state space (Lim et al., 2023). This belief approximation is broadly applicable, but it creates a fundamental representation problem. A particle belief is not naturally represented as a fixed-dimensional vector. It is an unordered, potentially variable-size collection whose elements jointly approximate a probability distribution. Consequently, a useful particle-belief representation should be insensitive to particle ordering, account for nonuniform particle weights, preserve decision-relevant distributional structure, and remain robust to the Monte Carlo variability of the filter. Standard neural-network inputs satisfy few of these requirements. Flattening the particles imposes an arbitrary ordering and ties the policy architecture to a fixed particle count. Sorting the particles introduces a hand-designed ordering that may be discontinuous and task-dependent. Compressing the belief to its mean and covariance is computationally efficient Moss et al. (2023), but can make fundamentally different beliefs indistinguishable. For example, a unimodal belief and a symmetric multimodal belief can have similar low-order statistics while requiring entirely different information-gathering or risk-sensitive actions. The choice of belief encoding is not an implementation detail that can be overlooked, it determines what subsets of beliefs can be discriminated, what policy paradigms can ingest it and thus the class of problems it can solve.

Despite the importance of this design choice in contexts where the composability, interpretability, and flexibility advantages of particle filters are important, particle-belief representation has rarely been studied as an independent problem. Existing work has generally studied particle-belief representations within broader architectures. For example, Ma et al. (2020b) introduce an MGF-based representation as part of an end-to-end particle-filter reinforcement-learning system, making it difficult to isolate the effect of the representation itself. This leaves a central question unresolved:

> *What are the most effective particle-belief encodings for reinforcement learning, and how should they be constructed?*

We study this question in particle-filter belief MDPs, holding the particle filter, downstream reinforcement-learning algorithm, and policy architecture fixed while varying the belief encoder

and how it is trained. We compare low-order weighted statistics, observation-stacking, cumulant-generating-function features, permutation-invariant pooling networks, and attention-based set encoders across eight tasks with different belief structures and decision requirements.

Our contributions are as follows:

1. A controlled comparison of particle-belief encoders with different representational biases, while holding the particle filter, RL algorithm, and policy architecture fixed.
2. A systematic evaluation of end-to-end training, reconstruction and task-specific pretraining, frozen versus fine-tuned transfer, and latent-metric alignment.
3. An eight-task evaluation suite combining established POMDP domains with diagnostic tasks designed to isolate active information gathering, multimodality, cluster comparison, and parity-constrained belief structure.

Our results show that encoder effectiveness depends on both the structure of the belief and the information required for the downstream decision. Pooling-based set encoders and gradient CGF features are reliable under end-to-end training, whereas the Set Transformer achieves the highest performance with appropriate pretraining. Task-specific pretraining produces large gains when its target captures decision-relevant belief structure, but can be harmful when it discards uncertainty needed for information gathering; reconstruction and latent alignment provide more selective benefits. CGF features remain comparatively compact and are largely insensitive to pretraining.

2 Related Work

POMDPs: Since solving a POMDP exactly is intractable (Papadimitriou & Tsitsiklis, 1987), numerous methods approximate its solution using online tree search (Ye et al., 2017; Sunberg & Kochenderfer, 2018). These methods simulate future action–observation trajectories from the current belief. Although effective, they repeat the search at every decision and are constrained by search depth, branching, and the cost of belief updates. Extensions of online tree search support continuous state, action, and observation spaces (Lim et al., 2021), but efficiently exploring continuous actions and reasoning over long horizons remain challenging. BetaZero addresses the latter by learning policy and value approximations that guide and bootstrap tree search for discrete action spaces (Moss et al., 2023). Like our approach, BetaZero maintains a particle belief, but its neural networks receive only a Gaussian-style summary consisting of the particle mean and standard deviation. Direct policy learning provides a complementary alternative by amortizing decision making into training and avoiding online tree search at deployment. This makes the choice of particle-belief encoding especially important, since the policy can act only on information preserved by the representation.

Reinforcement Learning for POMDPs: Many reinforcement-learning approaches handle partial observability without explicitly estimating a belief, instead conditioning the policy on a recent observation window or a learned representation of the interaction history. Frame stacking provides a fixed history window (Mnih et al., 2015), while recurrent policies (Bakker, 2001; Wierstra et al., 2007; Hausknecht & Stone, 2015) and transformers (Parisotto et al., 2020) learn implicit summaries over longer histories. Although effective, these approaches couple representation learning to policy optimization and do not provide an explicit, reusable representation of uncertainty. Other work incorporates belief structure directly through differentiable filters (Karkus et al., 2017; Jonschkowski et al., 2018; Karkus et al., 2018) and particle-based latent states (Ma et al., 2020a; Igl et al., 2018). Closest to our setting, DPFRL jointly learns a particle filter and summarizes its belief for the policy using the particle mean and learned moment-generating-function features (Ma et al., 2020c). Because the filter, belief representation, and policy are learned together, the contribution of the encoding itself is difficult to isolate. We instead hold the particle filter and reinforcement-learning procedure fixed while varying the particle-belief encoder and how it is trained.

Moment- and Cumulant-Generating Functions: The moment-generating function (MGF) characterizes a distribution when it exists in a neighborhood of the origin, while derivatives of its logarithm, the cumulant-generating function (CGF), recover the distribution's cumulants (McCullagh, 1987; Billingsley, 1995). Although the complete MGF, or equivalently the complete CGF, uniquely characterizes the distribution under these conditions, evaluating either function at finitely many probe points produces only a finite-dimensional summary. DPFRL uses MGF features evaluated at learned

points to summarize a learned particle belief for reinforcement learning (Ma et al., 2020c). We use the CGF because its logarithmic form retains the information in the MGF while enabling a more numerically stable log-sum-exp computation. We evaluate the empirical CGF or its gradient with respect to the probe locations, studying each as a standalone particle-belief representation.

Permutation-Invariant Set Encoders: Particle beliefs are unordered sets, making permutation invariance a natural requirement for their encoders. Deep Sets processes elements independently before sum pooling, while PointNet uses symmetric max pooling (Zaheer et al., 2017; Qi et al., 2017); neither explicitly models interactions among elements before aggregation. The Set Transformer instead uses permutation-equivariant self-attention to model such interactions before attention-based pooling, with inducing points reducing the cost of attention (Vaswani et al., 2017; Lee et al., 2019). Related set architectures have been used in reinforcement learning to process unordered sensory inputs and relational entity representations (Tang & Ha, 2021; Zambaldi et al., 2019). Set autoencoders can be pretrained by reconstructing unordered inputs using Earth Mover's Distance (EMD), which measures the minimum cost of transporting mass between two distributions (Fan et al., 2017; Achlioptas et al., 2018; Rubner et al., 2000). We use a differentiable Sinkhorn-based optimal-transport loss to reconstruct weighted particle beliefs Cuturi (2013).

3 Preliminaries

3.1 Partially Observable Markov Decision Processes

A POMDP is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \gamma, b_0)$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the state, action, and observation spaces, respectively. These spaces may be continuous, discrete, or hybrid. $T(s' \mid s, a)$ is the transition model; $Z(o \mid s', a)$ is the observation model; $R(s, a)$ is the reward; γ is the discount factor; and b_0 is the initial-state distribution. At step t, the agent takes action a_t, the state transitions to $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, and the agent receives reward $r_t = R(s_t, a_t)$ and observation $o_{t+1} \sim Z(\cdot \mid s_{t+1}, a_t)$. Because the state is not directly observed, the agent maintains a belief b_t, a probability distribution over $\mathcal{S}$. After action a_t and observation o_{t+1}, Bayes' rule gives $b_{t+1}(s') \propto Z(o_{t+1} \mid s', a_t) \int_{\mathcal{S}} T(s' \mid s, a_t) b_t(s)\, ds$. Treating the belief as the state yields a *belief MDP*. With reward $\rho(b, a) = \mathbb{E}_{s\sim b}[R(s, a)]$, its optimal policies are also optimal for the original POMDP (Kaelbling et al., 1998a). We seek $\pi^*(a \mid b)$ maximising $\mathbb{E}_\pi[\sum_t \gamma^t r_t]$.

3.2 Particle Filters

A particle filter (Gordon et al., 1993) approximates the belief by the normalized weighted set b_t^N in equation 1. Each update propagates particles through the transition model, reweights them using the latest observation, and optionally resamples when particle degeneracy warrants it. Our framework is agnostic to the particular filtering algorithm and requires only a weighted particle set of the form in equation 1. Belief encoders consume the particle–weight pairs as an unordered set and must therefore be invariant to their joint permutation; Section 4.1 provides further details.

3.3 Problem Statement: Encoding the Belief

We decompose the state as $s_t = (s_t^{\mathrm{o}}, s_t^{\mathrm{h}})$, where s_t^{o} is observed and s_t^{h} is hidden. At time t, the filter supplies a weighted particle approximation b_t^N to the posterior over s_t^{h}. Together, s_t^{o} and this posterior specify the complete belief over s_t, which is sufficient for control. A belief encoder maps the particle belief to a fixed-dimensional representation, $\phi(b_t^N) \in \mathbb{R}^{d_\ell}$. Because the ordering of the particle–weight pairs is arbitrary, reordering these pairs cannot change $\phi(b_t^N)$. The policy and value networks receive $f_t = [s_t^{\mathrm{o}} \| \phi(b_t^N)]$, where $\|$ denotes concatenation.

4 Methodology

A particle belief can be mapped to a fixed-dimensional, permutation-invariant representation in many ways, but different representations preserve different aspects of the belief. It remains unclear which information is important for downstream control and, for learned encoders, what supervision best preserves it. Training only through policy gradients provides sparse and indirect supervision;

task-agnostic reconstruction may retain irrelevant variation, while task-specific prediction may discard uncertainty needed for information gathering. We therefore study particle-belief representation along three coupled axes: encoder architecture, pretraining objective, and transfer to reinforcement learning. For each trainable encoder, we compare end-to-end policy learning with reconstruction- and task-specific pretraining, followed by either frozen or fine-tuned transfer; we additionally study whether aligning latent and belief-space geometry improves transfer. Throughout, we hold the belief estimate, observation branch, policy and value architectures, and reinforcement-learning algorithm fixed. This design reveals which representations work and how training signals affect performance.

4.1 Belief encoders

Statistical summaries: The Gaussian encoder concatenates the particle belief's weighted mean and covariance. The moment encoder concatenates its weighted mean with the coordinate-wise central moments $\mu_r = \mathbb{E}[(X - \mathbb{E}[X])^r]$ for $r = 2, \ldots, k$. Both are inexpensive permutation-invariant encoders but discard structure beyond these statistics.

Cumulant-generating features: For a random vector X, the cumulant-generating function (CGF) is $K(t) = \log \mathbb{E}[e^{t^\top X}]$, whenever the expectation is finite. Its derivatives at the origin recover the cumulants of X, including its mean and covariance. Evaluating the CGF away from the origin therefore provides nonlinear features that can capture distributional structure beyond a fixed collection of low-order moments. For a weighted particle belief $\{(x_i, w_i)\}_{i=1}^N$, the empirical CGF and its gradient are

$$\hat{K}(t_j) = \log \sum_{i=1}^{N} w_i e^{t_j^\top x_i}. \quad (2) \qquad \nabla \hat{K}(t_j) = \sum_{i=1}^{N} \frac{w_i e^{t_j^\top x_i}}{\sum_\ell w_\ell e^{t_j^\top x_\ell}} x_i. \quad (3)$$

The gradient is the particle mean under exponentially tilted weights and emphasizes particles aligned with the probe direction t_j. Given probes $\{t_j\}_{j=1}^m$, we form the representation by concatenating either the scalar values $\hat{K}(t_j)$ or the vectors $\nabla \hat{K}(t_j)$. Probe locations are initialized over a domain-scaled range and are either held fixed or treated as trainable parameters.

Learned set networks: Deep Sets and our PointNet-style encoder augment each particle state with its scaled normalized weight, $[x_i, Nw_i]$, and process every particle using a shared MLP. Deep Sets aggregates the resulting features using the weighted sum $\sum_i w_i \psi([x_i, Nw_i])$, whereas PointNet uses coordinate-wise max pooling after excluding particles with zero weight. Positive weights therefore enter PointNet through the per-particle weight channel rather than through the pooling operation. A second MLP maps the pooled features to the belief representation. The two encoders otherwise use matched architectures, isolating weighted-sum versus max aggregation. The Set Transformer receives the same augmented particle input and uses induced self-attention to model interactions among particles before attention-based pooling (Lee et al., 2019).

4.2 Offline pretraining

We pretrain only parameterized belief encoders using two objectives; in both cases, we discard the auxiliary decoder or prediction head and transfer only the encoder to reinforcement learning.

Belief reconstruction:

We jointly train the encoder and a shared set decoder to reconstruct the weighted empirical belief. The decoder uses N learned queries to attend to the latent representation and maps the resulting features through an MLP to N uniformly weighted particle locations. A natural reconstruction loss is Earth Mover's Distance (EMD), which measures the minimum cost of transporting probability mass from the input belief to its reconstruction (Rubner et al., 2000). Computing exact EMD is expensive, so we instead minimize the debiased Sinkhorn divergence, an efficient and differentiable approximation (Cuturi, 2013; Feydy et al., 2019).

Task-specific prediction: Alternatively, an MLP prediction head reads the encoded belief and predicts a task-relevant label. This objective tests whether retaining information directly relevant to control provides a better initialization than reconstructing the complete belief.

Latent metric alignment: As an optional regularizer for either pretraining objective, we encourage distances between latent representations to preserve distances between beliefs. For a

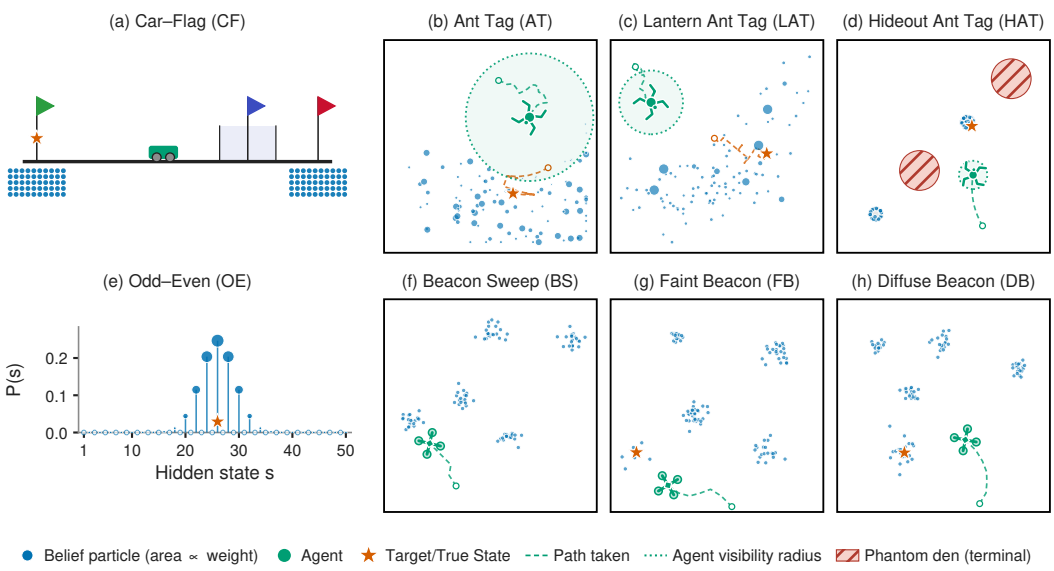


Figure 2: Overview of the eight evaluation tasks and their particle beliefs. Marker area is proportional to particle weight.

minibatch, let d_z contain the cosine distances between sampled pairs of latent codes and $d_{\mathcal{B}}$ the corresponding Sinkhorn distances between their particle beliefs, computed on the fly. We add $\mathcal{L}_{\text{align}} = \lambda[1 - \text{corr}(d_z, d_{\mathcal{B}})]$ to the pretraining loss after a short warm-up. Correlation makes the regularizer insensitive to the relative scale of the two metrics.

4.3 Encoder training and transfer

We evaluate three training strategies for parameterized belief encoders: (i) *end-to-end*, which jointly optimizes a randomly initialized encoder with the policy and value networks using PPO; (ii) *frozen transfer*, which initializes the encoder from pretraining and updates only the policy and value networks; and (iii) *fine-tuned transfer*, which initializes from pretraining and updates the encoder at 0.1 times the PPO learning rate.

5 Experiments

Our experiments ask three questions. First, which inductive biases are useful for encoding particle beliefs with qualitatively different geometry? Second, does offline pretraining improve control, and does its value depend on whether the objective preserves the full belief or only task-relevant information? Third, when a pretrained representation is transferred to reinforcement learning, is it better to freeze it or to adapt it conservatively? We answer these questions in a common PPO harness while varying only the belief encoder and its initialization within each task.

5.1 Task suite

We evaluate eight tasks from four environment families (Figure 2) and report them in three groups that differ in what the policy must extract from the belief: whether it has to move to gather information, compare the components of a multimodal belief, or resolve a categorical ambiguity. The policy receives the same non-belief observation in every belief-encoder condition; only the map from the weighted particle set to a fixed-dimensional representation changes.

Active information gathering (CF, AT, LAT, HAT): The policy must act to reduce its own uncertainty. We adapt Car–Flag and the Ant Tag family from the robot POMDP domains of Nguyen (2021). Car–Flag is a simple control task in which a hidden bit selects the rewarding end of a one-dimensional track and is revealed in an intermediate information region, yielding three possible

beliefs. The three Ant Tag variants require a MuJoCo ant to locate and tag a moving target whose unknown position is tracked using a weighted particle filter. In Ant Tag, the target becomes observable within a sensing radius and evades the ant more aggressively at close range. Lantern Ant Tag uses the same dynamics but a smaller sensing radius, leaving the target unobserved for most of the episode and increasing reliance on the belief. Hideout Ant Tag is a deliberately moment-matched variant. At the start of each episode, one of two den configurations is active. Each configuration contains two dens, and the configurations are point reflections of each other about the arena centre. The target is in one of the active configuration's dens. Within each configuration, a high-probability den near the centre counterbalances a low-probability den farther away, placing the belief mean at the centre. The configurations therefore have identical means and covariances despite having different multimodal particle distributions. The inactive pair acts as terminal hazards, so a policy that uses only the mean and covariance cannot identify the safe configuration, whereas success requires retaining higher-order structure in the belief.

Beacon tasks (BS, FB, DB): We introduce three new search-and-rescue environments that share a 20×20 arena, velocity actions in $[-1, 1]^2$, and 60-step episodes. Five survivors carry radio beacons at hidden, well-separated locations μ_j. At each step, beacon j transmits with probability ρ_j, and each received signal provides an anonymous location estimate drawn from $\mathcal{N}(\mu_j, \sigma_j^2 I)$. We represent the long-run distribution of these signals as the mixture $p(x) \propto \sum_j \rho_j \mathcal{N}(\mu_j, \sigma_j^2 I)$ and provide the policy with a weighted particle representation of this distribution at every step. This construction isolates belief representation from estimation error. Beacon Sweep rewards reaching all five survivors, removing each reached survivor from the belief. Faint and Diffuse Beacon permit only one rescue: the episode ends when the agent reaches its first survivor. In Faint Beacon, the buried survivor transmits less frequently and therefore corresponds to the component with the least mass. In Diffuse Beacon, it produces noisier location estimates and therefore corresponds to the component with the greatest spread. The agent is rewarded if it reaches the buried survivor and penalised if it reaches another survivor or the episode times out. The three tasks therefore isolate coverage, mass comparison, and scale comparison on otherwise identical state and action spaces.

Odd–Even (OE): A hidden integer $s^\star \in \{1, \ldots, 50\}$ is fixed for each 30-step episode. Each observation is drawn from a Gaussian centred at $s^\star$ but restricted to integers with the same parity, producing a weighted, comb-shaped posterior over the candidate integers. We represent this posterior exactly using one weighted particle per candidate integer. At each step, the policy predicts $s^\star$ and receives reward only for an exact match. Mean and variance capture the posterior's location and spread but not its parity-constrained, comb-shaped support, so a Gaussian encoding can conflate beliefs that place mass on different candidate integers.

5.2 Experimental comparisons

End-to-end representations: Our end-to-end study compares ten ways of constructing the policy input: the nine belief encoders defined in Section 4.1 and a frame-stacking baseline that concatenates the most recent k raw observations, thereby treating the finite observation history as an approximate Markov state. In the Ant Tag variants, each stacked frame contains the target sighting when in range (plus the filter's den prior in HAT); Odd–Even stacks the observation and time index; and the beacon tasks stack the current pings. We set k to the episode length for Car–Flag (160), Odd–Even (30), and the beacon tasks (60), and to 8 for the Ant Tag variants. The learned set encoders produce 64-dimensional belief representations and are configured to have comparable parameter counts. For the CGF encoders, we use 64 probe locations, yielding 64 features for K and 128 features for ∇K, with probe locations either fixed or learned (Section 4.1). The analytic encoders retain their natural dimensionality: the Gaussian encoder outputs the belief mean and covariance, while the moment encoder uses moments through order eight. We additionally evaluate sensitivity to the number of CGF probes and the maximum moment order on Faint and Diffuse Beacon in the appendix.

Pretraining and transfer: For the three learned set encoders and the two CGF feature maps with learned probes (K and ∇K), we evaluate reconstruction and task-specific pretraining. Each pretrained encoder is transferred to PPO either frozen or fine-tuned, as defined in Sections 4.2 and 4.3. The task-specific targets are the categorical posterior in Odd–Even, the target-cluster offset in the three beacon tasks, target position in Ant Tag and Lantern Ant Tag, and posterior mass over candidate dens in Hideout Ant Tag. For each encoder–objective pair, a single pretrained checkpoint (seed 0) initializes all downstream PPO runs, isolating variation due to downstream policy optimization.

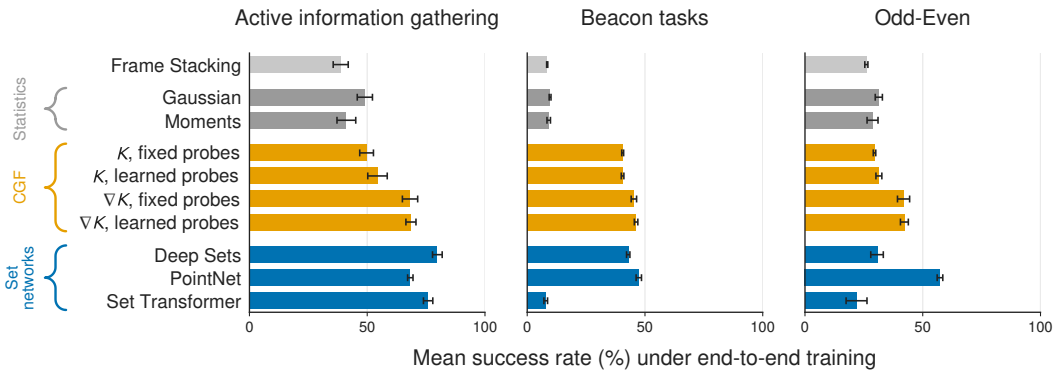


Figure 3: End-to-end training. Mean success rate per task group, one bar per encoder, coloured by family. Error bars are standard errors over five PPO seeds of the per-seed group mean.

We additionally test whether preserving belief-space geometry improves transfer by adding the latent metric-alignment term from Section 4.2 to both reconstruction and task-specific pretraining. We compare each aligned condition with its unaligned counterpart on all tasks except Car–Flag, which is evaluated only end-to-end, and Hideout Ant Tag, where unaligned pretraining already reaches the empirical performance ceiling. We use $\lambda = 0.2$ for the Ant Tag variants and Odd–Even, and $\lambda = 0.02$ otherwise, as selected by preliminary hyperparameter sweeps.

5.3 Training and evaluation protocol

For each task, all encoders use the same particle filter, preprocessing, PPO implementation, policy and value architectures, training horizon, and curriculum. We train five PPO seeds per condition for 1M environment steps on Car–Flag, 3M on Odd–Even and the beacon tasks, and 6M on the Ant Tag variants. We evaluate deterministic policies over 300 episodes and report mean task success with standard errors across seeds.

6 Results

Figures 3 and 4 summarize the main results, with per-task results reported in Appendix B. Error bars show standard errors across five PPO seeds.

The simple baselines are not uniformly weak. Gaussian statistics, higher-order moments, and frame stacking are competitive on Car–Flag, while moments and frame stacking nearly match the best encoders on Hideout Ant Tag. On the remaining tasks, however, these baselines generally trail the richer particle-belief representations.

On the active-information-gathering tasks, set encoders perform best overall under end-to-end training (left panel of Figure 3), with the gradient CGF representation close behind. These tasks require not only representing the belief, but also using uncertainty to gather information while controlling the agent. Across the two information-gathering tasks included in Figure 4, pretraining generally does not improve upon end-to-end training and sometimes reduces performance. Adding latent alignment often narrows this gap, particularly for reconstruction-pretrained encoders, but does not consistently exceed end-to-end training performance.

The beacon tasks show the clearest effect of the pretraining objective. Under end-to-end training, several encoders nearly solve Beacon Sweep, but all perform poorly on Faint Beacon and Diffuse Beacon; consequently, no encoder exceeds 50% success when averaged across the three tasks. The Set Transformer is a notable exception to the strong end-to-end performance on Beacon Sweep: it performs poorly when trained end to end, but approaches perfect success after reconstruction pretraining. Thus, learning to reconstruct the particle set appears sufficient to recover the belief geometry needed to sweep the beacons. However, the same pretraining provides little benefit on Faint Beacon and Diffuse Beacon, which require comparing properties of different particle clusters.

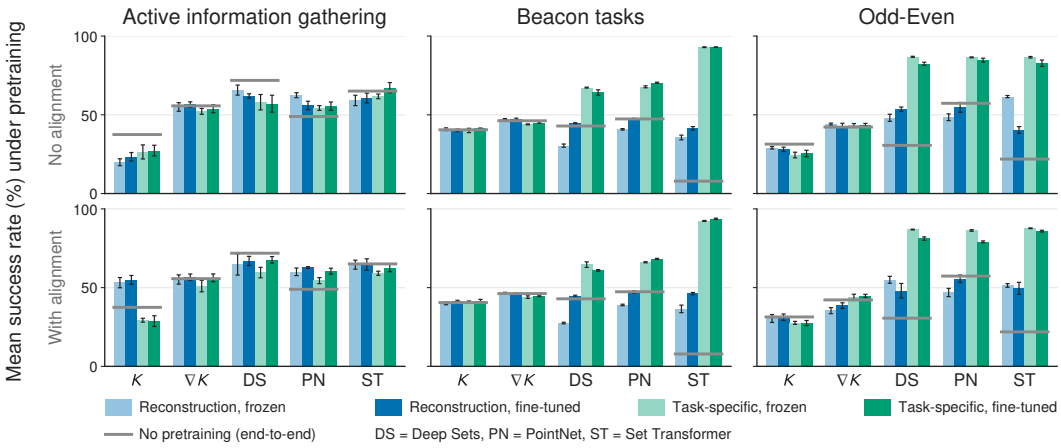


Figure 4: Pretraining and transfer for the five trainable encoders. Top: reconstruction and task-specific pretraining, each frozen or fine-tuned; bottom: the same with the latent alignment term. Bars average the tasks that have all eight conditions (AT and LAT for the active group); the grey line is the same encoder trained end-to-end; error bars as in Figure 3.

Task-specific pretraining improves all three set encoders on these comparison-based tasks, but Deep Sets and PointNet still fall short of the Set Transformer. Comparisons between particle clusters may be easier for the Set Transformer because its attention mechanism directly models particle interactions; Deep Sets and PointNet instead encode particles independently before pooling. The CGF representations gain little from pretraining, and latent alignment provides no additional benefit.

Odd–Even tests whether an encoding preserves the belief's parity-constrained, comb-shaped support rather than only its location and spread. Under end-to-end training, no encoder exceeds 57% success, with PointNet performing best and the gradient CGF representation second. Reconstruction pretraining improves the frozen Set Transformer from 22% to 62%, but provides no consistent benefit across encoders or transfer modes. Task-specific pretraining on the categorical posterior produces the clearest improvement: all three frozen set encoders reach 87% success, while their fine-tuned variants achieve 82–85%. Unlike on the beacon tasks, the three set architectures perform similarly under this objective, suggesting that direct supervision helps each preserve the fine-grained support needed for exact prediction. The CGF representations gain little from pretraining, and latent alignment again provides no consistent improvement.

Taken together, effective particle-belief encoding depends on the task structure and training objective. Pooling-based set encoders and gradient CGF features are reliable under end-to-end training, while the Set Transformer reaches the highest performance with appropriate pretraining. Task-specific pretraining helps when its target captures decision-relevant structure; reconstruction and latent alignment provide more selective, task-dependent benefits.

7 Conclusion

We studied particle-belief encoding as a distinct representation-learning problem for reinforcement learning, while holding the particle filter and downstream learning pipeline fixed. Our experiments show that the same belief estimate can lead to substantially different control performance depending on how it is represented. The central lesson is that representational fidelity alone is not sufficient: an effective encoding must make the aspects of the belief needed for decision making accessible to the policy while retaining uncertainty when information gathering is required. Thus, encoder architecture and training objective should be treated as part of the control design rather than as interchangeable implementation choices. Future work should develop task-agnostic objectives that preserve uncertainty and relational belief structure without requiring privileged task-specific labels, and evaluate whether these findings extend to higher-dimensional partially observable problems.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, including the code for the POMDP environments used to generate our training and evaluation data. The environment specifications (state and observation spaces, dynamics, and reward functions) were designed by the authors, and the implementations were verified against these specifications by two authors. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean and reformat datasets, or interpret results. Translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, edit the paper for readability, identify relevant literature, and format references. We have reviewed all AI-assisted work: all AI-assisted code was reviewed and tested by two authors; every AI-suggested reference was verified against the original source; and all AI-edited text was checked by the authors to ensure it accurately reflects our methods and claims. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects. All datasets used are publicly available and were used in accordance with their licenses. We are not aware of any conflicts of interest involving this work or its authors.

REPRODUCIBILITY STATEMENT

Section 5 describes the method and experimental setup, and Appendices C E D list all hyperparameters, and training details. Experiments were run across several compute nodes with differing hardware configurations. All experiments can be reproduced on one NVIDIA RTX 3080. We will publicly release the code and the trained model weights used for the final results upon publication.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *International Conference on Machine Learning (ICML)*, 2018.

Bram Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2001.

Patrick Billingsley. *Probability and Measure*. Wiley, 3 edition, 1995.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf.

Sampada Deglurkar, Michael H Lim, Johnathan Tucker, Zachary N Sunberg, Aleksandra Faust, and Claire Tomlin. Compositional learning-based planning for vision pomdps. In *Learning for Dynamics and Control Conference*, pp. 469–482. PMLR, 2023.

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and MMD using Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Neil J. Gordon, David J. Salmond, and Adrian F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2): 107–113, 1993.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for POMDPs. In *International Conference on Machine Learning (ICML)*, 2018.

Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. In *Robotics: Science and Systems (RSS)*, 2018.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew L. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998a.

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

Peter Karkus, David Hsu, and Wee Sun Lee. QMDP-Net: Deep learning for planning under partial observability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks with application to visual localization. In *Conference on Robot Learning (CoRL)*, 2018.

Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019.

Michael H Lim, Claire J Tomlin, and Zachary N Sunberg. Voronoi progressive widening: Efficient online solvers for continuous state, action, and observation pomdps. In *2021 60th IEEE conference on decision and control (CDC)*, pp. 4493–4500. IEEE, 2021.

Michael H Lim, Tyler J Becker, Mykel J Kochenderfer, Claire J Tomlin, and Zachary N Sunberg. Optimality guarantees for particle belief approximation of pomdps. *Journal of Artificial Intelligence Research*, 77:1591–1636, 2023.

Xiao Ma, Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5101–5108, 2020a.

Xiao Ma, Péter Karkus, David Hsu, Wee Sun Lee, and Nan Ye. Discriminative particle filter reinforcement learning for complex partial observations. *CoRR*, abs/2002.09884, 2020b. URL `https://arxiv.org/abs/2002.09884`.

Xiao Ma, Peter Karkus, David Hsu, Wee Sun Lee, and Nan Ye. Discriminative particle filter reinforcement learning for complex partial observations. In *International Conference on Learning Representations (ICLR)*, 2020c.

Peter McCullagh. *Tensor Methods in Statistics*. Chapman and Hall, 1987.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Robert J Moss, Anthony Corso, Jef Caers, and Mykel J Kochenderfer. Betazero: Belief-state planning for long-horizon pomdps using learned approximations. *arXiv preprint arXiv:2306.00249*, 2023.

Hai Nguyen. Pomdp robot domains. https://github.com/hai-h-nguyen/pomdp-domains, 2021.

Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 1987.

Emilio Parisotto, H. Francis Song, Jack W. Rae, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The Earth Mover's Distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

Zachary Sunberg and Mykel Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, pp. 259–263, 2018.

Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Sebastian Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2000.

Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, Nov 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1724-z. URL https://doi.org/10.1038/s41586-019-1724-z.

Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Solving deep memory POMDPs with recurrent policy gradients. In *International Conference on Artificial Neural Networks (ICANN)*, 2007.

Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. *Journal of Artificial Intelligence Research*, 58:231–266, January 2017. ISSN 1076-9757. doi: 10.1613/jair.5328.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Vinicius Zambaldi, David Raposo, Adam Santoro, et al. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations (ICLR)*, 2019.

A POMDP FACTORIZATION

Where unobserved $\mathcal{Y}$ and observed $\mathcal{X}$ components share a coordinate frame, as with a target location and the agent's position in the Beacon environments, the particles are expressed relative to the agent, $\tilde{y}^i = y^i - \mathbf{pos}(x_t)$ with $\mathbf{pos}(x_t)$ the agent's position, so that the belief is presented in the frame in which the agent acts. Quantities that the filter needs but the agent should not see—such as the parameters of the prior used to initialise the particles—are passed to the filter and withheld from the policy, except that the frame-stacking baseline on HAT receives the den prior (Appendix F).

In general, the Bayesian update over the hidden component $\mathcal{Y}$ of a POMDP need not have a closed form. A particle filter (Gordon et al., 1993) approximates this belief by the weighted set in equation 1 and maintains it through propagation, observation-based reweighting, and optional resampling. The initial particles are drawn from the prior over the hidden component, $y_0^i \sim \beta_0$. Particle filters represent multimodal and non-Gaussian beliefs without parametric assumptions, and were introduced to POMDP control by Thrun (2000).

B PER-TASK RESULTS

Tables 1–5 give every cell behind Figures 3 and 4: one row per task, one column per encoder and transfer mode, mean ± standard error over PPO seeds. Cells with fewer than five seeds are marked and listed in the captions. Beacon Sweep is reported here as the fraction of the five beacons collected; the summaries in the main text use the rate of collecting all five.

Table 1: End-to-end training: each encoder is trained from a random initialisation by PPO alone. Mean ± standard error over five PPO seeds, each seed evaluated on 300 episodes. Best per row in bold, second best underlined. Car–Flag: rate of episodes ending at the rewarding end; Ant Tag variants: rate of episodes in which the target is tagged; Odd–Even: exact-match rate over the steady-state steps 22–30; Beacon Sweep: fraction of the five beacons collected; Faint Beacon and Diffuse Beacon: rate of episodes that reach the correct beacon. Frame stack: the last k raw observation frames ($k = 160$ on Car–Flag, 8 on the Ant Tag variants, 30 on Odd–Even, 60 on the beacon tasks). Moments: central moments of orders 1–8.

	Statistics		CGF				Set networks			History
Task	Gaussian	Moments	K fixed	K learned	∇K fixed	∇K learned	Deep Sets	PointNet	Set Transf.	Frame Stacking
Car–Flag	0.80 ±0.12	0.70 ±0.12	0.70 ±0.12	0.70 ±0.12	<u>0.90</u> ±0.10	<u>0.90</u> ±0.10	**1.00** ±0.00	**1.00** ±0.00	**1.00** ±0.00	0.80 ±0.12
Ant Tag	0.55 ±0.06	0.18 ±0.05	0.48 ±0.02	0.58 ±0.07	0.74 ±0.02	0.75 ±0.01	<u>0.85</u> ±0.05	0.69 ±0.05	**0.87** ±0.02	0.02 ±0.01
Lantern Ant Tag	0.32 ±0.03	0.04 ±0.01	0.09 ±0.03	0.17 ±0.02	0.38 ±0.04	0.36 ±0.03	**0.59** ±0.04	0.29 ±0.02	<u>0.43</u> ±0.05	0.00 ±0.00
Hideout Ant Tag	0.29 ±0.05	0.72 ±0.01	0.71 ±0.01	0.72 ±0.01	0.70 ±0.04	<u>0.73</u> ±0.01	**0.75** ±0.01	**0.75** ±0.00	<u>0.73</u> ±0.02	<u>0.73</u> ±0.01
Odd–Even	0.31 ±0.02	0.29 ±0.02	0.30 ±0.01	0.31 ±0.01	<u>0.42</u> ±0.03	<u>0.42</u> ±0.02	0.31 ±0.03	**0.57** ±0.01	0.22 ±0.04	0.26 ±0.01
Beacon Sweep	0.22 ±0.03	0.19 ±0.04	0.86 ±0.01	0.89 ±0.00	0.92 ±0.01	0.94 ±0.01	<u>0.97</u> ±0.01	**0.98** ±0.01	0.21 ±0.01	0.24 ±0.00
Faint Beacon	0.13 ±0.01	0.12 ±0.01	**0.21** ±0.01	0.19 ±0.01	0.19 ±0.01	<u>0.20</u> ±0.01	0.18 ±0.01	0.15 ±0.01	0.12 ±0.01	0.12 ±0.01
Diffuse Beacon	0.13 ±0.01	0.13 ±0.01	0.19 ±0.01	0.17 ±0.01	<u>0.27</u> ±0.01	0.26 ±0.01	0.15 ±0.01	**0.30** ±0.03	0.11 ±0.01	0.14 ±0.01

Table 2: Reconstruction: reconstruction pretraining (weighted Sinkhorn autoencoder). One pretraining run (seed 0) initialises every PPO seed; the encoder is then either frozen (left) or fine-tuned at $0.1\times$ the PPO learning rate (right). Mean ± standard error over five PPO seeds, each seed evaluated on 300 episodes. Best per row in bold, second best underlined. Metrics as in Table 1. Not run: Car–Flag (Car–Flag has only three reachable beliefs, nothing to pretrain on).

	Frozen encoder					Fine-tuned encoder ($0.1\times$ LR)				
Task	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.
Ant Tag	0.31 ±0.05	0.70 ±0.02	**0.94** ±0.03	<u>0.90</u> ±0.02	0.79 ±0.03	0.38 ±0.03	0.78 ±0.03	0.81 ±0.05	0.77 ±0.05	0.78 ±0.03
Lantern Ant Tag	0.09 ±0.02	0.40 ±0.04	0.38 ±0.08	0.35 ±0.04	0.40 ±0.05	0.09 ±0.03	0.36 ±0.02	<u>0.43</u> ±0.04	0.34 ±0.03	**0.44** ±0.05
Hideout Ant Tag	0.71 ±0.02	0.72 ±0.02	0.72 ±0.01	**0.74** ±0.01	<u>0.73</u> ±0.02	**0.74** ±0.01	<u>0.73</u> ±0.01	0.72 ±0.02	0.69 ±0.06	**0.74** ±0.01
Odd–Even	0.29 ±0.01	0.44 ±0.01	0.48 ±0.02	0.48 ±0.02	**0.62** ±0.01	0.28 ±0.01	0.43 ±0.01	0.54 ±0.01	<u>0.55</u> ±0.03	0.40 ±0.02
Beacon Sweep	0.88 ±0.01	0.95 ±0.01	0.71 ±0.01	0.91 ±0.00	0.87 ±0.03	0.89 ±0.02	0.95 ±0.01	<u>0.97</u> ±0.01	**0.99** ±0.00	0.95 ±0.01
Faint Beacon	0.20 ±0.02	**0.24** ±0.01	0.13 ±0.01	0.17 ±0.01	0.13 ±0.02	0.19 ±0.01	0.21 ±0.01	<u>0.23</u> ±0.01	0.18 ±0.01	0.14 ±0.01
Diffuse Beacon	0.19 ±0.00	0.25 ±0.02	0.14 ±0.01	0.19 ±0.02	0.13 ±0.01	0.16 ±0.01	**0.27** ±0.02	0.16 ±0.01	<u>0.26</u> ±0.01	0.18 ±0.01

Table 3: Task-specific: task-specific pretraining (a supervised head on the task-relevant label). One pretraining run (seed 0) initialises every PPO seed; the encoder is then either frozen (left) or fine-tuned at $0.1\times$ the PPO learning rate (right). Mean $\pm$ standard error over five PPO seeds, each seed evaluated on 300 episodes. Best per row in bold, second best underlined. Metrics as in Table 1. Not run: Car–Flag (Car–Flag has only three reachable beliefs, nothing to pretrain on).

	Frozen encoder					Fine-tuned encoder ($0.1\times$ LR)				
Task	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.
Ant Tag	0.43 $\pm$0.08	0.71 $\pm$0.04	0.86 $\pm$0.01	0.78 $\pm$0.02	0.87 $\pm$0.03	0.47 $\pm$0.07	0.68 $\pm$0.05	0.88 $\pm$0.03	0.80 $\pm$0.00	**0.89** $\pm$0.02
Lantern Ant Tag	0.10 $\pm$0.02	0.33 $\pm$0.03	0.30 $\pm$0.06	0.31 $\pm$0.02	0.36 $\pm$0.02	0.08 $\pm$0.01	0.39 $\pm$0.02	0.26 $\pm$0.05	0.31 $\pm$0.03	**0.45** $\pm$0.06
Hideout Ant Tag	0.61 $\pm$0.08	0.74 $\pm$0.00	0.73 $\pm$0.01	0.72 $\pm$0.02	0.71 $\pm$0.01	0.73 $\pm$0.01	0.74 $\pm$0.01	0.76 $\pm$0.01	**0.77** $\pm$0.02	0.71 $\pm$0.02
Odd–Even	0.24 $\pm$0.02	0.43 $\pm$0.01	**0.87** $\pm$0.00	**0.87** $\pm$0.00	**0.87** $\pm$0.00	0.25 $\pm$0.02	0.44 $\pm$0.01	0.82 $\pm$0.01	0.85 $\pm$0.01	0.83 $\pm$0.02
Beacon Sweep	0.86 $\pm$0.01	0.91 $\pm$0.01	0.99 $\pm$0.00	**1.00** $\pm$0.00	**1.00** $\pm$0.00	0.87 $\pm$0.01	0.92 $\pm$0.00	**1.00** $\pm$0.00	**1.00** $\pm$0.00	**1.00** $\pm$0.00
Faint Beacon	0.18 $\pm$0.01	0.22 $\pm$0.01	0.40 $\pm$0.01	0.25 $\pm$0.01	**0.88** $\pm$0.01	0.20 $\pm$0.01	0.22 $\pm$0.01	0.40 $\pm$0.01	0.34 $\pm$0.00	0.87 $\pm$0.01
Diffuse Beacon	0.19 $\pm$0.02	0.21 $\pm$0.01	0.63 $\pm$0.01	0.79 $\pm$0.01	0.91 $\pm$0.00	0.20 $\pm$0.01	0.23 $\pm$0.01	0.52 $\pm$0.04	0.78 $\pm$0.01	**0.92** $\pm$0.01

Table 4: Reconstruction + alignment: reconstruction pretraining with the latent metric alignment term. One pretraining run (seed 0) initialises every PPO seed; the encoder is then either frozen (left) or fine-tuned at $0.1\times$ the PPO learning rate (right). Mean $\pm$ standard error over five PPO seeds, each seed evaluated on 300 episodes. Best per row in bold, second best underlined. Metrics as in Table 1. Not run: Car–Flag, Hideout Ant Tag (Car–Flag has only three reachable beliefs, nothing to pretrain on; Hideout Ant Tag has no alignment runs).

	Frozen encoder					Fine-tuned encoder ($0.1\times$ LR)				
Task	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.
Ant Tag	0.66 $\pm$0.05	0.75 $\pm$0.03	0.79 $\pm$0.08	0.87 $\pm$0.04	0.86 $\pm$0.05	0.72 $\pm$0.04	0.74 $\pm$0.05	**0.93** $\pm$0.01	0.83 $\pm$0.02	0.76 $\pm$0.05
Lantern Ant Tag	0.40 $\pm$0.03	0.36 $\pm$0.03	0.50 $\pm$0.08	0.33 $\pm$0.01	0.43 $\pm$0.06	0.38 $\pm$0.04	0.39 $\pm$0.02	0.41 $\pm$0.05	0.42 $\pm$0.02	**0.53** $\pm$0.06
Odd–Even	0.30 $\pm$0.03	0.35 $\pm$0.02	0.55 $\pm$0.02	0.47 $\pm$0.03	0.51 $\pm$0.01	0.31 $\pm$0.02	0.39 $\pm$0.02	0.48 $\pm$0.05	**0.56** $\pm$0.02	0.50 $\pm$0.04
Beacon Sweep	0.88 $\pm$0.01	0.96 $\pm$0.00	0.66 $\pm$0.01	0.88 $\pm$0.00	0.82 $\pm$0.06	0.89 $\pm$0.00	0.96 $\pm$0.01	0.97 $\pm$0.01	**1.00** $\pm$0.00	0.97 $\pm$0.01
Faint Beacon	0.16 $\pm$0.01	0.22 $\pm$0.01	0.12 $\pm$0.01	0.17 $\pm$0.01	0.16 $\pm$0.01	0.19 $\pm$0.01	0.22 $\pm$0.01	**0.23** $\pm$0.01	0.16 $\pm$0.00	0.22 $\pm$0.02
Diffuse Beacon	0.19 $\pm$0.02	0.23 $\pm$0.02	0.13 $\pm$0.01	0.17 $\pm$0.00	0.19 $\pm$0.01	0.19 $\pm$0.01	0.23 $\pm$0.01	0.15 $\pm$0.01	**0.26** $\pm$0.02	0.22 $\pm$0.01

Table 5: Task-specific + alignment: task-specific pretraining with the latent metric alignment term. One pretraining run (seed 0) initialises every PPO seed; the encoder is then either frozen (left) or fine-tuned at $0.1\times$ the PPO learning rate (right). Mean $\pm$ standard error over five PPO seeds, each seed evaluated on 300 episodes. Best per row in bold, second best underlined. Metrics as in Table 1. Not run: Car–Flag, Hideout Ant Tag (Car–Flag has only three reachable beliefs, nothing to pretrain on; Hideout Ant Tag has no alignment runs).

	Frozen encoder					Fine-tuned encoder ($0.1\times$ LR)				
Task	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.	CGF K	CGF ∇K	Deep Sets	PointNet	Set Transf.
Ant Tag	0.48 $\pm$0.02	0.68 $\pm$0.04	0.79 $\pm$0.04	0.77 $\pm$0.05	0.81 $\pm$0.03	0.48 $\pm$0.06	0.79 $\pm$0.03	0.87 $\pm$0.01	0.89 $\pm$0.02	**0.91** $\pm$0.02
Lantern Ant Tag	0.11 $\pm$0.01	0.34 $\pm$0.04	0.41 $\pm$0.03	0.32 $\pm$0.02	0.37 $\pm$0.00	0.10 $\pm$0.01	0.33 $\pm$0.04	**0.49** $\pm$0.04	0.32 $\pm$0.01	0.34 $\pm$0.04
Odd–Even	0.28 $\pm$0.01	0.44 $\pm$0.02	0.87 $\pm$0.00	0.86 $\pm$0.01	**0.88** $\pm$0.00	0.28 $\pm$0.02	0.45 $\pm$0.01	0.81 $\pm$0.01	0.79 $\pm$0.01	0.86 $\pm$0.01
Beacon Sweep	0.87 $\pm$0.01	0.94 $\pm$0.01	0.99 $\pm$0.00	**1.00** $\pm$0.00	0.99 $\pm$0.01	0.86 $\pm$0.02	0.95 $\pm$0.01	**1.00** $\pm$0.00	**1.00** $\pm$0.00	**1.00** $\pm$0.00
Faint Beacon	0.17 $\pm$0.01	0.20 $\pm$0.01	0.31 $\pm$0.01	0.25 $\pm$0.00	0.88 $\pm$0.01	0.20 $\pm$0.02	0.18 $\pm$0.01	0.32 $\pm$0.01	0.31 $\pm$0.01	**0.89** $\pm$0.01
Diffuse Beacon	0.22 $\pm$0.01	0.19 $\pm$0.01	0.64 $\pm$0.06	0.74 $\pm$0.01	0.90 $\pm$0.01	0.22 $\pm$0.01	0.22 $\pm$0.01	0.51 $\pm$0.02	0.74 $\pm$0.01	**0.92** $\pm$0.01

C BEACON TASKS: ENVIRONMENT DETAILS

Table 6: Beacon task parameters. Particle counts n_j sum to 100 over the active beacons; ρ_j is the ping probability, seen only by the frame-stacking baseline; $d(\cdot)$ is the agent's distance to a location.

	Beacon Sweep	Faint Beacon	Diffuse Beacon
Episode length	60	60	60
Min. separation of μ_j	2.5	3.0	3.0
Fix scatter σ_j	$U[0.3, 0.7]$	$U[0.3, 0.7]$	$U[0.3, 0.9]$; the widest exceeds the next by ≥ 0.2
Particle counts n_j	equal over live beacons	unique minimum $n_{\min} \sim U\{6, \ldots, 15\}$; every other $\geq n_{\min} + 5$	equal
Ping probability ρ_j	1 for live beacons	$n_j / \max_k n_k$	1
Target	every live beacon	$\arg\min_j n_j$	$\arg\max_j \sigma_j$
Active beacons (train $\rightarrow$ eval)	$1 \rightarrow 5$	$2 \rightarrow 5$	$2 \rightarrow 5$
Contact radius (train $\rightarrow$ eval)	$1.6 \rightarrow 0.6$	0.6	0.6
Termination	all reached, or timeout	first contact, or timeout	first contact, or timeout
Reward	$+10$ per beacon reached; -0.01 per step	$+20$ target, -5 other, -40 timeout; -0.01 per step	$+20$ target, -5 other, -20 timeout; -0.01 per step
Shaping potential	$20\,(\#\text{reached}) - d(\text{nearest live } \mu_j)$	$-d(\text{target } \mu_j)$	$-d(\text{target } \mu_j)$
Evaluation metric	fraction reached; all-five rate	correct / wrong / timeout	correct / wrong / timeout

Table 6 lists the three beacon tasks. Survivor locations μ_j are drawn uniformly in $[2, 18]^2$ by rejection until every pair is at least the minimum separation apart; the agent starts uniformly in the arena and moves by $p_{t+1} = \text{clip}(p_t + a_t)$, $a_t \in [-1, 1]^2$. The belief handed to the encoders is redrawn at every step: n_j particles from $\mathcal{N}(\mu_j, \sigma_j^2 I)$ per active beacon, clipped to the arena, shuffled, equally weighted, and expressed relative to the agent (divided by the arena half-width, 10). It is the stationary belief of a listener that clusters and counts anonymous pings, each group kept at its fix spread; it is not a posterior over μ_j, which would sharpen with further pings. Component identity, mass and spread, the decision variables of the three tasks, are exact in it up to sampling noise, and no filtering error enters. Reward shaping is potential-based (discount 0.99) on the true centres, which the policy never sees, and leaves the optimal policy unchanged; evaluation scores outcomes only, over 300 episodes with five active beacons and contact radius 0.6 (episode i seeded $42 + i$, deterministic actions). The curricula on the number of active beacons and, for Beacon Sweep, on the contact radius are linear over the first 40% of training.

D ANT TAG VARIANTS: ENVIRONMENT DETAILS

Table 7: Ant Tag task parameters. d is the ant–target distance, r_{vis} and r_{tag} the visibility and tag radii, $u = \mathrm{clip}\big((r_{\mathrm{vis}} - d)/(r_{\mathrm{vis}} - r_{\mathrm{tag}}), 0, 1\big)$ e the target's urgency with evasion scale e, and $H(w)$ the entropy of the particle weights. Curricula are linear over the stated fraction of training; evaluation uses the end values. Lengths are in arena units (the ant's torso has radius 0.25).

	Ant Tag	Lantern Ant Tag	Hideout Ant Tag
Episode length	400	400	300
Cage half-width	4.5	4.5	7.0
r_{vis} (train $\to$ eval)	$100 \to 3.0$ (20–50%)	$100 \to 1.5$ (20–40%)	$100 \to 1.0$ (20–50%)
r_{tag}	1.5	1.0	0.6
Target step	0.5	0.3	0.5
Evasion scale e	$0 \to 1$ over 20–50% of training; 1 at evaluation		
Target motion	one move per step relative to the ant: away with probability $0.25 + 0.45u$, stay $0.25 - 0.20u$, left or right the rest; slides along the cage wall		leashed to a disc of radius 0.4 around its den: away $0.25 + 0.45u$, toward $0.25(1-u)$, stay $0.05u$, left or right the rest; released into the Ant Tag rule over the whole cage once the ant comes within 1.4 of the empty active den
Dens		–	four candidate centres on the diagonal at distances 2.4 (near) and 6.75 (far); a mirror bit makes one near den and the opposite far den active; the target is in the near den with probability $6.75/(2.4 + 6.75) = 0.74$, which pins the prior mean at the origin; the two inactive candidates are hazards
Spawn	ant and target uniform in the cage, at least 5.0 apart		ant uniform in a disc of radius 0.5 at the centre; target uniform in its den's disc
Initial particles (100)	uniform over the cage beyond 5.0 from the ant, equal weights		mass 0.37 on each near candidate and 0.13 on each far one; resolved to the two active dens at the first update
Filter update	the target's own motion rule, no added noise; a sighting multiplies weights by $\mathcal{N}(z;\, x,\, 0.1^2 I)$, no sighting multiplies particles inside r_{vis} by 0.1; partial resampling when the effective sample size drops below 50, with jitter of half the target step; Hideout Ant Tag also multiplies by 10^{-6} the particles that a silent or triggered spook rules out		
Reward	-1 per step, 0 on the tag; no timeout penalty		-1 per step, 0 on the tag, -300 on entering a hazard
Shaping (training only)	$-c_d \,\lVert \mathrm{ant} - \mathrm{belief\ mean} \rVert - 2H(w)+$ tag bonus; c_d $1 \to 0.15$ and bonus $0 \to 50$ over 20–50%		$-c_d \,\lVert \mathrm{ant} - \mathrm{belief\ mean} \rVert +$ tag bonus; c_d $1 \to 0$ and bonus $0 \to 50$ over 30–70%
Termination	tag, or timeout	tag, or timeout	tag, hazard, or timeout
Policy observation	the 31-d MuJoCo state of the ant (positions and velocities) with the target coordinates zeroed; the target reaches the policy only through the belief		
Evaluation metric	tag rate	tag rate	tag rate; a hazard is a failure

Table 7 lists the three Ant Tag variants. All three control the MuJoCo ant with eight torque actions in $[-1, 1]$; the target is a point that moves once per step by the listed rule. The particle filter runs the same rule on 100 particles, is re-initialised every episode, and is the only channel through which the target reaches the policy. During training the visibility radius starts at 100, so the target is always visible while the ant learns to walk, and shrinks to its evaluation value; the target's evasion switches on over the same window, and the shaping terms follow the listed schedules. Evaluation

uses the environment's own reward with the end value of every curriculum, over 300 episodes with deterministic actions. Hideout Ant Tag is built so that the belief mean carries no information: the two active dens lie on opposite sides of the origin with masses in inverse proportion to their distances, so the policy has to read the posterior's shape to decide which den to search, and entering either inactive candidate ends the episode at -300.

E ODD–EVEN: ENVIRONMENT DETAILS

Table 8: Odd–Even task parameters. s^* is the hidden integer, o_t the observation drawn at step t and b_t the belief; the belief is the exact posterior, so no filtering error enters.

	Odd–Even
Hidden state s^*	one integer drawn uniformly from $\{1, \dots, 50\}$ at reset; fixed for the episode
Episode length	30 steps, with a prediction at every step
Observation o_t	one integer per step, drawn from a discretised Gaussian centred on s^* with $\sigma = \sqrt{5} + 1 \approx 3.24$ and restricted to the integers in $[1, 50]$ of the same parity as s^*; one observation at reset and one after every action, so the prediction at step t uses the first t observations
Action	discrete, 50 choices: a prediction of s^*
Reward	$+1$ for an exact match, 0 otherwise; no shaping
Belief b_t	the exact posterior over the 50 states, one particle per integer with the posterior as its weight, from a uniform prior; every opposite-parity state has weight exactly 0 from the first observation on, and the mass on the remaining 25 states sharpens with each observation, $\max_s b_t(s) > 0.9$ by step 21
Belief frame	$(s - 25.5)/24.5 \in [-1, 1]$; 50 particles
Policy input besides the belief	the normalised step index $t/30$
Termination	timeout at 30 steps
Evaluation metric	exact-match rate over the steady-state steps 22–30; the transient steps 1–21 are excluded
Reference policies	steady-state exact-match rate of the posterior mode 0.88, of repeating the previous observation 0.26, of a uniform random guess 0.02

Table 8 lists the Odd–Even task. The belief is the exact posterior, one weight per state updated by each observation's likelihood, so the encoders see a categorical distribution with one particle per state and no filtering error enters. The first observation already rules out every state of the wrong parity, so the posterior is a comb from the first step on; what changes over the episode is how sharply its mass concentrates within the parity, and by step 21 the mode holds more than 0.9 of it. We therefore score the steady state, steps 22–30, where the optimal action is the posterior mode and rounding the posterior mean names a zero-mass state in about a fifth of the steps; evaluation uses 150 episodes (episode i seeded $42 + i$, deterministic actions).

F FRAME-STACKING BASELINE

The baseline has no encoder: the policy and value MLPs read the last k raw observation frames, concatenated newest-last and zero-padded at the start of an episode (reset-frame padding on the Ant Tag variants), under each task's PPO recipe, horizon and curriculum. The particle belief is built but not shown to the policy. Table 10 gives the frame of each family; Table 9 its results.

Table 9: Frame-stacking results, five PPO seeds, the evaluation protocol of Section 5.3. Beacon Sweep: fraction of beacons reached / all-five rate. Faint and Diffuse Beacon: correct rate, and in parentheses the correct rate among episodes that end in a contact (chance 0.2).

Task	k	Score	Other k
CF	160	0.80 (three seeds solve every episode; two commit to one end)	–
AT	8	0.017	–
LAT	8	0.000	–
HAT	8	0.730	–
OE	30	0.260	$k = 8$: 0.242
BS	60	0.24 / 0.003	$k = 20$: 0.24 / 0.001; $k = 1$: 0.19 / 0.000
FB	60	0.116 (0.185)	$k = 20$: 0.135 (0.200)
DB	60	0.136 (0.191)	$k = 20$: 0.140 (0.210)

Table 10: The frame-stacking baseline's input per task family.

Task	Frame	k	Input width
CF	position, velocity, priest reading (the native observation)	160	480
AT, LAT	the 31-d MuJoCo observation, with the target's coordinates unmasked while inside the visibility radius and zero outside it	8	248
HAT	as above, plus a constant 6-vector: the two active dens (scaled by the arena half-width) and their prior masses, the filter's initial belief	8	254
OE	the observation and the time index, $(t/30,\ o_t/50)$	30	60
BS, FB, DB	agent position, then five slots of (bit, Δx, Δy): this step's anonymous pings in random slot order, agent-relative and scaled; empty slots zero	60	1020

On the beacon tasks the raw observation is generated from the same hidden layout as the belief: after the agent moves, beacon j pings with probability ρ_j (Table 6), reporting $\mu_j + \sigma_j\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, clipped to the arena; a reached beacon no longer pings. The ping generator uses its own random stream, so layouts and evaluation episodes are identical to the belief-encoder runs. Two facts qualify this baseline's results. First, the belief encoders receive the converged mixture from the first step, whereas frame stacking starts each episode with an empty history, so the comparison favours the encoders; $k = 60$ spans the whole episode, and $k = 20$ (and $k = 1$ on Beacon Sweep) gave the same results. Second, the input suffices: a hand-written rule that reads the newest frame and moves toward the nearest ping reaches all five beacons in every one of the 300 evaluation episodes of Beacon Sweep, and scores 0.08 on Faint Beacon and 0.19 on Diffuse Beacon because it ignores mass and spread. What the flat policy fails to learn under PPO is a selection over slots whose order changes at every step, which a set encoder provides by construction.

G Ablation

As the number the number of moments was increased for k-moments, or as the number of probes was increased the performance of the RL policy doesn't improve. This is true across CGF variants.

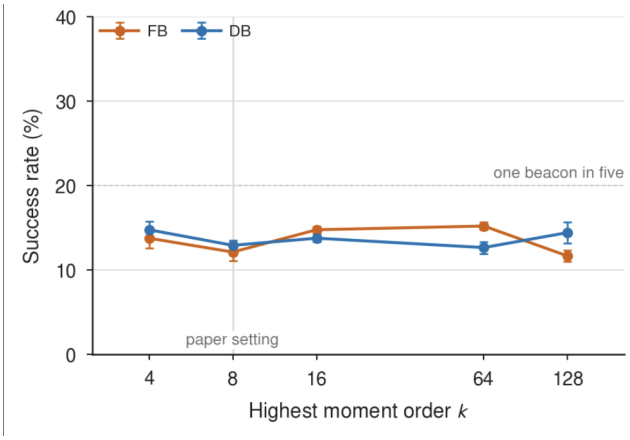


Figure 5: Sensitivity of the moment encoder to the highest moment order k on Faint Beacon (FB) and Diffuse Beacon (DB). Success rate is the fraction of [300] evaluation episodes in which the agent reaches the target beacon; points show the mean $\pm$ standard error over [five] PPO seeds. The vertical line marks the setting used in the main text, and the dashed line marks 20%, the rate of picking one of the five beacons uniformly at random.

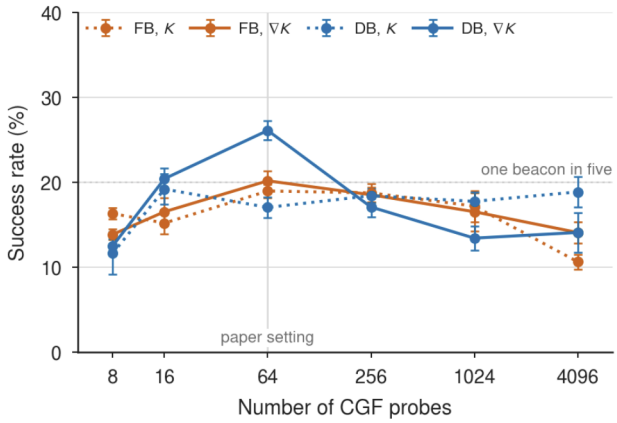


Figure 6: Sensitivity of the CGF encoder to the number of probes on Faint Beacon (FB) and Diffuse Beacon (DB), using CGF values (K, dotted) or gradients (∇K, solid) as features. Success rate, error bars, and reference lines as in Figure 5.