Planning with Latent Simulated Trajectories

Alexandre Piché[1 2], Valentin Thomas[1 2], Cyril Ibrahim [2], Julien Cornebise [2], Chris Pal [1 2 3 4]
[1] Mila, Université de Montréal, [2] Element AI,
[3] Mila, Polytechnique Montréal, [4] Canada CIFAR AI Chair

Abstract

In this work, we draw connections between planning and latent variable models[1]. Specifically, planning can be seen as introducing latent future optimal trajectories to improve the estimation of the agent's policy. This insight allows us to improve two model-based reinforcement learning (RL) algorithms: Cross Entropy Methods (CEM) and Sequential Monte Carlo Planning (SMCP). Finally, we demonstrate that our methods learn faster and achieve higher performance in early training on a continuous control benchmark.

1 Background

1.1 Control and Planning as Inference

Reinforcement Learning: We consider Markov Decision Processes (MDP) as a 6-tuple $\{\mathcal{S}, \mathcal{A}, p_{\text{env}}, r, \gamma, \mu\}$ where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, respectively. We denote $\mathbf{s}$ and $\mathbf{a}$ respectively as instances of states and actions. p_{env} represent the environment's transition dynamics, $r(\mathbf{s}_t, \mathbf{a}_t)$ is the reward function denoted by r_t, $\gamma \in [0, 1)$ is the discount factor, and μ is the distribution over initial states. We define a trajectory as a sequence of state-action pairs $\mathbf{x}_{1:T} = \{(\mathbf{s}_1, \mathbf{a}_1), \ldots, (\mathbf{s}_T, \mathbf{a}_T)\}$, with probability $p_\pi(\mathbf{x}_{1:T}) = \mu(\mathbf{s}_1) \prod_{t'=1}^{T-1} p_{\text{env}}(\mathbf{s}_{t'+1}|\mathbf{s}_{t'}, \mathbf{a}_{t'}) \prod_{t'=1}^{T} \pi(\mathbf{a}_{t'}|\mathbf{s}_{t'})$ under policy π. The RL problem is looking to find a policy π that maximizes the expected sum of rewards $\mathbb{E}_\pi[\sum_{t=1}^{T} \gamma^t r_t]$.

Control as Inference: Control as inference (Dayan & Hinton, 1997; Toussaint & Storkey, 2006; Toussaint, 2009; Rawlik et al., 2010; 2012; Ziebart, 2010; Levine & Koltun, 2013) frames the RL problem in the context of inference in a probabilistic graphical model. In the control as inference framework, the binary auxiliary variable $\mathcal{O}_t$ is introduced as being able to perform inference, and it denotes the optimality of the state-action pair $\mathbf{x}_t = (\mathbf{s}_t, \mathbf{a}_t)$. The probability of an optimal state-action pair is defined as $p(\mathcal{O}_t = 1|\mathbf{x}_t) = \exp(r(\mathbf{x}_t))$, where the 1 is omitted in the rest of the text for concision and where it is assumed that $r \leq 0$ without loss of generality. We can then consider the optimality variables as observed variables, and the state-action pair as latent variables, resulting in a probabilistic graphical model that corresponds to a Hidden Markov Model (HMM), see Figure 1. In this graphical model, the agent approximates the optimal policy which is expressed as $p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})$ [2]. The joint probability of a trajectory

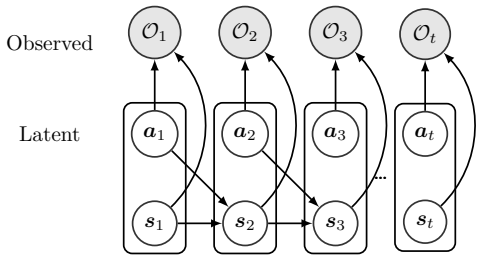


Figure 1: $\mathcal{O}_t$ is an observed optimality variable with probability $p(\mathcal{O}_t|\mathbf{x}_t) = \exp(r_t)$, where $\mathbf{x}_t = (\mathbf{s}_t, \mathbf{a}_t)$ are the state-action pair variables considered here as latent observations.

[1] Note that we used the definition of latent as found in statistics and not as the embedding space sometimes referred to as latent space.

[2] We will refer to this term as posterior approximation of the action and as an approximation of the optimal policy π^*.

occurring and being optimal under a policy π is given by:

$$p(\mathbf{x}_{1:T}, \mathcal{O}_{1:T}) = \mu(\mathbf{s}_1) \prod_{t=1}^{T-1} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \times \prod_{t=1}^{T} p(\mathbf{a}_t) \exp(r_t). \tag{1}$$

Planning as inference: Planning as inference (Attias, 2003; Piché et al., 2019) is the approximation of the distribution over simulated future optimal trajectories, $p(\mathbf{a}_t, \hat{\mathbf{x}}_{t+1:h}|\mathcal{O}_{1:T}, \mathbf{s}_t)$, where h is the planing horizon. This estimation task is related to Bayesian smoothing (Wiener, 1949; Särkkä, 2013) as we condition on the future observations: the optimality variables.

1.2 Latent Variables to Estimate a Posterior Distribution

In the context of model-based reinforcement learning, we will demonstrate how and why considering imagined trajectories as latent variables may be useful. Indeed, estimating the optimal policy $p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})$ is challenging as it depends on the future optimality variables. However, if we had access to samples $\mathbf{x}_{t+1:T} = (\mathbf{a}_{t+1}, \mathbf{s}_{t+1}, \dots \mathbf{a}_T, \mathbf{s}_T)$ from the optimal policy and the environment's dynamics, then estimating $p(\mathbf{a}_t, \mathbf{x}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})$ would be trivial using Equation 1. Fortunately, a large body of work dealing with latent variable models already exists. For the sake of clarity, in this section we denote the observed variables by $\mathbf{y} \triangleq (\mathbf{s}_t, \mathcal{O}_{t:T})$, the parameters of interest by $\mathbf{x} \triangleq \mathbf{a}_t$, and the latent variables by $\mathbf{z} \triangleq \mathbf{x}_{t+1:T}$.

The Expectation-Maximization (Dempster et al., 1977) (EM) Algorithm has been tremendously influential in showing how to optimize latent variable models to obtain key parameters of interest. Similar work by Tanner & Wong (1987) introduced the Imputation Posterior (IP) algorithm to compute the posterior $p(\mathbf{x}|\mathbf{y})$ by introducing latent variables $\mathbf{z}$. This approach is particularly well-suited for tasks where, if the latent variable $\mathbf{z}$ was known, the augmented data posterior $p(\mathbf{x}|\mathbf{z}, \mathbf{y})$ would be easier to compute than the original posterior distribution $p(\mathbf{x}|\mathbf{y})$. In this case, one could simply compute the posterior as the average of $p(\mathbf{x}, \mathbf{z}|\mathbf{y})$ over $\mathbf{z}$:

$$p(\mathbf{x}|\mathbf{y}) = \mathbb{E}_{p(\mathbf{z}|\mathbf{y})}[p(\mathbf{x}|\mathbf{z}, \mathbf{y})]. \tag{2}$$

The (refined) posterior will be the mixture of conditional density of $\mathbf{x}$ given the augmented data $\{\mathbf{z}\}_{i=1}^m$, where m is the number of samples. A problem arises as the predictive distribution over latent variables $p(\mathbf{z}|\mathbf{y})$ depends on the posterior of interest, $p(\mathbf{x}|\mathbf{y})$, as:

$$p(\mathbf{z}|\mathbf{y}) = \mathbb{E}_{p(\mathbf{x}|\mathbf{y})}[p(\mathbf{z}|\mathbf{x}, \mathbf{y})]. \tag{3}$$

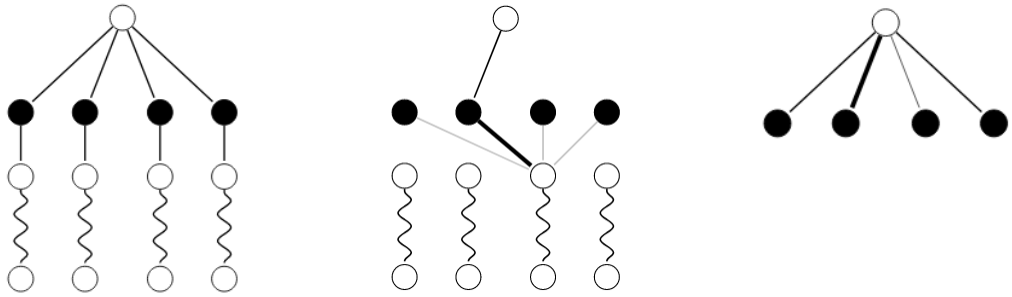
Tanner & Wong (1987) demonstrated that an approximation of $p(\mathbf{x}|\mathbf{y})$ can be used to build an iterative algorithm that will converge to the true posterior distribution $p(\mathbf{x}|\mathbf{y})$. The algorithm is given by:

1) Provide an initial distribution $p^{(0)}(\mathbf{x}|\mathbf{y})$ as the current approximation of the posterior.
2) Approximate the predictive distribution:
$$p^{(k+1)}(\mathbf{z}|\mathbf{y}) \leftarrow \mathbb{E}_{p^{(k)}(\mathbf{x}|\mathbf{y})}[p(\mathbf{z}|\mathbf{x}, \mathbf{y})].$$
3) Update the posterior approximation:
$$p^{(k+1)}(\mathbf{x}|\mathbf{y}) \leftarrow \mathbb{E}_{p^{(k+1)}(\mathbf{z}|\mathbf{y})}[p(\mathbf{x}|\mathbf{z}, \mathbf{y})].$$
4) Increment $k \leftarrow k + 1$ and return to step 2.

2 Latent Simulated Trajectories

To exhibit intelligent behaviour and choose the best actions, RL agents must be able to combine simulations originating from their internal model with their past experiences interacting with the world. We propose a simple planning method that combines simulation with the agent's policy inspired by Tanner & Wong (1987).

We first introduce the distributions required to derive our planning algorithm.



(a) We draw samples from the current approximation of $p(\mathbf{x}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})$.

(b) We update the action distribution using Equation 4.

(c) We re-weigh our action distribution proportional to the weights obtained.

Figure 2: The white nodes represent states and the black nodes represent actions. We illustrate the algorithm by explicitly showing how the third trajectory is used to update the weight of the second action. The algorithm is iterative, thus, at step (c) we either return to step (a) or sample an action from the current approximation.

- The posterior distribution $p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})$: probability of an action occurring given a state and the optimality variables observed.
- The predictive density $p(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})$: the distribution over simulated future trajectories given a state-action pair $(\mathbf{s}_t, \mathbf{a}_t)$ and the optimality variables observed.
- The joint density $p(\mathbf{a}_t, \hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})$: probability of a trajectory being generated given the optimality variables observed. It is proportional to the probability of a trajectory occurring and being optimal.

We will now examine the two parts of our algorithm Latent Simulated Trajectories (LST): i) incorporating simulations of optimal trajectories in the model-free policy, and ii) sampling simulated optimal trajectories.

2.1 Incorporating Simulations to Approximate the Optimal Policy

The agent can improve its current approximation of the posterior distribution over actions by considering the joint distribution with simulated trajectories $\hat{\mathbf{x}}_{t+1:T}$ and then marginalizing the simulated trajectories to recover the posterior of interest:

$$p^{(k+1)}(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T}) \leftarrow \mathbb{E}_{p^{k+1}(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}[p(\mathbf{a}_t|\hat{\mathbf{x}}_{t+1:T}, \mathbf{s}_t, \mathcal{O}_{t:T})]$$
$$= \mathbb{E}_{p^{k+1}(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}\left[\frac{p(\mathbf{s}_{t+1}|\mathbf{a}_t, \mathbf{s}_t)p^k(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})}{\mathbb{E}_{p^k(\mathbf{a}'_t|\mathbf{s}_t, \mathcal{O}_{t:T})}[p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}'_t)]}\right]. \quad (4)$$

The complete derivation can be found in the Appendix A.2. Intuitively, the numerator provides information on the likelihood of a transition given an action and its current probability under the current posterior approximation. The denominator corrects this probability by the ease of which it is to achieve this state under the current posterior approximation. The algorithm is provided in Appendix A.3. Our method contrasts with the Model Predictive Control (MPC) (Garcia et al., 1989) method which simply takes the first action of an estimated optimal trajectory.

3 Experiments

3.1 Continuous Control Benchmark

We compare our approach (LST) with different MPC methods on a suite of continuous control tasks. We used the Mujoco benchmark (Todorov et al., 2012) and the OpenAI gym environment (Brockman et al., 2016) to perform our experiments.

We benchmarked two classical planning algorithms endowed with the LST posterior: Cross Entropy Methods (CEM) (Rubinstein & Kroese, 2004) and Sequential Monte Carlo Planning (SMCP) (Piché et al., 2019). Our improved versions of CEM and SMCP are referred to as CEM-LST and SMCP-LST, while the traditional approaches are specified as CEM-MPC and SCMP-MPC. We also included soft actor critic (SAC) (Haarnoja et al., 2018) to highlight the advantages of model-based methods.

For all methods requiring a model of the environment p_{model}, we trained an ensemble of probabilistic models as proposed by Chua et al. (2018) by minimizing the negative log likelihood (NLL) of a Gaussian distribution with diagonal covariance $\mathcal{N}(\Delta\mathbf{s}_t; \mu_\theta(\mathbf{s}_t, \mathbf{a}_t), \sigma_\theta)$. We evaluate the likelihood of a sampled transitions by treating the ensemble model as a mixture of Gaussians for more robust uncertainty estimates (Lakshminarayanan et al., 2017).

Our algorithms are compared on 3 different environments: Reacher, InvertedPendulum and HalfCheetah. i) Reacher is the simplest environment and does not require consideration of uncertainty given that it is very deterministic. The improvement of LST over MPC methods are thus unsurprisingly very small. ii) In InvertedPendulum, a small change in state might result in large differences in the outcome, our methods outperformed MPC methods by a wide margin. iii) Taking into account the uncertainty of the model also provides performance gain on HalfCheetah. In summary, CEM-LST only achieved the highest return on the first interaction with the environment on two of the three tasks, whilst SMCP-LST achieved the highest return on all three environments for the interval observed.

Note that our results differ from Chua et al. (2018) as they modified the environments so that the reward could be computed directly from the state [3].

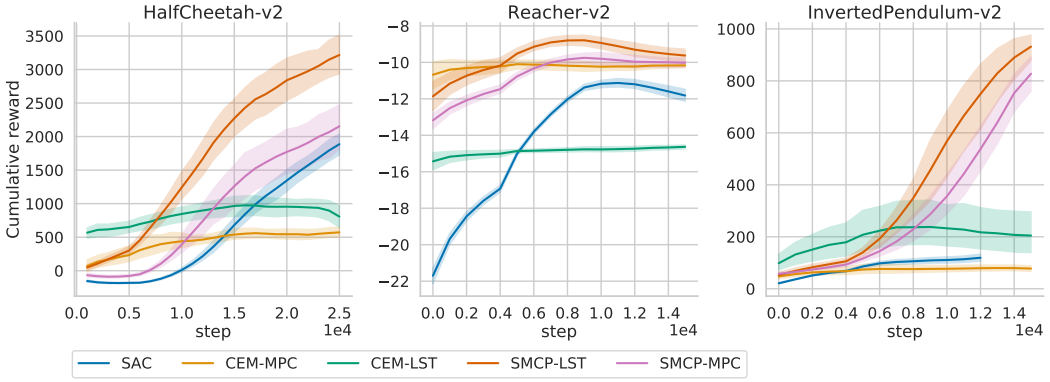


Figure 3: Early training learning curves. Our method is referred to as LST. We see that sampling actions from the posterior provided by LST rather that using MPC can lead to faster and higher asymptotical performance.

4 Discussion

In this work, we introduced a novel perspective on planning: the use of latent future optimal trajectories to refine the current policy. This perspective allowed us to improve CEM and SMCP. We believe our perspective on planning is an important stepping stone to build more efficient model-based reinforcement learning agents. We now discuss the limitations of our method, some potential solutions, and future direction of research.

One of the limitations of our method is that the joint distribution over trajectories and optimality (Equation 1) is a function of the exponential of the reward. Optimizing such an objective may lead to policies that do not maximize the average reward and behave overly optimistically (Levine, 2018). This behaviour may indeed lead the agent to take a risky action in order to have an unlikely but high reward.

[3] https://github.com/kchua/handful-of-trials/tree/e1a62f217508a384e49ecf7d16a3249e187bcff9/dmbrl/env

As mentioned in the results section, planning by sampling can be quite computationally expensive. Future work aims to reduce the computational burden by using fully differentiable trajectory optimization methods (Heess et al., 2015; Mishra et al., 2017) which is faster than sampling.

In conclusion, we believe that our planning framework can act as a stepping stone to designing efficient planning algorithms.

Acknowledgments

The authors would like to thank Stephanie Long, Salem Lahlou, Yoshua Bengio, and Kris Sankaran for proof-reading and commenting a previous version of this paper. Gunshi Gupta for reviewing the code. AP and VT were supported by Open Philantropy.

References

Hagai Attias. Planning by probabilistic inference. In AISTATS. Citeseer, 2003.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. arXiv preprint arXiv:1805.12114, 2018.

Peter Dayan and Geoffrey E Hinton. Using expectation-maximization for reinforcement learning. Neural Computation, 9(2):271–278, 1997.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society. Series B (methodological), pp. 1–38, 1977.

Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practice—a survey. Automatica, 25(3):335–348, 1989.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290, 2018.

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In Advances in Neural Information Processing Systems, pp. 2944–2952, 2015.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Advances in Neural Information Processing Systems, pp. 6402–6413, 2017.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. arXiv preprint arXiv:1805.00909, 2018.

Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In Advances in Neural Information Processing Systems, pp. 207–215, 2013.

Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. Prediction and control with temporal segment models. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 2459–2468. JMLR. org, 2017.

Alexandre Piché, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential monte carlo methods. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=ByetGn0cYX.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. An approximate inference approach to temporal optimization in optimal control. In Advances in neural information processing systems, pp. 2011–2019, 2010.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In Robotics: science and systems, volume 13, pp. 3052–3056, 2012.

RY Rubinstein and DP Kroese. A unified approach to combinatorial optimization, monte-carlo simulation, and machine learning. Springer-Verlag New York, LLC, 2004.

Simo Särkkä. Bayesian filtering and smoothing, volume 3. Cambridge University Press, 2013.

Martin A Tanner and Wing Hung Wong. The calculation of posterior distributions by data augmentation. Journal of the American statistical Association, 82(398):528–540, 1987.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 5026–5033. IEEE, 2012.

Marc Toussaint. Robot trajectory optimization using approximate inference. In Proceedings of the 26th annual international conference on machine learning, pp. 1049–1056. ACM, 2009.

Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In Proceedings of the 23rd international conference on Machine learning, pp. 945–952. ACM, 2006.

Norbert Wiener. Extrapolation, interpolation and smoothing of stationary time series-with engineering applications mit press. Google Scholar, 1949.

Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. PhD thesis, CMU, 2010.

A Appendix

A.1 Abbreviation and Notation

Table 1: Abbreviation

LST:	Latent Simulated Trajectories.
SMCP:	Sequential Monte Carlo Planning.
SAC:	Soft Actor Critic.
CEM:	Cross Entropy Method.
RS:	Random Shooting.
MCTS:	Monte Carlo Tree Search.
SMC:	Sequential Monte Carlo.
SMCP:	Sequential Monte Carlo Planning.
IS:	Importance Sampling.
MPC:	Model Predictive Control.

Table 2: Notation

$\mathbf{x}_{1:T}$	$\triangleq$	$\{\mathbf{s}_i, \mathbf{a}_i\}_{i=1}^T$ the state-action pairs.
V	$\triangleq$	value function.
$\mathcal{O}_t$	$\triangleq$	Optimality variable.
$p(\mathcal{O}_t \vert \mathbf{s}_t, \mathbf{a}_t)$	$\triangleq$	$\exp(r(\mathbf{s}_t, \mathbf{a}_t))$ Probability of a pair state action of being optimal.
p_{env}	$\triangleq$	Transition probability of the environment. Takes state and action $(\mathbf{s}_t, \mathbf{a}_t)$ as argument and return next state and reward $(\mathbf{s}_{t+1}, r_t)$.
p_{model}	$\triangleq$	Model of the environment. Takes state and action $(\mathbf{s}_t, \mathbf{a}_t)$ as argument and return next state and reward $(\mathbf{s}_{t+1}, r_t)$.
w_t	$\triangleq$	Importance sampling weight.
$p(\mathbf{x})$	$\triangleq$	Density of interest.
$q(\mathbf{x})$	$\triangleq$	Approximation of the density of interest.
$t \in \{1, \dots T\}$	$\triangleq$	time steps.
$n \in \{1, \dots N\}$	$\triangleq$	particle number.
h	$\triangleq$	horizon length.

A.2 Complete Derivation

$$
\begin{aligned}
p^{k+1}(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T}) &= \mathbb{E}_{p^{k+1}(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}[p^k(\mathbf{a}_t|\mathbf{s}_t, \hat{\mathbf{x}}_{t+1:T}, \mathcal{O}_{t:T})] \\
&= \mathbb{E}_{p^{k+1}(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}\left[\frac{p(\mathbf{a}_t, \hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}{p(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}\right] \\
&= \mathbb{E}_{p^{k+1}(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}\left[\frac{p(\mathbf{x}_{t+2:T}, \mathbf{a}_{t+1}|\mathbf{s}_{t+1}, \mathcal{O}_{t:T})p(\mathbf{s}_{t+1}|\mathbf{a}_t, \mathbf{s}_t)p^k(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})}{p(\mathbf{x}_{t+2:T}, \mathbf{a}_{t+1}|\mathbf{s}_{t+1}, \mathcal{O}_{t:T})p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathcal{O}_{t:T})}\right] \\
&= \mathbb{E}_{p^{k+1}(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}\left[\frac{p(\mathbf{s}_{t+1}|\mathbf{a}_t, \mathbf{s}_t)p^k(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})}{p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathcal{O}_{t:T})}\right] \\
&= \mathbb{E}_{p^{k+1}(\hat{\mathbf{x}}_{t+1:T}|\mathbf{s}_t, \mathcal{O}_{t:T})}\left[\frac{p(\mathbf{s}_{t+1}|\mathbf{a}_t, \mathbf{s}_t)p^k(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})}{\mathbb{E}_{p^k(\mathbf{a}'_t|\mathbf{s}_t, \mathcal{O}_{t:T})}[p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}'_t, \mathcal{O}_{t:T})]}\right]
\end{aligned}
$$

A.3 Algorithm

Algorithm 1 Planning with Latent Trajectories

for t in $\{1, \dots, T\}$ do
  $p^0(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T}) \leftarrow \pi(\mathbf{a}_t|\mathbf{s}_t)$.
  // Perform Planning
  for k in $\{1, \dots, K\}$ do
    Sample $\mathbf{a}_t \sim p^{k-1}(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})$.
    // Sample future optimal trajectories.
    Sample $\hat{\mathbf{x}}_{t:t+h} \sim p(\hat{\mathbf{x}}_{t+1:t+h}|\mathbf{s}_t, \mathbf{a}_t, \mathcal{O}_{t:T})$.
    // Equation 4
    Compute $p^{k+1}(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})$.
  end for
  // Sample from the current approximation.
  $\mathbf{a}_t \sim p^K(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T})$.
  $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot|\mathbf{s}_t, \mathbf{a}_t)$
  Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$
  Update π, V and p_{model} with $\mathcal{B}$
end for