CRITIC SEQUENTIAL MONTE CARLO

Anonymous authors
Paper under double-blind review

ABSTRACT

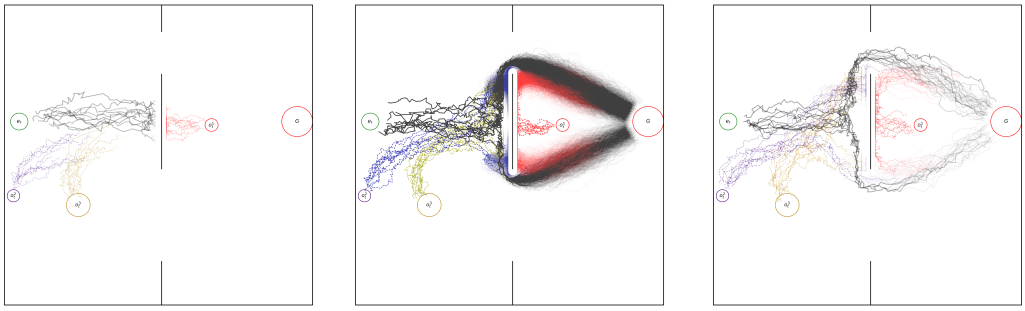
We introduce CriticSMC, a new algorithm for planning as inference built from a composition of sequential Monte Carlo with learned Soft-Q function heuristic factors. These heuristic factors, obtained from parametric approximations of the marginal likelihood ahead, more effectively guide SMC towards the desired target distribution, which is particularly helpful for planning in environments with hard constraints placed sparsely in time. Compared with previous work, we modify the placement of such heuristic factors, which allows us to cheaply propose and evaluate large numbers of putative action particles, greatly increasing inference and planning efficiency. CriticSMC is compatible with informative priors, whose density function need not be known, and can be used as a model-free control algorithm. Our experiments on collision avoidance in a high-dimensional simulated driving task show that CriticSMC significantly reduces collision rates at a low computational cost while maintaining realism and diversity of driving behaviors across vehicles and environment scenarios.

1 INTRODUCTION

Sequential Monte Carlo (SMC) (Gordon et al., 1993) is a popular, highly customizable inference algorithm that is well suited to posterior inference in state-space models (Arulampalam et al., 2002; Andrieu et al., 2004; Cappe et al., 2007). SMC is a form of importance sampling, which breaks down a high-dimensional sampling problem into a sequence of low-dimensional ones, making them tractable through repeated application of resampling. SMC in practice requires informative observations at each time step to be efficient when a finite number of particles is used. When observations are sparse, SMC loses its typical advantages and needs to be augmented with particle smoothing and backward messages to retain good performance (Kitagawa, 1994; Moral et al., 2009; Douc et al., 2011).

SMC can be applied to planning problems using the planning-as-inference framework (Ziebart et al., 2010; Neumann, 2011; Rawlik et al., 2012; Kappen et al., 2012; Levine, 2018; Abdolmaleki et al., 2018; Lavington et al., 2021). In this paper we are interested in solving planning problems with sparse, hard constraints, such as avoiding collisions while driving. In this setting, such a constraint is not violated until the collision occurs, but braking needs to occur well in advance to avoid it. Figure 1 demonstrates on a toy example how SMC requires excessive numbers of particles to solve such problems. In the language of optimal control (OC) and reinforcement learning (RL), collision avoidance is a *sparse reward problem*. In this setting, parametric estimators of future rewards (Nair et al., 2018; Riedmiller et al., 2018) are learned in order to alleviate the credit assignment problem (Sutton & Barto, 2018; Dulac-Arnold et al., 2021) and facilitate efficient learning.

In this paper we propose a novel formulation of SMC, called *CriticSMC*, where a learned critic, inspired by Q-functions in RL (Sutton & Barto, 2018), is used as a *heuristic factor* (Stuhlmüller et al., 2015) in SMC to ameliorate the problem of sparse observations. We borrow from the recent advances in deep-RL (Haarnoja et al., 2018a; Hessel et al., 2018) to learn a critic which approximates future likelihoods in a parametric form. While similar ideas have been proposed in the past (Rawlik et al., 2012; Piché et al., 2019), in this paper we instead suggest (1) using soft Q-functions (Rawlik et al., 2012; Chan et al., 2021; Lavington et al., 2021) as heuristic factors, and (2) choosing the placement of such factors to allow for efficient exploration of action-space through the use of putative particles (Fearnhead, 2004). Additionally, we design CriticSMC to be compatible with informative prior distributions, which may not include an associated (known) log-density function. In planning contexts, such priors can specify additional requirements that may be difficult to define via rewards, such as maintaining human-like driving behavior.



(a) SMC (10 particles) (b) SMC (10k particles) (c) CriticSMC (10 particles)

Figure 1: Illustration of the difference between CriticSMC and SMC in a toy environment in which a green ego agent is trying to reach the red goal without being hit by any of the three chasing adversaries. All plots show overlaid samples of environment trajectories conditioned on the ego agent achieving its goal. While SMC will asymptotically explore the whole space of environment trajectories, CriticSMC's method of using the critic as a heuristic within SMC encourages computationally efficient discovery of diverse high reward trajectories. SMC with a small number of particles fails here because the reward is sparse and the ego agent's prior behavioral policy assigns low probability to trajectories that avoid the barrier and the other agents.

We show experimentally that CriticSMC is able to refine the policy of a foundation (Bommasani et al., 2021) autonomous-driving behavior model to take actions that produce significantly fewer collisions while retaining key behavioral distribution characteristics of the foundation model. This is important not only for the eventual goal of learning complete autonomous driving policies (Jain et al., 2021; Hawke et al., 2021), but also immediately for constructing realistic infraction-free simulations to be employed by autonomous vehicle controllers (Suo et al., 2021; Bergamini et al., 2021; Ścibior et al., 2021; Lioutas et al., 2022) for training and testing. Planning, either in simulation or real world, requires a model of the world (Ha & Schmidhuber, 2018). While CriticSMC can act as a planner in this context, we show that it can just as easily be used for model-free online control without a world model. This is done by densely sampling putative action particles and using the critic to select amongst these sampled actions. We also provide ablation studies which demonstrate that the two key components of CriticSMC, namely the use of the soft Q-functions and putative action particles, significantly improve performance over relevant baselines with similar computational resources.

2 Preliminaries

Since we are primarily concerned with planning problems, we work within the framework of Markov decision processes (MDPs). An MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{P}_0, \mathcal{R}, \Pi\}$ is defined by a set of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, reward function $r(s, a, s') \in \mathcal{R}$, transition dynamics distribution $p(s'|a, s) \in \mathcal{P}$, initial state distribution $p_0(s) \in \mathcal{P}_0$, and policy distribution $\pi(a|s) \in \Pi$. Trajectories are generated by first sampling from the initial state distribution $s_1 \sim p_0$, then sequentially sampling from the policy $a_t \sim \pi(a_t|s_t)$ and then the transition dynamics $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ for T-1 time steps. Execution of this stochastic process produces a trajectory $\tau = \{(s_1, a_1), \ldots, (s_T, a_T)\} \sim p^\pi$, which is then scored using the reward function r. The goal in RL and OC is to produce a policy $\pi^* = \arg\max_\pi \mathbb{E}_{p^\pi}[\sum_{t=1}^{T} r(s_t, a_t, s_{t+1})]$. We now relate this stochastic process to inference.

2.1 Reinforcement Learning as Inference

RL-as-inference (RLAI) considers the relationship between RL and approximate posterior inference to produce a class of divergence minimization algorithms able to estimate the optimal RL policy. The posterior we target is defined by a set of observed random variables $O_{1:T}$ and latent random variables $\tau_{1:T}$. Here, O defines "optimality" random variables which are Bernoulli distributed with probability proportional to exponentiated reward values (Ziebart et al., 2010; Neumann, 2011; Levine, 2018). They determine whether an individual tuple $\tau_t = \{s_t, a_t, s_{t+1}\}$ is optimal ($O_t = 1$) or sub-optimal

1: **sampler** SIMPLE	1: **sampler** H-FACTORS	1: **sampler** CRITICSMC
2: $a_t^i \sim \pi(a_t \mid s_t^i)$	2: $a_t^i \sim \pi(a_t \mid s_t^i)$	2: $a_t^i \sim \pi(a_t \mid s_t^i)$
3: $\hat{s}_{t+1}^i \sim \boldsymbol{p(s_{t+1} \mid s_t^i, a_t^i)}$	3: $\hat{s}_{t+1}^i \sim \boldsymbol{p(s_{t+1} \mid s_t^i, a_t^i)}$	3:
4: $\hat{w}_t^i \leftarrow \bar{w}_{t-1}^i e^{\boldsymbol{r(s_t^i, a_t^i, \hat{s}_{t+1}^i)}}$	4: $\hat{w}_t^i \leftarrow \bar{w}_{t-1}^i e^{\boldsymbol{r(s_t^i, a_t^i, \hat{s}_{t+1}^i) + h_t^i}}$	4: $\hat{w}_t^i \leftarrow \bar{w}_{t-1}^i e^{\boldsymbol{h_t^i}}$
5: $\alpha_t^i \sim$ RESAMPLE$(\hat{w}_t^{1:N})$	5: $\alpha_t^i \sim$ RESAMPLE$(\hat{w}_t^{1:N})$	5: $\alpha_t^i \sim$ RESAMPLE$(\hat{w}_t^{1:N})$
6: $s_{t+1}^i \leftarrow \hat{s}_{t+1}^{\alpha_t^i}$	6: $s_{t+1}^i \leftarrow \hat{s}_{t+1}^{\alpha_t^i}$	6: $s_{t+1}^i \leftarrow \boldsymbol{p(s_{t+1} \mid s_t^{\alpha_t^i}, a_t^{\alpha_t^i})}$
7: $\bar{w}_t^i \leftarrow \frac{1}{N}$	7: $\bar{w}_t^i \leftarrow \frac{1}{N} e^{-\boldsymbol{h_t^{\alpha_t^i}}}$	7: $\bar{w}_t^i \leftarrow \frac{1}{N} e^{\boldsymbol{r(s_t^{\alpha_t^i}, a_t^{\alpha_t^i}, s_{t+1}^i) - h_t^{\alpha_t^i}}}$
8: **end sampler**	8: **end sampler**	8: **end sampler**
(a) No heuristic factors	(b) With heuristic factors	(c) CriticSMC version

Figure 2: Main loop of SMC without heuristic factors (left), with a naive heuristic factors h_t (middle) and with the placement we use in CriticSMC (right). We use $\hat{w}$ for pre-resampling weights and $\bar{w}$ for post-resampling weights and we elide the normalizing factor $W_t = \sum_{i=1}^N \hat{w}_t^i$ for clarity. The placement of h_t in CriticSMC crucially enables using putative action particles in Section 3.3.

($O_t = 0$). We replace $O_t = 1$ with O_t in the remainder of the paper for conciseness. While we can rarely compute the posterior $p(s_{1:T}, a_{1:T} \mid O_{1:T})$ in closed form, we assume the joint distribution

$$p(s_{1:T}, a_{1:T}, O_{1:T}) = p_0(s_1) \prod_{t=1}^{T} p(O_t \mid s_t, a_t, s_{t+1}) p(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t). \tag{1}$$

This joint distribution can be used following standard procedures from variational inference to learn or estimate the posterior distribution of interest (Kingma & Welling, 2014). How close the estimated policy is to the optimal policy often depends upon the chosen reward surface, the prior distribution over actions, and chosen policy distribution class. Generally, the prior is chosen to contain minimal information in order to maximize the entropy of the resulting approximate posterior distribution (Ziebart et al., 2010; Haarnoja et al., 2018a). Contrary to classical RL, we are interested in using informative priors whose attributes we want to preserve while maximizing the expected reward ahead. In order to manage this trade-off, we now consider more general inference algorithms for state-space models.

2.2 SEQUENTIAL MONTE-CARLO

SMC (Gordon et al., 1993) is a popular algorithm which can be used to sample from the posterior distribution in non-linear state-space models and HMMs. In RLAI, SMC sequentially approximates the filtering distributions $p(s_t, a_t \mid O_{1:t})$ for $t \in 1 \ldots T$ using a collection of weighted samples called particles. The crucial resampling step adaptively focuses computation on the most promising particles while still producing an unbiased estimation of the marginal likelihood (Moral, 2004; Chopin et al., 2012; Pitt et al., 2012; Naesseth et al., 2014; Le, 2017). The primary sampling loop for SMC in a Markov decision process is provided in Figure 2a, and proceeds by sampling an action a_t given a state s_t, generating a sample of the **next state s_{t+1}** using a model of the world, computing a weight $\hat{w}_t$ using the **reward function r**, and resampling from this particle population. The post-resampling weights $\bar{w}_t$ are assumed to be uniform for simplicity but non-uniform resampling schemes exist (Fearnhead & Clifford, 2003). Here, each timestep only performs simple importance sampling linking the posterior $p(s_t, a_t \mid O_{1:t})$ to $p(s_{t+1}, a_{t+1} \mid O_{1:t+1})$. When the observed likelihood information is insufficient, the particles may fail to cover the portion of the space required to approximate the next posterior timestep. For example, if all current particles have the vehicle moving at high speed towards the obstacle, it may be too late to brake and causing SMC to erroneously conclude that a collision was inevitable, while in fact it just did not explore braking actions earlier on in time.

As shown by Stuhlmüller et al. (2015), we can introduce arbitrary **heuristic factors h_t** into SMC before resampling, as shown in Figure 2b, mitigating the insufficient observed likelihood information. h_t can be a function of anything sampled up to the point where it is introduced, does not alter the asymptotic behavior of SMC, and can dramatically improve finite sample efficiency if chosen carefully. In this setting, the optimal choice for h_t is the marginal log-likelihood ahead $\sum_t^T \log p(O_{t:T} \mid s_t, a_t)$,

which is typically intractable to compute but can be approximated. In the context of avoiding collisions, this term estimates the likelihood of future collisions from a given state. A typical application of such heuristic factors in RLAI, as given by Piché et al. (2019), is shown in Figure 2b.

3 CRITICSMC

Historically, heuristic factors in SMC are placed alongside the reward, which is computed by taking a single step in the environment (Figure 2b). The crucial issue with this methodology is that updating weights requires computing the next state (Line 3 in Figure 2b), which can both be expensive in complex simulators, and would prevent the use in online control without a world model. In order to avoid this issue while maintaining the advantages of SMC with heuristic factors, we propose to score particles using only the heuristic factor, resample, then compute the next state and the reward, as shown in Figure 2c. We choose h_t which only depend on the previous state and actions observed, and not the future state, so that we can sample and score a significantly larger number of so-called putative action particles, thereby increasing the likelihood of sampling particles with a large h_t. In this section we first show how to construct such h_t, then how to learn an approximation to it, and finally how to take full advantage of this sampling procedure using putative action particles.

3.1 FUTURE LIKELIHOODS AS HEURISTIC FACTORS

We consider environments where planning is needed to satisfy certain hard constraints $C(s_t)$ and define the violations of such constraints as *infractions*. This makes the reward function (and thus the log-likelihood) defined in Section 2 sparse,

$$\log p(O_t|s_t, a_t, s_{t+1}) = r(s_t, a_t, s_{t+1}) = \begin{cases} 0, & \text{if } C(s_{t+1}) \text{ is satisfied} \\ -\beta_{\text{pen}}, & \text{otherwise} \end{cases} \tag{2}$$

where $\beta_{\text{pen}} > 0$ is a penalty coefficient. At every time-step, the agent receives a reward signal indicating if an infraction occurred (e.g. there was a collision). To guide SMC particles towards states that are more likely to avoid infractions in the future, we use h_t which approximate *future likelihoods* (Kim et al., 2020) defined as $h_t \approx \log p(O_{t:T}|s_t, a_t)$. Such heuristic factors up-weight particles proportionally to how likely they are to avoid infractions in the future but can be difficult to accurately estimate in practice.

As has been shown in previous work (Rawlik et al., 2012; Levine, 2018; Piché et al., 2019; Lavington et al., 2021), $\log p(O_{t:T}|s_t, a_t)$ corresponds to the "soft" version of the state-action value function $Q(s_t, a_t)$ used in RL, often called the *critic*. Following Levine (2018), we use the same symbol Q for the soft-critic. Under deterministic state transitions, the soft Q function satisfies the following equation, which follows from the exponential definition of the reward given in Equation 2 (a proof is provided in Section A.2 of the Appendix),

$$Q(s_t, a_t) := \log p(O_{t:T}|s_t, a_t) = r(s_t, a_t, s_{t+1}) + \log \mathop{\mathbb{E}}_{a_{t+1}\sim\pi(a_{t+1}|s_{t+1})} \left[e^{Q(s_{t+1}, a_{t+1})} \right]. \tag{3}$$

CriticSMC sets the heuristic factor $h_t = Q(s_t, a_t)$, as shown in Figure 2c. We note that alternatively one could use the state value function for the next state $V(s_{t+1}) = \log \mathbb{E}_{a_{t+1}}[\exp(Q(a_{t+1}, s_{t+1}))]$, as shown in Figure 2b. This would be equivalent to the SMC algorithm of Piché et al. (2019) (see Section A.2 of the Appendix), which was originally derived using the two-filter formula (Bresler, 1986; Kitagawa, 1994) instead of heuristic factors. The primary advantage of the CriticSMC formulation is that the heuristic factor can be computed before the next state, thus allowing the application of putative action particles.

3.2 LEARNING CRITIC MODELS WITH SOFT Q-LEARNING

Because we do not have direct access to Q, we estimate it parametrically with Q_ϕ. Equation 3 suggests the following training objective for learning the state-action critic (Lavington et al., 2021)

$$\begin{aligned} \mathcal{L}_{\text{TD}}(\phi) &= \mathop{\mathbb{E}}_{s_t, a_t, s_{t+1}\sim d_{\text{SAO}}} \left[\left(Q_\phi(s_t, a_t) - r(s_t, a_t, s_{t+1}) - \mathop{\log\,\mathbb{E}}_{a_{t+1}\sim\pi(a_{t+1}|s_{t+1})} \left[e^{Q_{\perp(\phi)}(s_{t+1}, a_{t+1})} \right] \right)^2 \right] \\ &\approx \mathop{\mathbb{E}}_{s_t, a_t, s_{t+1}\sim d_{\text{SAO}}} \left[\mathop{\mathbb{E}}_{a_{t+1}^{1:K}\sim\pi(a_{t+1}|s_{t+1})} \left[\left(Q_\phi(s_t, a_t) - \tilde{Q}_{\text{TA}}(s_t, a_t, s_{t+1}, \hat{a}_{t+1}^{1:K}) \right)^2 \right] \right], \end{aligned} \tag{4}$$

where d_{SAO} is the *state-action occupancy* (SAO) induced by CriticSMC, $\perp$ is the stop-gradient operator (Foerster et al., 2018) indicating that the gradient of the enclosed term is discarded, and the approximate target value $\tilde{Q}_{\text{TA}}$ is defined as

$$\tilde{Q}_{\text{TA}}(s_t, a_t, s_{t+1}, \hat{a}_{t+1}^{1:K}) = r(s_t, a_t, s_{t+1}) + \gamma \log \frac{1}{K} \sum_{j=1}^{K} e^{Q_{\perp(\phi)}(s_{t+1}, \hat{a}_{t+1}^{j})}. \tag{5}$$

The discount factor γ is introduced to reduce variance and improve the convergence of Soft-Q iteration (Bertsekas, 2019; Chan et al., 2021). For stability, we replace the bootstrap term $Q_{\perp(\phi)}$ with a ϕ-averaging target network Q_ψ (Lillicrap et al., 2016), and use prioritized experience replay (Schaul et al., 2016), a non-uniform sampling procedure. These modifications are standard in deep RL, and help improve stability and convergence of the trained critic (Hessel et al., 2018). We note that unlike Piché et al. (2019), we learn the soft-Q function for the (static) prior policy, dramatically simplifying the training process, and allowing faster sampling at inference time.

3.3 Putative Action Particles

Sampling actions given states is often computationally cheap when compared to sampling states following transition dynamics. Even when a large model is used to define the prior policy, it is typically structured such that the bulk of the computation is spent processing the state information and then a relatively small probabilistic head can be used to sample many actions. To take advantage of this, we temporarily increase the particle population size K-fold when sampling actions and then reduce it by resampling before the new state is computed. This is enabled by the placement of heuristic factors between sampling the action and computing the next state, as highlighted in Figure 2c. Specifically, at each time step t for each particle i we sample K actions $\hat{a}_t^{i,j}$, resulting in $N{\cdot}K$ putative action particles (Fearnhead, 2004). The critic is then applied as a heuristic factor to each putative particle, and a population of size N re-sampled following the next time-step using these weighted examples. The full algorithm is given in Algorithm 1.

Algorithm 1 Critic Sequential Monte Carlo

procedure CRITICSMC(p_0, p, π, r, Q N, K, T)
  Sample $s_1^{1:N} \sim p_0(s_1)$
  Set $\bar{w}_0^{1:N} \leftarrow \frac{1}{N}$
  for $t \in 1 \dots T$ **do**
    for $n \in 1 \dots N$ **do**
      for $k \in 1 \dots K$ **do**
        Sample $\hat{a}_t^{n,k} \sim \pi(a_t | s_t^n)$
        Set $\hat{w}_t^{n \cdot N + k} \leftarrow \frac{1}{K} \bar{w}_{t-1}^n e^{Q(s_t^n, \hat{a}_t^{n,k})}$
      end for
    end for
    Set $W_t \leftarrow \sum_{i=1}^{N \cdot K} \hat{w}_t^i$
    Sample $\alpha_t^{1:N} \sim$ RESAMPLE $\left(\frac{\hat{w}_t^{1:N \cdot K}}{W_t}\right)$
    for $n \in 1 \dots N$ **do**
      Set $i \leftarrow \lfloor \alpha_t^n / K \rfloor + 1$
      Set $j \leftarrow (\alpha_t^n \bmod K) + 1$
      Set $a_t^n \leftarrow \hat{a}_t^{i,j}$
      Sample $s_{t+1}^n \sim p(s_{t+1} | s_t^i, \hat{a}_t^{i,j})$
      Set $\bar{w}_t^n \leftarrow \frac{1}{N} W_t e^{r(s_t^i, \hat{a}_t^{i,j}, s_{t+1}^n) - Q(s_t^i, \hat{a}_t^{i,j})}$
    end for
  end for
  return $s_{1:T}^{1:N}, a_{1:T}^{1:N}, \bar{w}_{1:T}^{1:N}$
end procedure

For low dimensional action spaces, it is possible to sample actions densely under the prior, eliminating the need for a separate proposal distribution. This is particularly beneficial in settings where the prior policy is only defined implicitly by a sampler and its log-density cannot be quickly evaluated everywhere. In the autonomous driving context, the decision leading to certain actions can be complex, but the action space is only two- or three-dimensional. Using CriticSMC, a prior generating human-like actions can be provided as a sampler without the need for a density function. Lastly, CriticSMC can be used for model-free online control through sampling putative actions from the current state, applying the critic, and then selecting a single action through resampling. This can be regarded as a prior-aware approach to selecting actions similar to algorithms proposed by Abdolmaleki et al. (2018); Song et al. (2019).

4 Experiments

We demonstrate the effectiveness of CriticSMC for probabilistic planning in two environments: a multi-agent point-mass toy environment and a high-dimensional driving simulator. In both environments infractions are defined as collisions with either other agents or the walls. Since the environment dynamics are known and deterministic, we do not learn a state transition model of the world and there is no need to re-plan actions in subsequent time steps. We also show that CriticSMC successfully

avoids collisions in the driving environment when deployed in a model-free fashion. Finally, we show that both the use of putative particles and the Soft-Q function instead of the standard Hard-Q result in significant improvement in terms of reducing infractions and maintaining behavior close to the prior.

4.1 Toy Environment

In the toy environment, depicted in Figure 1, the prior policy is a Gaussian random walk towards the the goal position without any information about the position of the other agents and the barrier. All external agents are randomly placed, and move adversarially towards the ego agent with some added Gaussian noise. The ego agent commits an infraction if any of the following is true: 1) colliding with any of the other agents, 2) hitting a wall, 3) moving outside the perimeter of the environment. Details of this environment can be found in the Appendix.

We compare CriticSMC using 50 particles and 1024 putative action particles on the planning task against several baselines, namely the prior policy, rejection sampling with 1000 maximum trials, and the SMC method of Piché et al. (2019) with 50 particles. We randomly select 500 episodes with different initial conditions and perform 6 independent rollouts for each episode. The **prior policy** has an infraction rate of **0.86**, **rejection sampling** achieves **0.71** and **SMC of Piché et al. (2019)** yields an infraction rate of **0.11**. **CriticSMC** reduces infraction rate to **0**.

4.2 Human-like Driving Behavior Modeling

Human-like driving behavior models are increasingly used to build realistic simulation for training self-driving vehicles (Suo et al., 2021; Bergamini et al., 2021; Ścibior et al., 2021), but they tend to suffer from excessive numbers of infractions, in particular collisions. In this experiment we take an existing model of human driving behavior, ITRA (Ścibior et al., 2021), as the prior policy and attempt to avoid collisions, while maintaining the human-likeness of predictions as much as possible. The environment features non-ego agents, for which we replay actions as recorded in the INTERACTION dataset (Zhan et al., 2019). The critic receives a stack of the last two ego-centric ego-rotated birdview images (Figure 3) of size $256 \times 256 \times 3$ as partial observations of the full state. This constitutes a challenging, image-based, high-dimensional continuous control environment, in contrast to Piché et al. (2019), who apply their algorithm to low dimensional vector-state spaces in the Mujoco simulator Todorov et al. (2012); Brockman et al. (2016). The key performance metric in this experiment is the average collision rate, but we also report the average displacement error (ADE_6) using the minimum error across six samples for each prediction, which serves as a measure of human-likeness. Finally, the maximum final distance (MFD) metric is reported to measure the diversity of the predictions. The evaluation is performed using the validation split of the INTERACTION dataset, which neither ITRA nor the critic saw during training.

We evaluate CriticSMC on a model-based planning task against the following baselines: the prior policy (ITRA), rejection sampling with 5 maximum trials and the SMC incremental weight update rule proposed by Piché et al. (2019) using 5 particles. CriticSMC uses 5 particles and 128 putative particles, noting the computational cost of using the putative particles is negligible. We perform separate evaluation in each of four locations from the INTERACTION dataset, and for each example in the validation set we execute each method six times independently to compute the performance metrics. Table 1 shows that CriticSMC reduces the collision rate substantially more than any of the baselines and that it suffers a smaller decrease in predictive error than the SMC of Piché et al. (2019). All methods are able to maintain diversity of sampled trajectories on par with the prior policy.

Next we test CriticSMC as a model-free control method, not allowing it to interact with the environment until an action for a given time step is selected, which is equivalent to using a single particle in CriticSMC. Specifically, at each step we sample 128 putative action particles and resample one of them based on critic evaluation as a heuristic factor. We use Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) as a model-free baseline, noting that other SMC variants are not applicable in this context, since they require inspecting the next state to perform resampling. We show in Table 2 that CriticSMC is able to reduce collisions without sacrificing realism and diversity in the predictions. Here SAC does notably worse in terms of both collision rate as well as ADE. This is unsurprising as CriticSMC takes advantage of samples from the prior, which is already performant in both metrics, while SAC must be trained from scratch. This example highlights how CriticSMC utilizes prior information more efficiently than black-box RL algorithms like SAC.

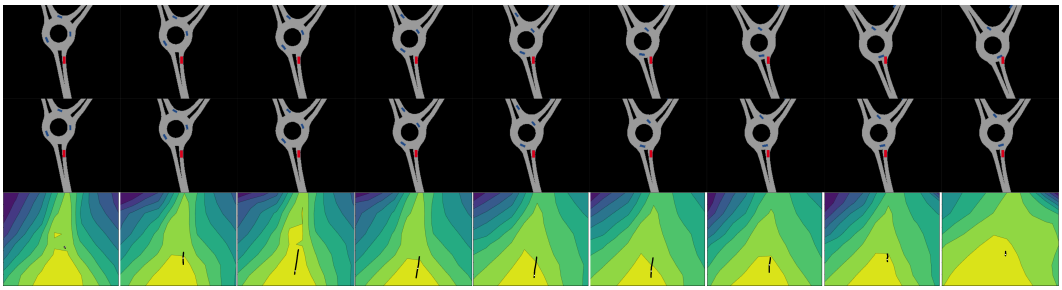


Figure 3: Collision avoidance arising from using CriticSMC for control of the red ego agent in a scenario from the INTERACTION dataset. There are three rows: the top shows the sequence of states leading to a collision arising from choosing actions from the prior policy, the middle row shows that control by CriticSMC's implicit policy avoids the collision, and the third row is a contour plot illustrating the relative values of the critic (brighter corresponds to higher expected reward) evaluated at the current state over the entire action space of acceleration (vertical axis) and steering (horizontal axis). The black dots are 128 actions sampled from the prior policy. The white dot indicates the selected action. Best viewed zoomed onscreen. For more examples see Figure 5 in the Appendix.

Table 1: Infraction rates for different inference methods performing model-predictive planning tested on four locations from the INTERACTION dataset (Zhan et al., 2019).

Location	Method	Collision Infraction Rate	MFD	ADE_6
DR_DEU_Merging_MT	Prior	0.02522	2.2038	0.3024
	Rejection Sampling	0.01758	2.3578	0.3071
	SMC by Piché et al. (2019)	0.02191	2.3388	0.4817
	CriticSMC	**0.01032**	2.2009	0.3448
DR_USA_Intersection_MA	Prior	0.00874	3.1369	0.3969
	Rejection Sampling	0.00218	3.2100	0.3908
	SMC by Piché et al. (2019)	0.00351	2.8490	0.4622
	CriticSMC	**0.00085**	2.8713	0.4479
DR_USA_Roundabout_FT	Prior	0.00583	3.1004	0.4080
	Rejection Sampling	0.00133	3.0211	0.4046
	SMC by Piché et al. (2019)	0.00166	3.0086	0.4814
	CriticSMC	**0.00066**	2.9736	0.4439
DR_DEU_Roundabout_OF	Prior	0.00583	3.5536	0.4389
	Rejection Sampling	0.00216	3.4992	0.4287
	SMC by Piché et al. (2019)	0.00342	3.2836	0.5701
	CriticSMC	**0.00083**	3.4248	0.4450

4.3 Method Ablation

Effect of Using Putative Action Particles We evaluate the importance of putative action particles, via an ablation study varying the number of particles and putative particles in CriticSMC and standard SMC. Table 3 contains results that show both increasing the number of particles and putative articles has significant impact on performance. Putative particles are particularly important, since a large number of them can typically be generated with a small computational overhead.

Comparison of Training the Critic With the Soft-Q and Hard-Q Objective Next we compare the fitted Q iteration (Watkins & Dayan, 1992), which uses the maximum over Q at the next stage to update the critic (i.e., $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$), with the fitted soft-Q iteration used by CriticSMC (Eq. 4). The results, displayed in Table 4, show that the Hard-Q heuristic factor leads to a significant reduction in collision rate over the prior, but produces a significantly higher ADE_6 score. We attribute this to the risk-avoiding behavior induced by hard-Q.

Table 2: Infraction rates for performing model-free online control against the prior and SAC policies tested on four locations from the INTERACTION dataset (Zhan et al., 2019).

Location	Method	Collision Infraction Rate	MFD	ADE_6
DR_DEU_Merging_MT	Prior	0.02522	2.2038	0.3024
	SAC	0.03899	0.0	1.1548
	CriticSMC	**0.01376**	2.1985	0.3665
DR_USA_Intersection_MA	Prior	0.00874	3.1369	0.3969
	SAC	0.02700	0.0	4.1141
	CriticSMC	**0.00285**	2.9595	0.4641
DR_USA_Roundabout_FT	Prior	0.00583	3.1004	0.4080
	SAC	0.04501	0.0	1.7987
	CriticSMC	**0.00183**	3.0125	0.4567
DR_DEU_Roundabout_OF	Prior	0.00583	3.5536	0.4389
	SAC	0.06400	0.0	3.4583
	CriticSMC	**0.00233**	3.5173	0.4459

Table 3: Infraction rates for SMC and CriticSMC with a varying the number of particles and putative particles, tested on 500 random initial states using the proposed toy environment.

Method	Putative Particles	Particles				
		1	5	10	20	50
SMC	1	0.776	0.6092	0.4576	0.3178	0.1836
CriticSMC	1	0.776	0.368	0.1346	0.0334	0.0106
SMC	1024	0.578	0.3664	0.237	0.1568	0.0904
CriticSMC	1024	0.048	0.01	0.002	0.002	**0.0**

Execution Time Comparison Figure 4 shows the average execution time it takes to predict 3 seconds into the future given 1 second of historical observations for the driving behavior modeling experiment. This shows that the run-time of all algorithms is of the same order, while the collision rate of CriticSMC is significantly lower, demonstrating the low overhead of using putative action particles.

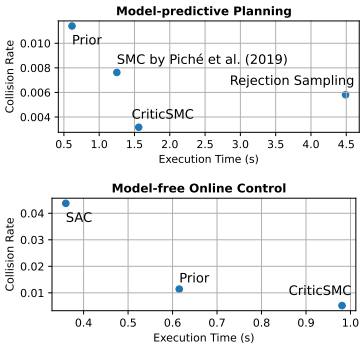


Figure 4: Execution time comparison between the baseline methods and CriticSMC for both model-predictive planning and model-free online control. The collision infraction rate is averaged across all four INTERACTION locations.

5 Related Work

SMC methods (Gordon et al., 1993; Kitagawa, 1996; Liu & Chen, 1998), also known as particle filters, are a well-established family of inference methods for generating samples from posterior distributions. Their basic formulations perform well on the filtering task, but poorly on smoothing (Godsill et al., 2004) due to particle degeneracy. These issues are usually addressed using backward simulation (Lindsten & Schön, 2013) or rejuvenation moves (Gilks & Berzuini, 2001; Andrieu et al., 2010). These solutions improve sample diversity but are not sufficient in our context, where normal SMC often fails to find even a single infraction-free sample. Similarly to CriticSMC, Piché et al. (2019) propose using the value function as a backward message in SMC for planning. Their method is equivalent to what is obtained using the equations from Figure 2b with $h_t = V(s_{t+1}) = \log \mathbb{E}_{a_{t+1}}[\exp(Q(a_{t+1}, s_{t+1}))]$ (see proof in Section A.2 of the Appendix). This formulation cannot accommodate putative action particles and learns a parametric policy alongside $V(s_{t+1})$, instead of the applying the soft Bellman update (Asadi & Littman, 2017; Chan et al., 2021) to a fixed prior.

In our experiments we used the bootstrap proposal (Gordon et al., 1993), which samples from the prior model, but in cases where the prior density can be efficiently computed, using a better proposal

Table 4: Infraction rates for the Hard-Q and the Soft-Q objectives tested on the location DR_DEU_Merging_MT from the INTERACTION dataset (Zhan et al., 2019).

Method	Critic Objective	Collision Infraction Rate	MFD	ADE_6	Progress
Prior	-	0.02522	2.2038	0.3024	16.43
CriticSMC	Hard-Q	0.01911	0.9383	1.0385	14.98
	Soft-Q	**0.01376**	2.1985	0.3665	15.91

distribution can bring significant improvements. Such proposals can be obtained in a variety of ways, including using unscented Kalman filters (van der Merwe et al., 2000) or neural networks minimizing the forward Kullback-Leibler divergence (Gu et al., 2015). CriticSMC can accommodate proposal distributions, but even when the exact smoothing distribution is used as a proposal, backward messages are still needed to avoid the particle populations that focus on the filtering distribution.

As we show in this work, CriticSMC can be used for planning as well as model-free online control. The policy it defines in the latter case is not parameterized explicitly, but rather obtained by combining the prior and the critic. This is similar to classical Q-learning (Watkins & Dayan, 1992), which obtains the implicit policy by taking the maximum over all actions of the Q function in discrete action spaces. This approach has been extended to continuous domains using ensembles (Deisenroth & Rasmussen, 2011; Ryu et al., 2020; Lee et al., 2021) and quantile networks (Bellemare et al., 2017). The model-free version of CriticSMC is also very similar to soft Q-learning described as described by Haarnoja et al. (2017); Abdolmaleki et al. (2018), and analyzed by Chan et al. (2021).

Imitating human driving behavior has been successful in learning control policies for autonomous vehicles (Bojarski et al., 2016; Hawke et al., 2019) and to generate realistic simulations (Bergamini et al., 2021; Ścibior et al., 2021). In both cases, minimizing collisions, continues to present one of the most important issues in autonomous vehicle research. Following a data-driven approach, Suo et al. (2021) proposed auxiliary losses for collision avoidance, while Igl et al. (2022) used adversarially trained discriminators to prune predictions that are likely to result in infractions. To the best of our knowledge, ours is the first work to apply a critic targeting the backward message in this context.

6 Discussion

CriticSMC increases the efficiency of SMC for planning in scenarios with hard constraints, when the actions sampled must be adjusted long before the infraction takes place. It achieves this efficiency through the use of a learned critic which approximates the future liklihood using putative particles that densely sample the action space. The performance of CriticSMC relies heavily on the quality of the critic and in this work we display how to take advantage of recent advances in deep RL to obtain one. One avenue for future work is devising more efficient algorithms for learning the soft Q function such as proximal updates (Schulman et al., 2017) or the inclusion of regularization which guards against deterioration of performance late in training (Kumar et al., 2020).

The design of CriticSMC is motivated by the desire to accommodate implicit priors defined as samplers, such as the ITRA model (Ścibior et al., 2021) we used in our self-driving experiments. For this reason we avoided learning explicit policies to use as proposal distributions, since maintaining similarity with the prior can be extremely complicated. Where the prior density can be computed, learned policies could be successfully accommodated. This is particularly important when the action space is high-dimensional and it is difficult to sample it densely using putative particles.

In this work we focused on environments with deterministic transition dynamics, but CriticSMC could also be applied when dynamics are stochastic. In these settings, the planning as inference framework suffers from optimism bias (Levine, 2018; Chan et al., 2021), even when exact posterior can be computed, which is usually mitigated by carefully constructing the variational family. For applications in real world planning, CriticSMC relies on having a model of transition dynamics and the fidelity of that model is crucial for achieving good performance. Learning such models from observations is an active area of research (Ha & Schmidhuber, 2018; Chua et al., 2018; Nagabandi et al., 2020). Finally, we focused on avoiding infractions, but CriticSMC is applicable to planning with any reward surfaces and to sequential inference problems more generally.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Christophe Andrieu, A. Doucet, Sumeetpal S. Singh, and Vladislav Z. B. Tadić. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 92(3):423–438, 2004. doi: 10.1109/JPROC.2003.823142.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2009.00736.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2009.00736.x. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2009.00736.x.

M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50 (2):174–188, 2002. doi: 10.1109/78.978374.

Kavosh Asadi and Michael L. Littman. An alternative softmax operator for reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 243–252. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/asadi17a.html.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Blazej Osinski, Hugo Grimmett, and Peter Ondruska. SimNet: Learning Reactive Self-driving Simulations from Real-world Observations. *arXiv:2105.12332 [cs]*, May 2021. URL http://arxiv.org/abs/2105.12332. arXiv: 2105.12332.

Dimitri Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. Technical Report arXiv:1604.07316, arXiv, April 2016. URL http://arxiv.org/abs/1604.07316. arXiv:1604.07316 [cs] type: article.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang,

Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, abs/2108.07258, 2021.

Yoram Bresler. Two-filter formulae for discrete-time non-linear bayesian smoothing. *International Journal of Control*, 43(2):629–641, 1986. doi: 10.1080/00207178608933489. URL https://doi.org/10.1080/00207178608933489.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016. URL http://arxiv.org/abs/1606.01540. arXiv: 1606.01540.

Olivier Cappe, Simon J. Godsill, and Eric Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007. doi: 10.1109/JPROC.2007.893250.

Alan Chan, Hugo Silva, Sungsu Lim, Tadashi Kozuno, A Rupam Mahmood, and Martha White. Greedification operators for policy optimization: Investigating forward and reverse kl divergences. *arXiv preprint arXiv:2107.08285*, 2021.

Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos. Smc2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, Oct 2012. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2012.01046.x. URL http://dx.doi.org/10.1111/j.1467-9868.2012.01046.x.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472. Citeseer, 2011.

R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, 2005. ISSN 1845-5921. doi: 10.1109/ispa.2005.195385. URL http://dx.doi.org/10.1109/ISPA.2005.195385.

Randal Douc, Aurélien Garivier, Eric Moulines, and Jimmy Olsson. Sequential Monte Carlo smoothing for general state space hidden Markov models. *The Annals of Applied Probability*, 21(6):2109 – 2145, 2011. doi: 10.1214/10-AAP735. URL https://doi.org/10.1214/10-AAP735.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Paul Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, January 2004. ISSN 0960-3174. doi: 10.1023/B:STCO.0000009418.04621.cd.

Paul Fearnhead and Peter Clifford. On-line inference for hidden markov models via particle filters. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 65(4):887–899, 2003. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3647589.

Jakob N. Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric P. Xing, and Shimon Whiteson. DiCE: The Infinitely Differentiable Monte Carlo Estimator. In *ICML*, pp. 1524–1533, 2018. URL http://proceedings.mlr.press/v80/foerster18a.html.

Walter R. Gilks and Carlo Berzuini. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001. ISSN 1467-9868. doi: 10.1111/1467-9868.00280. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00280. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00280.

Simon J. Godsill, Arnaud Doucet, and Mike West. Monte carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, 2004. ISSN 01621459. URL http://www.jstor.org/stable/27590362.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113(6), April 1993. ISSN 0956-375X. URL https://digital-library.theiet.org/content/journals/10.1049/ip-f-2.1993.0015.

Shixiang Gu, Zoubin Ghahramani, and Richard E. Turner. Neural Adaptive Sequential Monte Carlo. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2629–2637, Cambridge, MA, USA, 2015. MIT Press.

David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/haarnoja17a.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018a. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemyslaw Mazur, Sean Micklethwaite, Nicolas Griffiths, Amar Shah, and Alex Kendall. Urban Driving with Conditional Imitation Learning. Technical Report arXiv:1912.00177, arXiv, December 2019. URL http://arxiv.org/abs/1912.00177. arXiv:1912.00177 [cs] type: article.

Jeffrey Hawke, E Haibo, Vijay Badrinarayanan, and Alex Kendall. Reimagining an autonomous vehicle. *ArXiv*, abs/2108.05805, 2021.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning Realistic and Diverse Agents for Autonomous Driving Simulation. Technical Report arXiv:2205.03195, arXiv, May 2022. URL http://arxiv.org/abs/2205.03195. arXiv:2205.03195 [cs] type: article.

Ashesh Jain, Luca Del Pero, Hugo Grimmett, and Peter Ondruska. Autonomy 2.0: Why is self-driving always 5 years away? *ArXiv*, abs/2107.08142, 2021.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Geon-Hyeong Kim, Youngsoo Jang, Hongseok Yang, and Kee-Eung Kim. Variational inference for sequential data with future likelihood estimates. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5296–5305. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/kim20d.html.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Genshiro Kitagawa. The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623, 1994. URL https://EconPapers.repec.org/RePEc:spr:aistmt:v:46:y:1994:i:4:p:605-623.

Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996. doi: 10.1080/10618600.1996.10474692. URL https://www.tandfonline.com/doi/abs/10.1080/10618600.1996.10474692.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Jonathan Wilder Lavington, Michael Teng, Mark Schmidt, and Frank Wood. A closer look at gradient estimators with reinforcement learning as inference. In *Deep RL Workshop NeurIPS 2021*, 2021.

Tuan Anh Le. Unbiasedness of the Sequential Monte Carlo Based Normalizing Constant Estimator. https://www.tuananhle.co.uk/notes/smc-evidence-unbiasedness.html, 2017. Accessed: 2022-04-29.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6131–6141. PMLR, 2021.

Sergey Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. *arXiv:1805.00909 [cs, stat]*, May 2018. URL http://arxiv.org/abs/1805.00909. arXiv: 1805.00909.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

Fredrik Lindsten and Thomans B. Schön. *Backward Simulation Methods for Monte Carlo Statistical Inference*. Foundations and Trends in Machine Learning. 2013. URL https://ieeexplore.ieee.org/document/8187580.

Vasileios Lioutas, Adam Scibior, and Frank Wood. TITRATED: Learned human driving behavior without infractions via amortized inference. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=M8D5iZsnrO.

Jun S. Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998. doi: 10.1080/01621459.1998.10473765. URL https://doi.org/10.1080/01621459.1998.10473765.

P.D. Moral, A. Doucet, and S.S. Singh. *Forward smoothing using sequential Monte Carlo*. CUED/F-INFENG/TR. University of Cambridge, Department of Engineering, 2009. URL https://books.google.ca/books?id=jgFxmwEACAAJ.

Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Its Applications. Springer-Verlag, New York, 2004. ISBN 978-0-387-20268-6. doi: 10.1007/978-1-4684-9393-1. URL https://www.springer.com/gp/book/9780387202686.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. Sequential monte carlo for graphical models. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 1862–1870, Cambridge, MA, USA, 2014. MIT Press.

Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pp. 1101–1112. PMLR, 2020.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299, 2018. doi: 10.1109/ICRA.2018.8463162.

Gerhard Neumann. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 817–824, 2011.

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.

Michael K. Pitt, Ralph dos Santos Silva, Paolo Giordani, and Robert Kohn. On some properties of markov chain monte carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012. ISSN 0304-4076. doi: https://doi.org/10.1016/j.jeconom.2012.06.004. URL https://www.sciencedirect.com/science/article/pii/S0304407612001510. Bayesian Models, Methods and Applications.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

Daniele Reda, Tianxin Tao, and Michiel van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Motion, Interaction and Games*, pp. 1–10. 2020.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4344–4353. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/riedmiller18a.html.

Moonkyung Ryu, Yinlam Chow, Ross Anderson, Christian Tjandraatmadja, and Craig Boutilier. CAQL: Continuous Action Q-Learning. Technical Report arXiv:1909.12397, arXiv, February 2020. URL http://arxiv.org/abs/1909.12397. arXiv:1909.12397 [cs, stat] type: article.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, Puerto Rico, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Adam Ścibior, Vasileios Lioutas, Daniele Reda, Peyman Bateni, and Frank Wood. Imagining The Road Ahead: Multi-Agent Trajectory Prediction via Differentiable Simulation. In *2021 IEEE 24rd International Conference on Intelligent Transportation Systems (ITSC)*, 2021.

H Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, et al. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*, 2019.

Andreas Stuhlmüller, Robert X. D. Hawkins, N. Siddharth, and Noah D. Goodman. Coarse-to-fine sequential monte carlo for probabilistic programs. 2015. URL http://arxiv.org/abs/1509.02962.

Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors. *arXiv:2101.06557 [cs]*, January 2021. URL http://arxiv.org/abs/2101.06557. arXiv: 2101.06557.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric Wan. The Unscented Particle Filter. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://papers.nips.cc/paper/2000/hash/f5c3dd7514bf620a1b85450d2ae374b1-Abstract.html.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kümmerle, Hendrik Königshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088 [cs, eess]*, 2019.

Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

A APPENDIX

A.1 SEQUENTIAL MONTE CARLO

Here we briefly give an overview of the sequential Monte Carlo (SMC) algorithm adapted for Markov decision processes (MDPs). We borrow the notation from Section 2 in the main paper. Obtaining state-action pairs $(s_{1:T}, a_{1:t})$ that maximize the expected sum of rewards corresponds to sampling state-action pairs from the posterior $p(s_{1:T}, a_{1:T}|O_{1:T})$. The SMC inference algorithm (Gordon et al., 1993) approximate the filtering distributions $p(s_t, a_t|O_{1:t})$. In general, SMC is assuming the existence of a proposal distribution $q(s_{t+1}, a_{t+1}|s_t, a_t)$ but for simplicity we instead use bootstrap proposals that use the prior policy. The algorithm samples N independent particles from the initial distribution $s_1^n \sim p_0(s_1)$ where each particle has uniform weights $w_0^n = 1/N$. At each iteration, the algorithm advances each particle one step forward by sampling actions from $\hat{a}_t^n \sim \pi(a_t|s_t^n)$ and then compute the next states by $\hat{s}_{t+1}^n \sim p(s_{t+1}|s_t^n, \hat{a}_t^n)$ accumulating the optimality likelihoods $p(O_t^n|s_t^n, \hat{a}_t^n, \hat{s}_{t+1}^n) = e^{r(s_t^n, \hat{a}_t^n, \hat{s}_{t+1}^n)}$ in the corresponding particle weight, saving the sum of weights and normalizing them before proceeding to the next time step. SMC suffers from weight disparity which can lead to a reduced effective sample size of particles. This is mitigated by introducing a resampling step RESAMPLE$(\bar{w}_t^{1:N})$ at every iteration to help SMC select promising particles with high weights that have higher chance of surviving whereas particles with low weights most likely will get discarded. See Douc & Cappe (2005) for an extensive overview of different resampling schemes. Algorithm 2 summarizes the SMC process for MPDs using bootstrap proposals.

Algorithm 2 Sequential Monte Carlo

procedure SMC(p_0, p, π, r N, T)
  Sample $s_1^{1:N} \sim p_0(s_1)$
  Set $\bar{w}_0^{1:N} \leftarrow \frac{1}{N}$
  for $t \in 1 \dots T$ **do**
    for $n \in 1 \dots N$ **do**
      Sample $\hat{a}_t^n \sim \pi(a_t|s_t^n)$
      Sample $\hat{s}_{t+1}^n \sim p(s_{t+1}|s_t^n, \hat{a}_t^n)$
      Set $\hat{w}_t^n \leftarrow \bar{w}_{t-1}^n e^{r(s_t^n, \hat{a}_t^n, \hat{s}_{t+1}^n)}$
    end for
    Set $W_t \leftarrow \sum_{i=1}^N \hat{w}_t^i$
    Sample $\alpha_t^{1:N} \sim$ RESAMPLE $\left(\frac{\hat{w}_t^{1:N}}{W_t}\right)$
    for $n \in 1 \dots N$ **do**
      Set $a_t^n \leftarrow \hat{a}_t^{\alpha_t^n}$
      Set $s_{t+1}^n \leftarrow \hat{s}_{t+1}^{\alpha_t^n}$
      Set $\bar{w}_t^n \leftarrow \frac{1}{N} W_t$
    end for
  end for
  return $s_{1:T}^{1:N}, a_{1:T}^{1:N}, \bar{w}_{1:T}^{1:N}$
end procedure

A.2 DERIVATIONS

Soft-Q function Below is the derivation of the soft-Q function defined as the log probability of the backward message.

$$\begin{aligned} Q(s_t, a_t) &:= \log p(O_{t:T}|s_t, a_t) \\ &= \log p(O_t|s_t, a_t) + \log p(O_{t+1:T}|s_t, a_t), \end{aligned} \tag{6}$$

where

$$\log p(O_t|s_t, a_t) = \mathop{\mathbb{E}}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[r(s_t, a_t, s_{t+1}) \right], \tag{7}$$

and

$$\log p(O_{t+1:T}|s_t, a_t) = \log \int_{s_{t+1}} \int_{a_{t+1}} p(s_{t+1}|s_t, a_t)\pi(a_{t+1}|s_{t+1})p(O_{t+1:T}|s_{t+1}, a_{t+1})da_{t+1}ds_{t+1}$$
$$= \log \mathop{\mathbb{E}}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)} \left[\mathop{\mathbb{E}}_{a_{t+1}\sim\pi(a_{t+1}|s_{t+1})} \left[e^{Q(s_{t+1},a_{t+1})} \right] \right]. \tag{8}$$

If we assume the dynamics $p(s_{t+1}|s_t, a_t)$ of the environment to be deterministic we simplify Equation 6 to

$$Q(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \log \mathop{\mathbb{E}}_{a_{t+1}\sim\pi(a_{t+1}|s_{t+1})} \left[e^{Q(s_{t+1},a_{t+1})} \right]. \tag{9}$$

SMC using value function as heuristic factors Piché et al. (2019) proposed using state values $V(s_t)$ as backward messages in SMC for planning. Based on the two-filter formula (Bresler, 1986; Kitagawa, 1994), they derive the following weight update rule

$$w_t = w_{t-1} \mathop{\mathbb{E}}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)} \left[\exp\left(r(s_t, a_t, s_{t+1}) + V(s_{t+1}) - \log \mathop{\mathbb{E}}_{s_t\sim p(s_t|s_{t-1},a_{t-1})} \left[\exp\left(V(s_t)\right) \right] \right) \right]. \tag{10}$$

We omit the term $-\log \pi_\theta(a_t|s_t)$ since we assume to use bootstrap proposals instead of learning them. Thus, the current action is sampled as $a_t \sim \pi(a_t|s_t)$. In our framework, for simplicity, we assume the use of deterministic state transition dynamics $p(s_{t+1}|s_t, a_t)$ which simplifies the update rule to

$$w_t = w_{t-1} \exp\left(r(s_t, a_t, s_{t+1}) + V(s_{t+1}) - V(s_t) \right). \tag{11}$$

Piché et al. (2019) in practice trained an SAC-based policy and used the learned state-action value functions $Q(s_t, a_t)$ to approximate state values $V(s_t)$. Following a similar experimentation setting, we use a soft approximation of the value function terms using state-action value functions $Q(s_t, a_t)$ as described by Levine (2018) using

$$V(s_t) = \log \mathop{\mathbb{E}}_{a_t\sim\pi(a_t|s_t)} \left[\exp(Q(s_t, a_t)) \right]. \tag{12}$$

This results in the following particle weight update rule

$$w_t = w_{t-1} \exp\left(r(s_t, a_t, s_{t+1}) + \log \mathop{\mathbb{E}}_{\hat{a}_{t+1}\sim\pi(a_{t+1}|s_{t+1})} \left[\exp(Q(s_{t+1}, \hat{a}_{t+1})) \right] - \log \mathop{\mathbb{E}}_{\hat{a}_t\sim\pi(a_t|s_t)} \left[\exp(Q(s_t, \hat{a}_t)) \right] \right). \tag{13}$$

We can then define the heuristic factor used in Figure 2b of the main paper as

$$h_t = \log \mathop{\mathbb{E}}_{\hat{a}_{t+1}\sim\pi(a_{t+1}|s_{t+1})} \left[\exp(Q(s_{t+1}, \hat{a}_{t+1})) \right] - \log \mathop{\mathbb{E}}_{\hat{a}_t\sim\pi(a_t|s_t)} \left[\exp(Q(s_t, \hat{a}_t)) \right] \tag{14}$$

which utilizes a soft approximation of the value function. It is worth emphasising that a next state sample s_{t+1} from the environment model is required which makes the use of putative action particles (see Section 3.3 of the main paper) inefficient and expensive contrary to the proposed CriticSMC method.

A.3 Toy Environment Experiment Details

In this environment, the ego agent is described by $e_t = (x_t^e, y_t^e, r^e)$ where x, y is the position in the square coordinate system $[0, 1]^2$ and r^e is the radius. We randomly position other agents $o_t^i = (x_t^{o^i}, y_t^{o^i}, r^{o^i})$ where $i \in [0, 5]$. In addition, there is a partial barrier in the middle with gates $g^k = (x^{g^k}, y^{g^k}, w^{g^k})$ where x^{g^k}, y^{g^k} are the coordinates of the center of the gate k, w^{g^k} is the width of the opening and $k \in [1, 3]$. Finally, a goal position $G = (x^G, y^G, r^G)$ is positioned on the other side of the barrier. The ego and the other agents are moving by displacement actions $a_t = (\Delta x_t, \Delta y_t)$.

The state representation consists of relative distances between the ego agent and the other agents, the center of the gates and the goal position. A two-layer fully connected neural network with a ReLU

activation function and size of 64 takes as input this representation and produces a state encoding. A similar network takes as input the two-dimensional displacement actions and produces the action encoding. Finally, another two-layer network takes as input the concatenation of the state and actions encodings and produce the Q values.

We train the model using a single Nvidia RTX 2080Ti GPU. The prioritized experience replay buffer has a size of 1 million stored experiences. The discount factor is set to 0.99, the batch size to 256 and the learning rate to 0.001. Finally, we sample 1024 actions during running CriticSMC while training the critic model.

A.4 Driving Behavior Model Experiment Details

The prior model we picked for this experiment is ITRA (Ścibior et al., 2021) but any other probabilistic behavior model can be used. We follow the same architecture and training procedure as described in Ścibior et al. (2021). The prior model is trained on the INTERACTION (Zhan et al., 2019) dataset and the task is that given 10 timesteps of observed behavior, predict the next 30 timesteps of future trajectories. For the critic, we used the same convolution neural network architecture as the prior model. The critic takes as input the last two observed birdviews images and encodes them separately. The concatenation of the two representations along with the action encoding is processed by a final layer that produces the Q value. The architecture for these layers is the same as in Section A.3.

We train the critic model using a single Nvidia RTX 2080Ti GPU. The prioritized experience replay buffer has a size of 1.5 million stored experiences. The discount factor is set to 0.99, the batch size to 256 and the learning rate to 0.001. Finally, we sample 128 actions during running CriticSMC while training the critic model.

A.4.1 Reinforcement Learning Environment

The environment used to train the RL agents takes as input a location from the INTERACTION dataset and trains a single-agent policy where all non-ego actors rollout according to ground truth. Because the CriticSMC algorithm rolls out every agent according to ground truth for the first ten frames of each trajectory before prediction, we simply remove these frames and begin executing the policy on frame eleven. At time step t, the policy takes the previous and current birdview images (b_{t-1}, b_t) where each image has a size $256{\times}256{\times}1$. The stacked birdview images make the total input for the policy and value function to be $2{\times}256{\times}256{\times}1$. The policy produces an action $a_t \in [-1, 1]^2$ which corresponds to the bicycle kinematic model's relative action space (see Ścibior et al. (2021) for more details). The differentiable simulator (Ścibior et al., 2021) then uses a_t to update its state and returns the next birdview image b_{t+1}. In this setting the policy distribution that is learned follows a squashed normal distribution (Haarnoja et al., 2018b), as is standard for the SAC implementations (Haarnoja et al., 2018b). The stochastic policy learned by SAC is tailored towards exploration and thus behaves poorly. For this reason we only report its deterministic behavior (e.g. the mode of the policy) in Table 2 of the main paper. For each of the four locations that were evaluated, the RL agents were run over three different learning rate schedules and three different reward structures for a minimum of 150k time steps. The policy uses the same convolutional neural network architecture as in CriticSMC and is updated according to the soft actor-critic algorithm in stable-baselines3 (Brockman et al., 2016). Table 5 shows the hyper-parameter settings used for training.

Reward Surfaces In the three rewards settings which we tested, there were a number of different feedback mechanisms which were used to produce the desired behavior (i.e. low-collision probability and low ADE). The first, was a score based reward upon an estimate of the log-probability under ITRA. To compute this "score reward", the environment passes the pair of birdview images (b_t, b_{t+1}) to ITRA, which generates the hypothetical action a_t^{ITRA} that ITRA would have taken to make the state transition from b_t to b_{t+1}. Then, the environment sets the reward to be a monotonic function of the likelihood of a_t under a normal distribution centred around a_t^{ITRA}: $r_{t+1} \equiv \tanh\left(\log p(a_t; a_t^{\text{ITRA}}, \Sigma)\right)$ where $p(\cdot; a_t^{\text{ITRA}}, \Sigma) = \mathcal{N}(a_t^{\text{ITRA}}, \Sigma)$ for some covariance Σ.

Next, we include five simpler reward surfaces which have been shown to improve performance in the literature (Reda et al., 2020). First, the "action reward" is a linear function of the absolute difference between the action output by the policy and the action which ITRA would have taken at time step t:

Reinforcement Learning Baseline Parameters		
Parameter Name	Parameter Value(s)	Parameter Description
Σ	I_2	The covariance matrix of the multivariate normal distribution centred around hypothetical ITRA action a_t^{ITRA}.
α_1	0.15	Coefficient for score reward, this parameter scales how closely the agent should track estimated log-likelihood of actions under the ITRA model.
α_2	2	Coefficient for action reward, this incentives the policy to be as close to the mode of ITRA as possible without access to a score function over those actions.
α_3	0.05	Coefficient for action difference reward, this incentivizes the agent to produce sequences of actions which are smoother, and therefore often more human-like.
$\alpha_{4,5,6}$	0 or 1	Boolean coefficients selecting whether infraction, survival, or ground truth rewards are used.
γ	0.99	Discount factor, set to encourage lower variance gradient estimates, but greedier policy behavior (Sutton & Barto, 2018).
Learning Rate	0.0002, 0.00012, 0.00008	Learning rate for optimization (in this case the Adam Optimizer).
Batch Size	256	Number of examples used in each gradient decent update for both the critic and policy networks.
Buffer Size	500000	Size of SAC experience buffer (equivalent to maximum number of steps which can be taken within the environment).
Learning Starts	1000	Number of exploration steps used (e.g. a uniform distribution over actions) before learned stochastic policy is used to gather interactions.
τ	0.005	Polyak parameter averaging coefficient which improves convergence of deep Q learning algorithms (Haarnoja et al., 2018b).
Latent-Features	256	Number of neurons used in the output of the feature encoder, and which is fed to the standard two layer multi-layer perceptron defined by standard SAC algorithms (Haarnoja et al., 2018b).

Table 5: Hyper-parameters for the reinforcement learning baseline used in Section 4.2. All hyper-parameters which were not listed above, use the default values provided by the SAC implementation of stable-baselines3 (Brockman et al., 2016).

$\big(2 - ||a_t - a_t^{\text{ITRA}}||_1\big)$. Second, the action difference reward is the scaled absolute difference between the current and previous actions: $||a_t - a_{t-1}||_1$. Third, the environment computes the "ground-truth reward" r_{t+1} by evaluating s_t against the ground truth data from the INTERACTION dataset. In particular, the environment sets the reward to be a linear function of the negative Euclidean distance at time $t+1$ between the xy-coordinate of the ego-vehicle according to the simulator, s_{t+1}, and that according to ground truth, s_{t+1}^{GT}: $100 - ||s_{t+1} - s_{t+1}^{\text{GT}}||_2$. Fourth, we include a "survival reward" of 1 if the agent does not commit an infraction at step t. Lastly, the infraction reward is -5 if the agent commits any type of infraction at step t and 5 otherwise.

Using these five feedback mechanisms, we consider three different reward surfaces. Each of which are defined following reward calculation:

$$\begin{aligned} r_{t+1} = \alpha_1 r_{t+1}^{\text{SCORE}} + \alpha_2 r_{t+1}^{\text{ACTION}} + \alpha_3 r_{t+1}^{\text{ACTION DIFF}} \\ + \alpha_4 r_{t+1}^{\text{INFRACTION}} + \alpha_5 r_{t+1}^{\text{SURVIVE}} + \alpha_6 r_{t+1}^{\text{GROUND TRUTH}} \end{aligned} \tag{15}$$

In the first reward setting which was considered, we set all coefficients α_i to zero except the SURVIVE reward, and thus refer to this reward type as the survival reward setting. Next we consider a setting where we set all α_i to zero except the GROUND TRUTH reward, and refer to this setting as the ground-truth setting. Lastly, we considered a setting where: $\alpha_1 = 0.15$, $\alpha_2 = 2.0$, $\alpha_3 = 0.05$, and the remaining α_i are all set to zero. We refer to this setting as the ITRA setting, as it includes the most information about the ITRA model. To arrive at the final result, models where trained under all three of these settings, evaluated, and then chosen based upon the lowest collision infraction rate.

A.5 NOTATIONS AND ABBREVIATIONS

$s_{1:T}$	$\triangleq$	sequence of states
$a_{1:T}$	$\triangleq$	sequence of actions
$\pi(a_t \mid s_t)$	$\triangleq$	prior policy
$p(s_{t+1} \mid s_t, a_t)$	$\triangleq$	state transition dynamics density
$r(s_t, a_t, s_{t+1})$	$\triangleq$	reward value received at timestep t
$p(O_t \mid s_t, a_t, s_{t+1})$	$\triangleq$	optimality probability defined as the exponentiated reward
$Q(s_t, a_t)$	$\triangleq$	soft state-action value function referred to as the critic
Q_ϕ	$\triangleq$	parametric approximation of the critic
Q_ψ	$\triangleq$	fixed target critic model used for computing the TD error
h_t	$\triangleq$	heuristic factor at timestep t
$\hat{w}_t$	$\triangleq$	pre-resampling particle weight
$\bar{w}_t$	$\triangleq$	post-resampling particle weight
W_t	$\triangleq$	normalizing factor
α_t^i	$\triangleq$	ancestral indices for each particle i
β_{pen}	$\triangleq$	penalty coefficient
γ	$\triangleq$	discount factor
T	$\triangleq$	horizon length
$t \in 1 \dots T$	$\triangleq$	timesteps
$n \in 1 \dots N$	$\triangleq$	particle number
$k \in 1 \dots K$	$\triangleq$	putative action particle number

Table 6: Notations

SMC:	Sequential Monte Carlo
MDP:	Markov Decision Process
RL:	Reinforcement Learning
HMM:	Hidden Markov Model
RLAI:	Reinforcement Learning as Inference
TD:	Temporal Difference
SAC:	Soft Actor Critic
MPC:	Model Predictive Control
ADE:	Average Displacement Error
MFD:	Maximum Final Distance

Table 7: Abbreviations

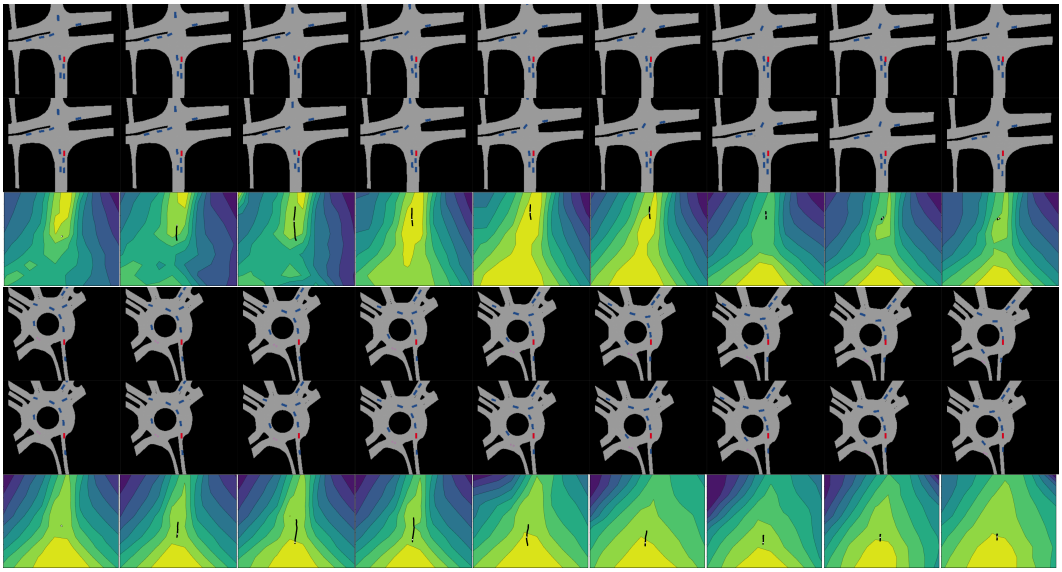


Figure 5: More examples similar to Figure 3 of the main paper.