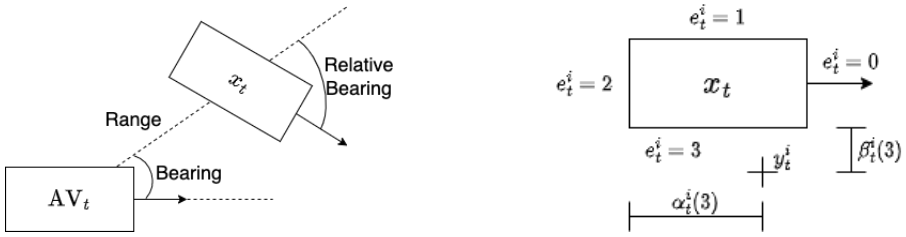


(a) Input features provided to the observation model's neural network.

(b) Output representation of an observed point y_t^i using the triplet $\left(e_t^i, \alpha_t^i(e_t^i), \beta_t^i(e_t^i)\right)$.

Figure 3: Input and output representations for the observation model.

A Observation Model

For the problem setting described in Section 5, our observation model is concerned with the distribution of 2D points around the peripheries of observed road users. Such models are reviewed in [25]. Let the i^{th} point from an observation y_t be y_t^i. We assumed independence across different points, i.e.

$$g_\theta(y_t|x_t) = \prod_i g_\theta\left(y_t^i|x_t\right). \tag{12}$$

In order to conveniently sample y_t^i or to measure its likelihood, we represented each point via a triplet, $\left(e_t^i, \alpha_t^i\left(e_t^i\right), \beta_t^i\left(e_t^i\right)\right)$, where $e_t^i \in \{0,1,2,3\}$ is a categorical variable encoding the edge of the road user's bounding box, $\alpha_t^i\left(e_t^i\right)$ is the corresponding parallel offset, and $\beta_t^i\left(e_t^i\right)$ is the corresponding perpendicular offset (see Figure 3b). Such a triplet uniquely determines y_t^i, and we used following generative model to sample such points:-

$$e_t^i \sim \text{Categorical}\left(\phi_\theta^0\left(x_t\right), \phi_\theta^1\left(x_t\right), \phi_\theta^2\left(x_t\right), \phi_\theta^3\left(x_t\right)\right),$$

$$\begin{aligned}
\text{if } e_t^i = 0, \quad \alpha_t^i &\sim \text{Uniform}(0, w) \text{ and} \\
\beta_t^i &\sim \text{Laplace}\left(\mu = \phi_\theta^4\left(x_t\right), b = \phi_\theta^5\left(x_t\right)\right), \\
\text{if } e_t^i = 1, \quad \alpha_t^i &\sim \text{Uniform}(0, l) \text{ and} \\
\beta_t^i &\sim \text{Laplace}\left(\mu = \phi_\theta^6\left(x_t\right), b = \phi_\theta^7\left(x_t\right)\right), \\
\text{if } e_t^i = 2, \quad \alpha_t^i &\sim \text{Uniform}(0, w) \text{ and} \\
\beta_t^i &\sim \text{Laplace}\left(\mu = \phi_\theta^8\left(x_t\right), b = \phi_\theta^9\left(x_t\right)\right), \\
\text{if } e_t^i = 3, \quad \alpha_t^i &\sim \text{Uniform}(0, l) \text{ and} \\
\beta_t^i &\sim \text{Laplace}\left(\mu = \phi_\theta^{10}\left(x_t\right), b = \phi_\theta^{11}\left(x_t\right)\right),
\end{aligned}$$

where l and w are the length and the width of the road user's bounding box respectively, and $\phi_\theta^{0:11}\left(x_t\right)$ are the outputs from a neural network with weights θ. While the mapping $\left(e_t^i, \alpha_t^i\left(e_t^i\right), \beta_t^i\left(e_t^i\right)\right) \to y_t^i$ is unique, the reverse mapping has four representations depending on the choice of e_t^i, i.e., e_t^i is a latent variable. The likelihood of a point, $g_\theta\left(y_t^i|x_t\right)$, was therefore obtained by:-

$$g_\theta\left(y_t^i|x_t\right) = \sum_{e_t^i=0}^{3} \phi_\theta^{e_t^i}\left(x_t\right) p_\theta\left(\alpha_t^i\left(e_t^i\right), \beta_t^i\left(e_t^i\right)|e_t^i, x_t\right), \tag{13}$$

where we project $y_t^i \to \left(e_t^i, \alpha_t^i\left(e_t^i\right), \beta_t^i\left(e_t^i\right)\right)$ for each $e_t^i \in \{0,1,2,3\}$, and marginalise over it. We use a feed-forward neural network with 4 hidden layers, each with 16 Tanh units, which outputs 12 parameters, $\phi_\theta^{0:11}\left(x_t\right)$. We provide the network with 5 input features - range, bearing, relative bearing, length, and width - of the road-user's bounding box as measured from the observing AV's viewpoint (see Figure 3a).

For each of the experiments in Section 5 (with synthetic and real data), we used the same observation model design. Moreover, we trained an observation model using supervision from a dataset of manually labelled trajectories, $\bar{x}_{0:T}$, by maximising the AOTLL. This model was then used to generate the synthetic data and also as a baseline for the experiments with real data.

B Transition Model

As described in Section 4, the class of SSMs that we are concerned with involves a transition function, $f_\theta(x_t|x_{t-1})$, that factorises into a policy which produces actions, $\pi_\theta(a_t|x_{t-1})$, and a deterministic, differentiable, and injective motion model, such that $x_t = \tau(x_{t-1}, a_t)$. In this section, we provide more details on the motion model and on the policies used for the experiments described in Section 5.

B.1 Motion Model

All experiments used a **state space** that consists of the 2D Pose (x, y, θ) of an observed road user, in addition to its instantaneous linear speed v and curvature κ. Moreover, the **action space** used consists of linear acceleration a and pinch p, i.e., the instantaneous rate of change of curvature. This choice of actions, i.e. acceleration and pinch, naturally maps to the controls exercised by road users (vehicles users in particular), i.e., to gas and rate of change of steering respectively. The calculations below compute the next state, $(x(t), y(t), \theta(t), v(t), \kappa(t))$, from the previous state $(x_0, y_0, \theta_0, v_0, \kappa_0)$ under the influence of constant actions (a, p) for $t \in [0, \Delta t]$.

$$\text{Clearly} \quad \dot{\kappa}(t) = p \quad \Rightarrow \kappa(t) = \kappa_0 + pt, \tag{14}$$

$$\text{and} \quad \dot{v}(t) = a \quad \Rightarrow v(t) = v_0 + at. \tag{15}$$

$$\text{Since} \quad \dot{\theta}(t) = v(t)\kappa(t), \text{ we have} \tag{16}$$

$$\dot{\theta}(t) = v_0\kappa_0 + (v_0 p + a\kappa_0)t + apt^2, \tag{17}$$

$$\Rightarrow \quad \theta(t) = \theta_0 + v_0\kappa_0 t + (v_0 p + a\kappa_0)\frac{t^2}{2} + ap\frac{t^3}{3}. \tag{18}$$

$$\text{Finally, using} \quad \dot{x}(t) = v(t)\cos\theta(t), \text{ and } \dot{y}(t) = v(t)\sin\theta(t), \text{ we have} \tag{19}$$

$$x(t) = x_0 + \int_0^t v(s)\cos\theta(s)\,\mathrm{d}s, \tag{20}$$

$$\text{and} \quad y(t) = y_0 + \int_0^t v(s)\sin\theta(s)\,\mathrm{d}s \tag{21}$$

$$= y_0 + \int_0^t v(s)\cos\left(\frac{\pi}{2} - \theta(s)\right)\mathrm{d}s. \tag{22}$$

To make these integrals analytically tractable, we drop the cubic term in θ_t, i.e. $ap\frac{t^3}{3}$, and use the integral[2] $\int (a + bs)\cos(c + ds + es^2)\,\mathrm{d}s$ with appropriate coefficients to compute $x(t)$ and $y(t)$. This approximation is justified as for the experiments described in Section 5, we use a small Δt of ≈ 0.33 seconds.

B.2 Policy

For *synthetic data*, we designed a simple state-dependent stochastic policy that modulated its mean acceleration and pinch as a function of speed. The mapping from state to actions is described in Figure 4. The policy then had 2 learnable parameters - the standard deviations of its acceleration and pinch. The advantage of having only 2 learnable parameters is that it allowed us to easily verify if a method was converging to the correct values or not.

On *real data*, we used a more expressive policy which produces a multivariate Gaussian distribution over acceleration and pinch conditioned on its inputs. The policy's inputs are the instantaneous speed and curvature of an object, which are then fed to a 3-layer feedforward neural network, with 2 hidden layers with 32 ReLU units, outputting 5 parameters - the means of acceleration and pinch, and the 3 elements of the lower triangular matrix representation of the covariance of the two. This gives a total of 1,317 learnable parameters.

[2]The closed form integral was obtained using Wolfram Alpha.

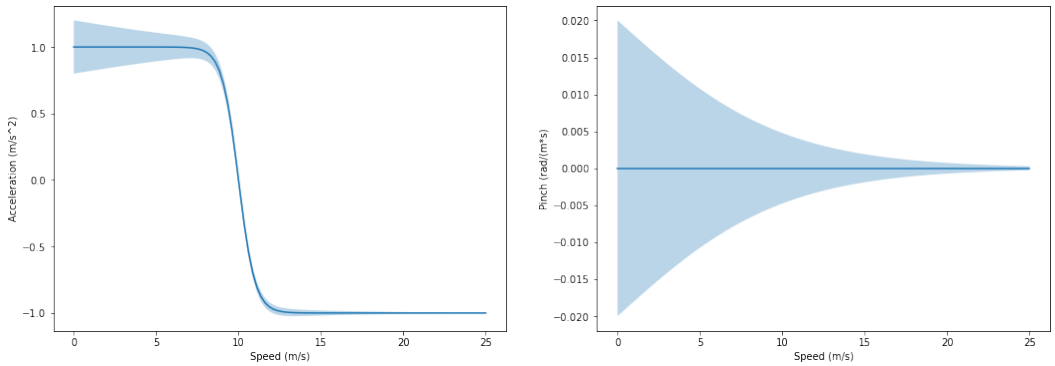
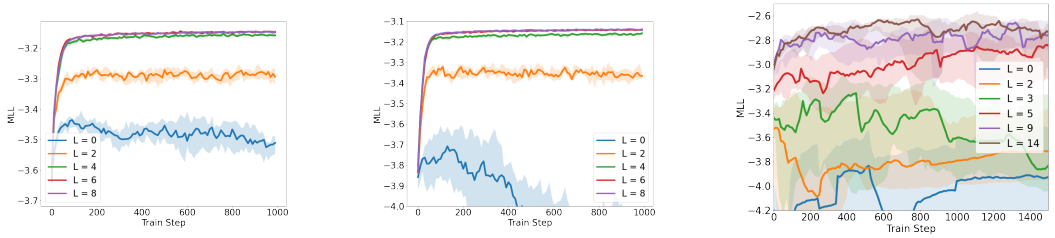


Figure 4: State to action mapping of the simple policy. The solid line represents the mean action taken at a given speed, while the shaded regions represent one standard deviation of Gaussian noise around that action. At a high-level, the policy accelerates at low speeds and decelerates at higher speeds. The policy also applies less pinch at higher speeds.



(a) MLL of test synthetic data with 25 step trajectories using PF-SEFI with different values for the smoothing lag (L).

(b) MLL of test synthetic data with 50 step trajectories using PF-SEFI with different values for the smoothing lag (L).

(c) MLL of test real data with 60 step trajectories using PF-SEFI with different values for the smoothing lag (L) using 2048 training particles.

Figure 5: Marginal Log Likelihood (MLL) of synthetic test data for models trained using PF-SEFI with different values for the smoothing lag (L) plotted against the corresponding training steps.

C Training Setup

In this section, we present our training setup for the experiments described in Section 5. We show sample trajectories from each of the datasets (synthetic and real), and provide the set of hyper-parameters that were used for each of the experiments. All experiments (including the ones used for picking the best set of hyper-parameters) were repeated 10 times to obtain the median and interquartile ranges that are shown in Figures 2, 5, and 7, and in Table 1. All experiments used the default settings of the Adam optimiser in TensorFlow with a learning rate of 0.01. We found that smaller learning rates yield similar results, but require proportionately longer training time, while larger learning rates cause instability.

C.1 Training on Synthetic Data

Sample trajectories from the generated synthetic data are shown in Figure 8. These samples were generated using a hand-crafted policy (see Section B.2), the motion model derived in Section B.1, and an observation model trained with supervised learning (see Section A). We generated two datasets (with 25 and 50 steps respectively), each containing 10 scenes with 100 objects each for training, and 2 scenes also with 100 objects each for evaluation. For every train step, we used all 100 objects from a single randomly sampled training scene, while for every evaluation step, we used all 100 objects from both evaluation scenes.

Table 2 tabulates the set of hyper-parameters that were used for the experiments discussed in Section 5. Figures 5a and 5b show the effect of different values for the smoothing lag hyper-parameter (L) for PF-SEFI on synthetic data, while 5c shows the effect of L on real data. We observed that for

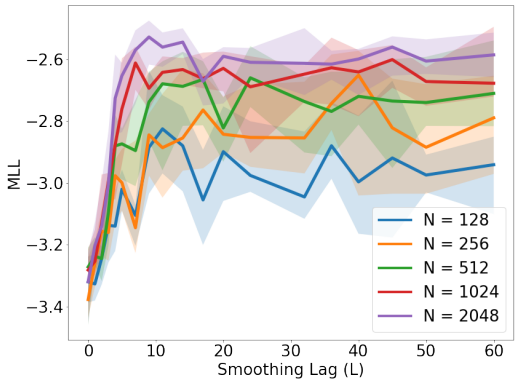
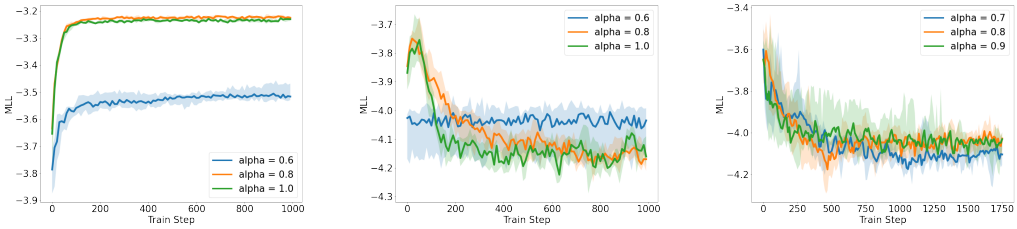


Figure 6: Maximum MLL of test real data with 60 step trajectories using PF-SEFI sweeping over different values for the smoothing lag (L) using different numbers of particles for training.



(a) MLL of test synthetic data with 25 step trajectories using PFNET with different values for the trade-off parameter (α).

(b) MLL of test synthetic data with 50 step trajectories using PFNET with different values for the trade-off parameter (α).

(c) MLL of test real data with 60 step trajectories using PFNET with different values for the trade-off parameter (α).

Figure 7: Marginal Log Likelihood (MLL) of synthetic and real test data for models trained using PFNET with different values for the trade-off parameter (α) plotted against the corresponding training steps.

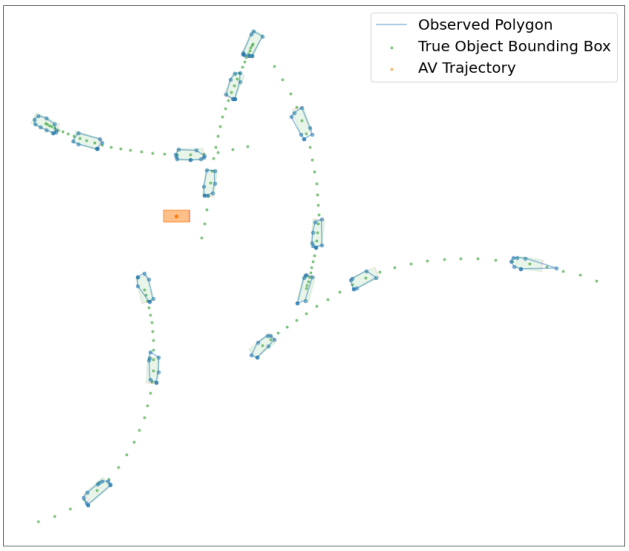
synthetic data, the performance started to plateau at $L = 8$, and hence picked $L = 8$ for the final experiments. On real data, performance continues to improve at higher values of L up to around $L = 14$ and we also note that variance in training is high when L is too small. Moreover, Figures 7a and 7b show the effect of different values for the trade-off parameter (α) for PFNET. While learning failed on synthetic data with 50 steps, we observed that $\alpha = 0.8$ marginally outperformed $\alpha = 1.0$ and significantly outperformed $\alpha = 0.6$.

C.2 Training on Real Data

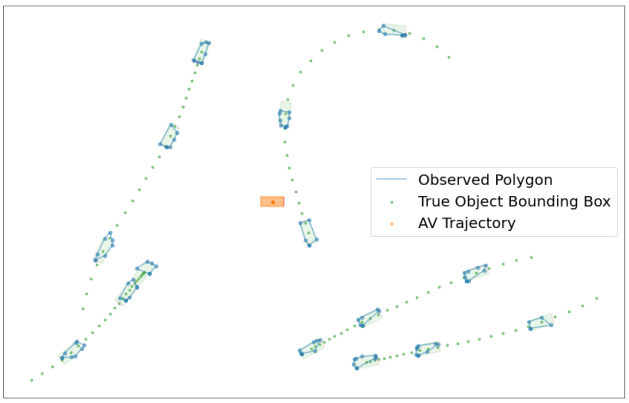
For training on real data, we used a dataset of real-world road-user trajectories observed by an AV. Many of the frames in this set were also labelled manually by human-labelers, allowing us to

Table 2: Hyper-parameters used for experiments A (synthetic data with 25 step trajectories) and B (synthetic data with 50 step trajectories). Smoothing lag (L) is only relevant for PF-SEFI, and the trade-off parameter (α) is only relevant for PFNET.

Hyper-Parameter	Value
Learning Rate	0.01
Number of Epochs	100
Smoothing Lag (L) for PF-SEFI	8
Trade-off Parameter (α) for PFNET	0.8
Number of Particles for Training	1024
Number of Particles for Evaluation	4096



(a)



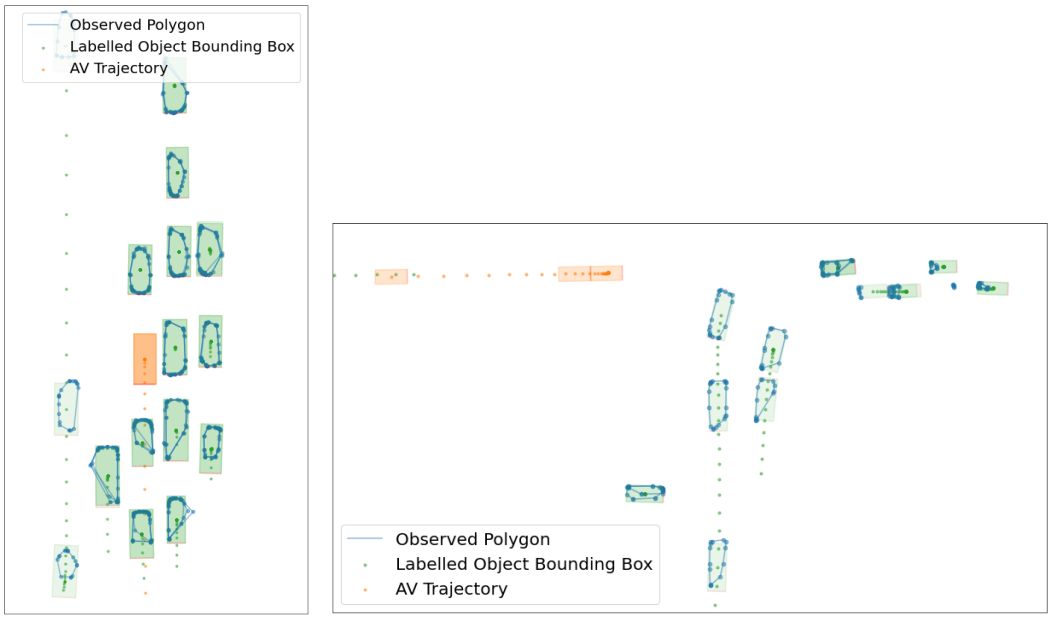
(b)

Figure 8: Examples of synthetic trajectories. The figure shows 3 snapshots of the state of multiple objects, each occurring 10 steps apart from each other. The green boxes are the true states of the objects at different timesteps, while the blue polygons are the observations that the AV (orange) makes. The green dots are the true (x, y) coordinates of the objects at all timesteps.

compute relevant tracking metrics such as ADE, AYE, and AOTLL under the different models that we learned.

All trajectories were approximately 20 seconds long, where each step corresponded to 0.33 seconds in real time, giving 60 steps in discrete time. The dataset consisted of a training set with 1502 trajectories, and a test set containing 404 trajectories. Figure 9 shows some example trajectories.

We ran hyper-parameter sweeps over learning rates, the smoothing lag (L) for PF-SEFI, the trade-off parameter (α) for PFNET, and the number of training particles. The final results are using the best settings of these parameters which we list in Table 3, though we note that results were quite insensitive to most of these settings. Figure 6 shows the effect of L on the maximum achieved test MLL on real data for different numbers of training particles. As can be seen, PF-SEFI does significantly better as L increases from 0 to 10, highlighting the added benefit of smoothing. On the other hand, it is quite insensitive to the smoothing lag L beyond this point, except that the variance appears to increase when L gets too large. We also find that PF-SEFI consistently improves with more particles.



(a) The AV in a crowded multi-lane road surrounded by mostly stationary objects.

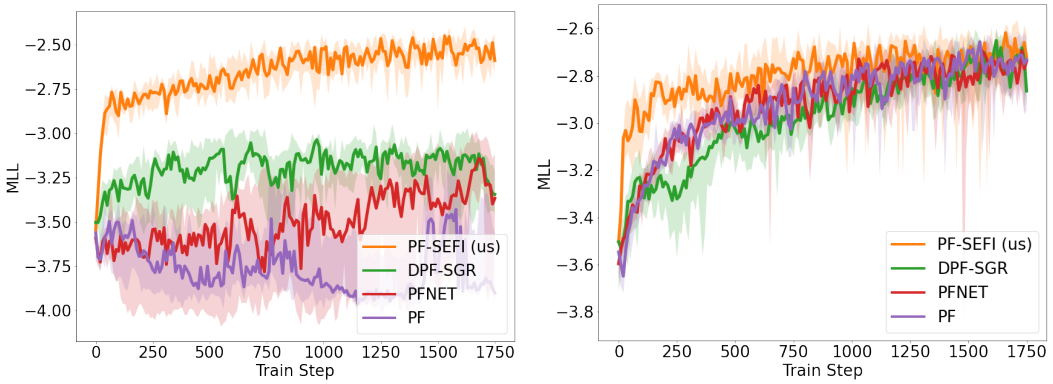
(b) The AV decelerating, yielding for other objects at an intersection.

Figure 9: Examples of real-data trajectories. The figure shows 3 snapshots of the state of multiple objects, each occurring 5 seconds apart from each other. The green boxes are the labelled bounding boxes of objects at different timesteps, while the blue polygons are the observations that the AV (orange) makes. The dots represent the labelled (x, y) coordinates of the objects over time.

Figure 10: Sampled observations on the same labelled states from the real data that were shown in Figure 9. Notice the qualitative similarity in the observations in this figure relative to Figure 9.

Table 3: Hyper-parameters used for experiment C (real data with 60 step trajectories). Smoothing lag (L) is only relevant for PF-SEFI, and the trade-off parameter (α) is only relevant for PFNET.

Hyper-Parameter	Value
Learning Rate	0.01
Global Grad Norm Clipping	0.5
Number of Epochs	15
Smoothing Lag (L) for PF-SEFI	15
Trade-off Parameter (α) for PFNET	0.8
Number of Particles for Training	4096
Number of Particles for Evaluation	4096



(a) MLL of test real data after training models using 30 step sequences.

(b) MLL of test real data after training models using 15 step sequences.

Figure 11: Marginal Log Likelihood (MLL) of real test data for models trained on shorter sequences plotted against the corresponding training steps.

When training on real data, all methods were subject to very large gradients at times, especially earlier on when failing to track objects is much more likely, leading to very high losses and corresponding gradients. In order to stabilise training, we clipped the maximum global norm of all gradients to 0.5. This is particularly necessary for the methods that require differentiating through the filter.

D Sampled Observations from Learned Observation Model

Figure 10 shows sampled observations from the observation model learned on real data using PF-SEFI. These observations were sampled using the checkpoint that produced the highest MLL, and for the same (labelled) states that were shown in Figure 9. The qualitative similarity between the real and sampled observations indicates the efficacy of our method for learning generative models that can be used to sample observations in closed-loop simulation.

E Training on Shorter Sequences

In the case of synthetic data, we note that some methods such as DPF-SGR performed poorly when trained on 50 step sequences (Figure 2b), however performed much better when trained on shorter 25 step sequences (Figure 2a). We conducted similar experiments on real data to see if a similar improvement can be attained. Figure 11 shows the results when training DPF-SGR and other baselines on 30 and 15 step sequences of real data (instead of 60). At 30 steps (Figure 11a), we find that all 3 baselines still fail to learn good models, while PF-SEFI performs almost as well as on length 60 sequences. At 15 steps (Figure 11b), however, the baselines do actually perform much better, though the final performance for all methods is worse than PF-SEFI on 60 steps.

Table 4: Metrics computed on held out synthetic test data comparing PF-SEFI (us) against baselines DPF-SGR, PFNET, and vanilla PF, on two experiments - (A) learning from synthetic data with 25 steps, and (B) learning from synthetic data with 50 steps. The Smoothing ADE and AYE reported here are the same as in Table 1. They are contrasted with the Filtering ADE and AYE, which measure the performance of the learned models in the online setting of state estimation.

Exp.	Method	Smoothing ADE (m)	Filtering ADE (m)	Smoothing AYE (rad)	Filtering AYE (rad)
A	TRUE	0.090 ± 0.001	0.144 ± 0.001	0.014 ± 0.000	0.034 ± 0.000
	PF-SEFI (us)	0.186 ± 0.021	0.233 ± 0.009	0.016 ± 0.000	0.039 ± 0.000
	DPF-SGR	$\mathbf{0.165 \pm 0.008}$	$\mathbf{0.222 \pm 0.008}$	$\mathbf{0.014 \pm 0.000}$	$\mathbf{0.035 \pm 0.001}$
	PFNET	0.264 ± 0.019	0.333 ± 0.030	0.017 ± 0.000	0.047 ± 0.001
	PF	0.245 ± 0.021	0.345 ± 0.024	0.017 ± 0.000	0.047 ± 0.001
B	TRUE	0.088 ± 0.001	0.154 ± 0.001	0.012 ± 0.000	0.038 ± 0.000
	PF-SEFI (us)	$\mathbf{0.165 \pm 0.013}$	$\mathbf{0.239 \pm 0.007}$	$\mathbf{0.014 \pm 0.000}$	$\mathbf{0.043 \pm 0.000}$
	DPF-SGR	2.828 ± 0.415	2.888 ± 0.232	0.142 ± 0.016	0.189 ± 0.008
	PFNET	2.809 ± 0.176	3.133 ± 0.108	0.148 ± 0.008	0.213 ± 0.009
	PF	2.502 ± 0.042	3.207 ± 0.106	0.137 ± 0.007	0.212 ± 0.008

This highlights an advantage of PF-SEFI, which is that it is relatively invariant to the length of sequences that it is trained on. Depending on the problem setting, there is usually a minimum sequence length required to obtain enough information to learn the correct models. If that sequence length is longer than the maximum sequence length for which a method such as DPF-SGR is stable to train, then one must sacrifice model quality for stable learning by cutting the sequences to shorter subsequences or by subsampling the sequences, throwing away some of the observations. In this case, shortening the sequences to 15 steps allowed for reasonable (though suboptimal) models to be learned using the baseline methods. In other problems it may well be the case that even more trimming and/or subsampling would be needed.

F Performance on the Filtering Task

In Section 5, we considered metrics such as ADE and AYE that pertain to the task of state estimation in the offline setting (known as smoothing), i.e., where we assume access to the entire sequence of observations, i.e. $y_{1:T}$. In this section, we additionally consider the online setting (known as filtering) where the task is state estimation at each time step t, with observations from time step 0 up to time step t, i.e. $p(x_t|y_{0:t})$. This setting is relevant for the use of our learned models on-board the AV.

Table 4 reports ADE and AYE using both the smoothing and filtering distribution for the offline and online task of state estimation respectively. The patterns are unchanged relative to the ones observed in Section 5 and in Table 1. However, these results highlight the applicability of the learned in both settings.

G Training with Higher Dimensional Observations

In Section 5, we reported experiments with 32 dimensional observations (16 2D points). In this section, we report additional experiments with even higher dimensional observations (32 and 64 2D points, i.e., 64 and 128 dimensional observations respectively) on the synthetic dataset with 25 steps, trained using PF-SEFI. In Table 5 we summarise the empirical findings of these experiments. In each case, the learned models match the performance of the true models as measured by MLL. These results suggest that PF-SEFI scales well with higher dimensional observations.

H Effect of a Noisy AV State on Learning

In Section 5, we assume that the AV state is known precisely. In practice, we expect there to be some minimal errors in state estimation. To test our sensitivity to the same, we ran additional experiments with the 25 steps synthetic dataset by injecting Gaussian noise (with a standard deviation of 0.5m in x and y, and 0.05rad in θ) in the AV's 2D pose. We retrained our models using PF-SEFI in the

Table 5: MLL computed on held out synthetic test data comparing PF-SEFI (us) against the true models on synthetic dataset generated with 32 (A), 64 (D), and 128 (E) dimensional observations.

Exp.	Method	MLL
A	TRUE	-3.161 $\pm$ 0.003
	PF-SEFI (us)	-3.152 $\pm$ 0.006
D	TRUE	-3.171 $\pm$ 0.003
	PF-SEFI (us)	-3.163 $\pm$ 0.007
E	TRUE	-3.188 $\pm$ 0.003
	PF-SEFI (us)	-3.177 $\pm$ 0.006

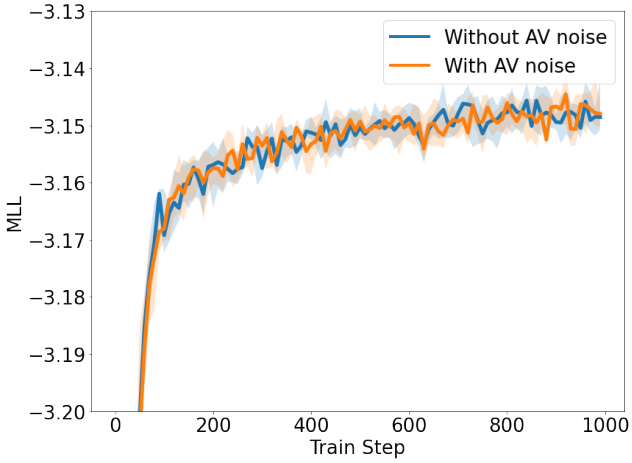


Figure 12: Marginal Log Likelihood (MLL) of synthetic test data using models trained on synthetic data with noisy AV states.

presence of such noise, and observe no change in MLL at convergence over the held out test data (see Figure 12), nor in training stability.