Supplementary Material for: How to Train Your Differentiable Filter

Alina Kloss · Georg Martius · Jeannette Bohg

A Technical Background

In the following section, we briefly review the Bayesian filtering algorithms that we use as basis for our differentiable filters.

A.1 Kalman Filter

The Kalman filter (Kalman, 1960) is a closed-form solution to the filtering problem for systems with a linear process and observation model and Gaussian additive noise:

$$f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{q}_t \qquad \mathbf{q}_t \sim N(0, \mathbf{Q}_t) \tag{S1}$$

$$h(\mathbf{x}_t) = \mathbf{H}\mathbf{x}_t + \mathbf{r}_t \qquad \mathbf{r}_t \sim N(0, \mathbf{R}_t) \tag{S2}$$

The belief about the state $\mathbf{x}$ is represented by the mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of a normal distribution. At each timestep, the filter predicts $\hat{\boldsymbol{\mu}}_t$ and $\hat{\boldsymbol{\Sigma}}_t$ using the process model. The innovation $\mathbf{i}_t$ is the difference between the predicted and actual observation and is used to correct the prediction. The Kalman Gain $\mathbf{K}$ trades-off the process noise $\mathbf{Q}$ and the observation noise $\mathbf{R}$ to determine the magnitude of the update.

Prediction Step:

$$\hat{\boldsymbol{\mu}}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_t \tag{S3}$$

$$\hat{\boldsymbol{\Sigma}}_t = \mathbf{A}\boldsymbol{\Sigma}_{t-1}\mathbf{A}^T + \mathbf{Q}_{t-1} \tag{S4}$$

Update Step:

$$\mathbf{S}_t = \mathbf{H}\hat{\boldsymbol{\Sigma}}_t\mathbf{H}^T + \mathbf{R}_t \tag{S5}$$

$$\mathbf{K}_t = \hat{\boldsymbol{\Sigma}}_t\mathbf{H}^T\mathbf{S}_t^{-1} \tag{S6}$$

$$\mathbf{i}_t = \mathbf{z}_t - \mathbf{H}\hat{\boldsymbol{\mu}}_t \tag{S7}$$

$$\boldsymbol{\mu}_t = \hat{\boldsymbol{\mu}}_t + \mathbf{K}_t\mathbf{i}_t \tag{S8}$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I}_n - \mathbf{K}_t\mathbf{H})\hat{\boldsymbol{\Sigma}}_t \tag{S9}$$

A.2 Extended Kalman Filter (EKF)

The EKF (Sorenson, 1985) extends the Kalman filter to systems with non-linear process and observation models. It replaces the linear models for predicting $\hat{\boldsymbol{\mu}}$ in Equation S3 and the corresponding observations $\hat{\mathbf{z}}$ in Equation S7 with non-linear models $f(\cdot)$ and $h(\cdot)$. For predicting the state covariance $\boldsymbol{\Sigma}$ and computing the Kalman Gain $\mathbf{K}$, these non-linear models are linearized around the current mean of the belief. The Jacobians $\mathbf{F}_{|\mu_t}$ and $\mathbf{H}_{|\mu_t}$ replace $\mathbf{A}$ and $\mathbf{H}$ in Equations S4 - S6 and S9. This first-order approximation can be problematic for systems with strong non-linearity, as it does not take the uncertainty about the mean into account (Van Der Merwe, 2004).

A.3 Unscented Kalman Filter (UKF)

The UKF (Julier and Uhlmann, 1997; Van Der Merwe, 2004) was proposed to address the aforementioned problem of the EKF. Its core idea, the *Unscented Transform* (Julier and Uhlmann, 1997), is to represent a Gaussian random variable that undergoes a non-linear transformation by a set of specifically chosen points in state space, the so called *sigma points* $\boldsymbol{\chi} \in \mathbf{X}$.

$$\lambda = \alpha^2(\kappa + n) - n \tag{S10}$$

$$\boldsymbol{\chi}^0 = \boldsymbol{\mu}$$

$$\boldsymbol{\chi}^i = \boldsymbol{\mu} \pm (\sqrt{(n+\lambda)\boldsymbol{\Sigma}})_i \qquad \forall i \in \{1...n\} \tag{S11}$$

$$w_m^0 = \frac{\lambda}{\lambda + n} \qquad w_c^0 = \frac{\lambda}{\lambda + n} + (1 - \alpha^2 + \beta)$$

$$w_m^i = w_c^i = \frac{0.5}{\lambda + n} \qquad \forall i \in \{1...2n\} \tag{S12}$$

Here, n is the number of dimensions of the state $\mathbf{x}$. Each sigma point $\boldsymbol{\chi}^i$ has two weights w_m^i and w_c^i. The parameters α and κ control the spread of the sigma points and how strongly the original mean $\boldsymbol{\chi}^0$ is weighted in comparison to the other sigma points. $\beta = 2$ is recommended if the true distribution of the system is Gaussian.

The statistics of the transformed random variable can then be calculated from the transformed sigma points. For example, in the prediction step of the UKF, the non-linear transform is the process model (Eq. S13) and the new mean and covariance of the belief are computed in Equations S14 and S15.

Alina Kloss, Georg Martius
Max Planck Institute for Intelligent Systems, Tübingen, Germany
E-mail: firstname.lastname@tuebingen.mpg.de

Jeannette Bohg
Stanford University, Stanford, USA
E-mail: bohg@stanford.edu

$$\hat{\mathbf{X}}_t = f(\mathbf{X}_{t-1}, \mathbf{u}_t) \tag{S13}$$

$$\hat{\boldsymbol{\mu}}_t = \sum_i w_m^i \hat{\boldsymbol{\chi}}_t^i \tag{S14}$$

$$\hat{\boldsymbol{\Sigma}}_t = \sum_i w_c^i (\hat{\boldsymbol{\chi}}_t^i - \hat{\boldsymbol{\mu}}_t)(\hat{\boldsymbol{\chi}}_t^i - \hat{\boldsymbol{\mu}}_t)^T + \mathbf{Q}_t \tag{S15}$$

In the observation update step, **S**, **K** and **i** from Equations S5, S6 and S7 are likewise replaced by the following:

$$\hat{\mathbf{z}}_t = \sum_i w_m^i h(\hat{\boldsymbol{\chi}}_t^i) \tag{S16}$$

$$\mathbf{S}_t = \sum_i w_c^i (h(\hat{\boldsymbol{\chi}}_t^i) - \hat{\mathbf{z}}_t)(h(\hat{\boldsymbol{\chi}}_t^i) - \hat{\mathbf{z}}_t)^T + \mathbf{R}_t \tag{S17}$$

$$\mathbf{K}_t = \sum_i w_c^i (\hat{\boldsymbol{\chi}}_t^i) - \hat{\boldsymbol{\mu}}_t)(h(\hat{\boldsymbol{\chi}}_t^i) - \hat{\mathbf{z}}_t)^T \mathbf{S}_t^{-1} \tag{S18}$$

$$\mathbf{i}_t = \mathbf{z}_t - \mathbf{H}\hat{\boldsymbol{\mu}}_t \tag{S19}$$

In theory, the UKF conveys the nonlinear transformation of the covariance more faithfully than the EKF and is thus better suited for strongly non-linear problems (Thrun et al., 2005). In contrast to the EKF, it also does not require computing the Jacobian of the process and observation models, which can be advantageous when those models are learned.

In practice, tuning the parameters of the UKF can, however, sometimes be challenging. If $\alpha^2(\kappa + n)$ is too big, the sigma points are spread too far from the mean and the prediction uncertainty increases. However, for $0 < \alpha^2(\kappa + n) < n$, the sigma point $\boldsymbol{\chi}^0$, which represents the original mean, is weighted negatively. This not only seems counter-intuitive, but strongly negative w^0 can also negatively affect the numerical stability of the UKF(Wu et al., 2006), which sometimes causes divergence of the estimated mean. In addition, if $\frac{\kappa}{n+\kappa} < 0$, the estimated covariance matrix is not guaranteed to be positive semi definite any more. This problem can be solved by changing the way in which $\boldsymbol{\Sigma}$ is computed (see Appendix III in Julier et al. (2000)).

A.4 Monte Carlo Unscented Kalman Filter (MCUKF)

The UKF represents the belief over the state with as few sigma points as possible. However, finding the correct scaling parameters α, κ and β can sometimes be difficult, especially if the state is high dimensional. Instead of relying on the Unscented Transform to calculate the mean and covariance of the next belief, we can also resort to Monte Carlo methods, as proposed by Wüthrich et al. (2016).

In practice, this means replacing the carefully constructed sigma points and their weights in Equations S11 and S12 with uniformly weighted samples from the current belief. The rest of the UKF algorithm remains the same, but more sampled pseudo sigma points are necessary to represent the distribution of the belief accurately.

A.5 Particle Filter (PF)

In contrast to the different variants of the Kalman filter explained before, the Particle filter Gordon et al. (1993) does not assume a parametric representation of the belief distribution. Instead, it represents the belief with a set of weighted *particles*. This allows the filter to track multiple hypotheses about the state at the same time and makes it a popular choice for tasks like localization or visual object tracking (Thrun et al., 2005).

Table S1: Sensor model and heteroscedastic observation noise architecture. Both fully connected output layers (for **z** and *diag*(**R**)) get fc 2's output as input.

Layer	Output Size	Kernel	Stride	Activation
Input **D**	100 × 100 × 3	-	-	-
conv 1	50 × 50 × 4	9 × 9	2	ReLU
conv 2	25 × 25 × 8	9 × 9	2	ReLU
fc 1	16	-	-	ReLU
fc 2	32	-	-	ReLU
z	2	-	-	-
diag(**R**)	2	-	-	-

An initial set of particles $\boldsymbol{\chi}_0^i \in \mathbf{X}_0$ is drawn from the initial belief and initialized with uniform weights π. In the prediction step, new particles are generated by applying the process model to the old particle set and sampling additive process noise:

$$\mathbf{X}_t = f(\mathbf{X}_{t-1}, \mathbf{u}_t, \mathbf{q}_t) \tag{S20}$$

In the observation update step, the weight π_t^i of each particle $\boldsymbol{\chi}_t^i$ is updated using current observation $\mathbf{z}_t$ by

$$\pi_t^i = \pi_{t-1}^i p(\mathbf{z}_t | \boldsymbol{\chi}_t^i) \qquad \forall \boldsymbol{\chi}_t^i \in \mathbf{X}_t \tag{S21}$$

A potential problem of the PF is particle deprivation: Over time, many particles will receive a very low likelihood $p(\mathbf{z}_t | \boldsymbol{\chi}_t^i)$, and eventually the state would be represented by too few particles with high weights. To prevent this, a new set of particles with uniform weights can be drawn (with replacement) from the old set according to the weights. This resampling step focuses the particle set on regions of high likelihood and is usually applied after each timestep.

B Extended Experiments: Simulated Disc Tracking

In the following, we present additional information about the experiments we performed for evaluating the DFs. This includes detailed information about the network architectures for each task, extended results and additional experiments.

B.1 Network Architectures and Initialization

The network architectures for the sensor model and heteroscedastic observation noise model are shown in Table S1. Tables S2 and S3 show the architecture for the learned process model and the heteroscedastic process noise. We denote fully connected layers by *fc* and convolutional layers by *conv.*

For the initial belief, we use $\boldsymbol{\Sigma}_{\text{init}} = 25 * \mathbf{I}_4$. When training from scratch, we initialize **Q** and **R** with $\mathbf{Q} = 100 * \mathbf{I}_4$ and $\mathbf{R} = 900 * \mathbf{I}_2$, reflecting the high uncertainty of the untrained models.

Table S2: Learned process model architecture

Layer	Output Size	Activation
Input $\mathbf{x}$	4	-
fc 1	32	ReLU
fc 2	64	ReLU
fc 3	64	ReLU
$\Delta\mathbf{x}$ (fc)	4	-

Table S3: Heteroscedastic process noise model architecture

Layer	Output Size	Activation
Input $\mathbf{x}$	4	-
fc 1	32	ReLU
fc 2	32	ReLU
diag($\mathbf{Q}$) (fc)	4	-

B.2 Implementation and Parameters

All experiments for evaluating different design choices and filter-specific parameters are performed on a dataset with 15 distractors and constant process noise ($\sigma_p = 0.1, \sigma_v = 2$). The filters are trained end-to-end on L_{NLL} and learn the sensor and process model as well as heteroscedastic observation and constant process noise models. We repeat each experiment two times to account for different initializations of the weights and report mean and standard errors.

B.2.1 dUKF

Experiment: The original version of the UKF by Julier and Uhlmann (1997) uses a simple parameterization where $\alpha = 1$ and $\beta = 0$ are fixed and only κ varies. The authors recommend setting $\kappa = 3 - n$. α and β are used in the later proposed *scaled unscented transform* (Julier, 2002), for which Van Der Merwe (2004) suggest setting $\kappa = 0$, $\beta = 2$ and α to a small positive value.

We evaluate the original, simple parameterization as well as the one for the scaled transform. For the first, we test training the dUKF with κ values in $[-10, 10]$. In the second case, we evaluate $\alpha \in \{0.001, 0.1, 0.5\}$ but do not vary β, for which the value 2 is optimal when working with Gaussians.

Results: As discussed in Section 6.2.1 of the main document, the results show no significant differences between the different parameter settings or between using the original parameterization from Julier and Uhlmann (1997) and the scaled transform. Only for $\kappa < -n$, the training failed due to a non-invertible matrix in the calculation of the Kalman Gain.

B.2.2 dMCUKF

The results discussed in Section 6.2.2 of the main document are visualized in Figure S1.

B.2.3 dPF: Belief Representation

The results discussed in Section 6.2.3 are visualized in Figure S2.

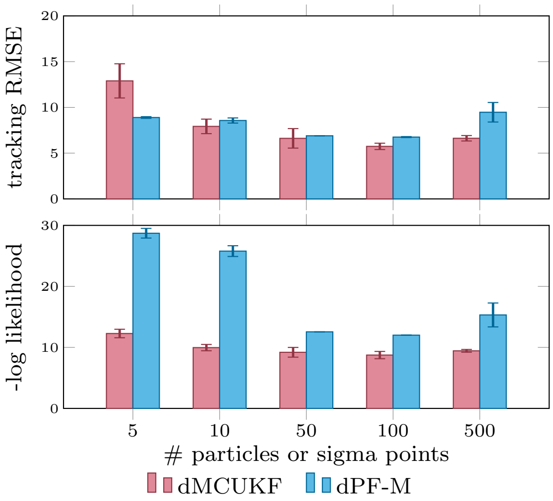

Fig. S1: Results on disc tracking: Tracking error and negative log likelihood of the dMCUKF and dPF-M for different numbers of sampled sigma points or particles during training and 500 sigma points / particles for testing.

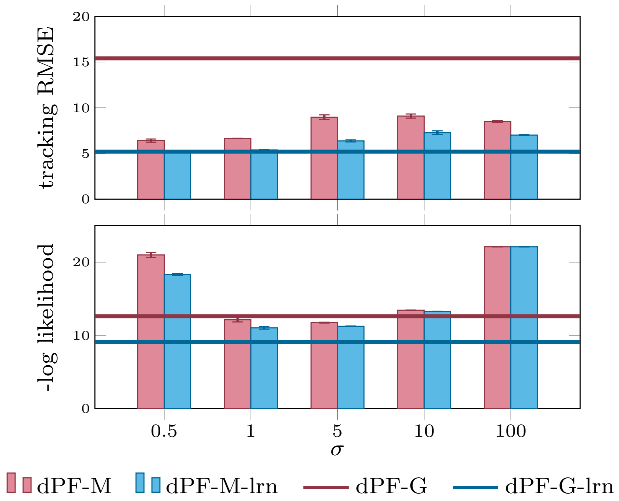

Fig. S2: Results on disc tracking: Tracking error and negative log likelihood of the dPF-M and dPF-G, each with using the analytical or learned (-lrn) observation update. The dPF-M and dPF-M-lrn are also evaluated for different values of the fixed per-particle covariance matrix $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$ in the GMM.

B.2.4 dPF: Observation Update

The likelihood for the observation update step of the dPF can be implemented with an analytical Gaussian likelihood function (dPF-(G/M)) or with a neural network (dPF-(G/M)-lrn) as in Jonschkowski et al. (2018) and Karkus et al. (2018).

Jonschkowski et al. (2018) predict the likelihood based on an encoding of the sensory data and the observable components of the (normalized) particle states. Our implementation, too, takes the 64-dimensional encoding of the raw observations (fc 3 in Table S1) and the observable particle state components as input. However, we decide not to normalize the particles, since having prior knowledge about the mean and standard deviation of each state component in the dataset might give an unfair advantage to the method over other variants.

Results: Results for comparing the learned and analytical observation update can be found in Figure S2. Using a learned instead of an analytical likelihood function for updating the particle weights improves the tracking error of the dPF-M from 10.3±0.1 to 8.3±0.1 and the NLL from 29.6±0.2 to 28.7±0.1. For the dPF-G, the difference is even more dramatic, with an RMSE of 23.3±1.1 vs. 8.0±0.3 and an NLL of 31.0±0.05 vs. 27.5±0.1.

B.2.5 dPF: Resampling

The results for Section 6.2.5 of the main document are visualized in Figure S3.

B.2.6 dPF: Number of Particles

The results discussed in Section 6.2.6 of the main document are visualized in Figure S1.

B.3 Noise Models

B.3.1 Heteroscedastic Observation Noise

Table S4 extends Table 1 in the main document. It contains results for the dPF-G and on additional datasets with different numbers of distractors and different magnitudes of the positional process noise.

B.3.2 Heteroscedastic Process Noise

Table S5 extends Table 2 in the main document. It contains results for the dPF-G and on additional datasets with different magnitudes of the positional process noise.

B.3.3 Correlated Noise

So far, we have only considered noise models with diagonal covariance matrices. In this experiment, we want to see if DFs can learn to identify correlations in the noise.

Experiment: We create a new dataset with 30 distractors and constant, correlated process noise. The ground truth process noise covariance matrix is

$$\mathbf{Q}_{gt} = \begin{pmatrix} 9. & -3.6 & 1.2 & 5.4 \\ -3.6 & 9. & -0.6 & 0. \\ 1.2 & -0.6 & 4. & 0. \\ 5.4 & 0. & 0. & 4. \end{pmatrix}$$

We compare the performance of DFs that learn noise models with diagonal or full covariance matrix on datasets with and without correlated process noise. Both the learned process and the observation noise model are also heteroscedastic.

Results: Results are shown in Table S6. Overall, we note that learning correlated noise models has a small but consistent positive effect on the tracking performance of all DFs, even when the ground truth noise is not correlated. On the dataset with correlated ground truth process noise, we also observe an improvement of the likelihood scores.

In terms of the Bhattacharyya distance between true and learned **Q**, learning correlated models leads to a slight improvement for correlated ground truth noise and to slightly worse scores otherwise. This indicates that the models are able to uncover some, but not all correlations in the underlying data.

In summary, while learning correlated noise models does not influence the results negatively, it also does not lead to a very pronounced improvement over models with diagonal covariance matrices. Uncovering correlations in the process noise thus seems to be even more difficult than learning accurate heteroscedastic noise models.

B.4 Benchmarking

Table S7 extends Table 3 from the main document. It contains results for the dPF-G and dPF-G-lrn and on additional datasets with lower positional process noise and heteroscedastic process noise.

C Extended Experiments: KITTI Visual Odometry

C.1 Network Architectures and Initialization

Sensor Network The network architectures for the sensor model and the heteroscedastic observation noise model are shown in Table S8. At each timestep, the input consists of the current RGB image and the difference image between the current and previous image. The network architecture for the sensor model is the same as was used in Haarnoja et al. (2016) and Jonschkowski et al. (2018).

Process Model Tables S9 and S10 show the architecture for the learned process model and the heteroscedastic process noise. For both models, we found it to be important not to include the absolute position of the vehicle in the input values: The value range for the positions is not bounded, and especially for the dUKF variants, novel values encountered at test time often lead to a divergence of the filter.

Excluding these values from the network inputs for predicting the state update also makes sense intuitively, since they are not required for computing the update analytically, either. For the state-dependent process noise, we not only exclude the position, but also the orientation of the car, as any relationships between vehicle pose and noise that could be learned would be specific to the training trajectories.

In addition, we provide the process model with the sine and cosine of θ as input instead of using the raw orientation, to facilitate the learning. In general, dealing with angles in the state vector requires special attention: First, we correct angles to the range between $[-\pi, \pi]$ after every operation on the state vector. Second, it is important to correctly calculate the difference between angles (e.g. in the loss function) to avoid differences over 180deg. And third, computing the mean of several angles, e.g. for the particle mean in the dPF, requires converting the angles to a vector representation.

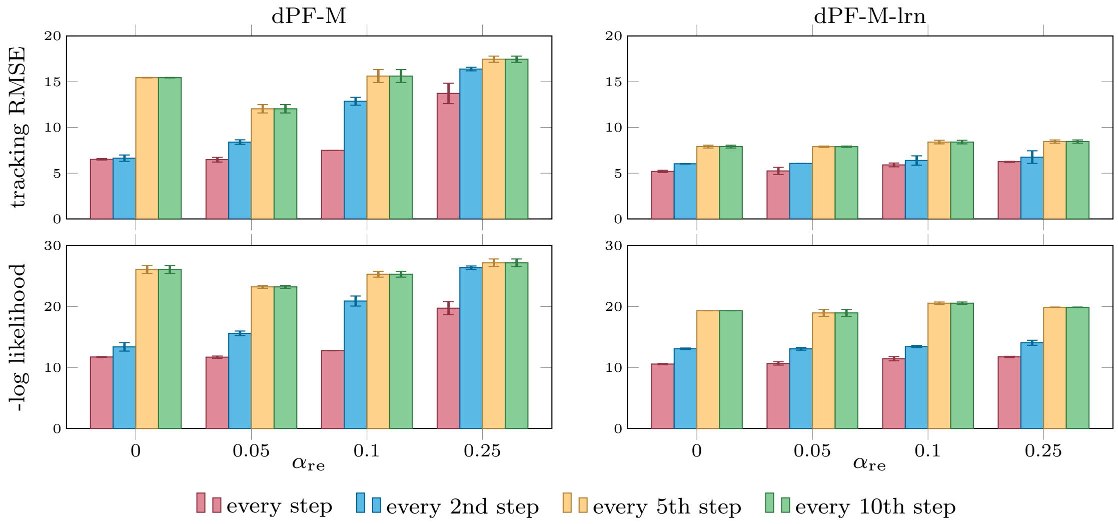

Fig. S3: Results on disc tracking: Tracking error and negative log likelihood of the two dPF-M variants for different resampling rates and values of the soft resampling parameter α_{re}.

Table S4: Results for disc tracking: End-to-end learning of the noise models through the DFs on datasets with 5 or 30 distractors and different levels of process noise. While $\mathbf{Q}$ is always constant, we evaluate learning constant (const.) or heteroscedastic (hetero) observation noise $\mathbf{R}$. We show the tracking error (RMSE), negative log likelihood (NLL), the correlation coefficient between predicted $\mathbf{R}$ and the number of visible pixels of the target disc (corr.) and the Bhattacharyya distance between true and learned process noise model ($D_{\mathbf{Q}}$). The best results per DF are highlighted in bold.

				$\sigma_{q_p} = 0.1$				$\sigma_{q_p} = 3.0$				$\sigma_{q_p} = 9.0$		
		R	RMSE	NLL	corr.	$D_{\mathbf{Q}}$	RMSE	NLL	corr.	$D_{\mathbf{Q}}$	RMSE	NLL	corr.	$D_{\mathbf{Q}}$
5 distractors	dEKF	const.	14.1	13.6	-	2.722	16.3	14.1	-	0.081	28.8	15.7	-	0.019
		hetero.	**9.8**	**11.9**	-0.71	**1.204**	**9.8**	**11.5**	-0.74	**0.007**	**18.7**	**13.2**	-0.66	**0.007**
	dUKF	const.	14.3	13.7	-	2.828	17.1	14.2	-	0.071	30.2	15.8	-	0.026
		hetero.	**9.9**	**11.8**	-0.70	**0.557**	**9.6**	**11.3**	-0.74	**0.011**	**21.7**	**14.2**	-0.66	**0.013**
	dMCUKF	const.	14.5	13.7	-	2.389	16.5	14.2	-	0.258	30.7	15.8	-	0.02
		hetero.	**9.9**	**11.8**	-0.71	**0.272**	**9.9**	**11.6**	-0.73	**0.016**	**21.0**	**14.4**	-0.65	**0.004**
	dPF-G	const.	14.6	13.7	-	**3.318**	17.3	14.1	-	**0.257**	29.2	15.7	-	**0.04**
		hetero.	**12.0**	**12.8**	-0.47	3.348	**13.8**	**13.4**	-0.47	0.297	**23.2**	**14.6**	-0.63	0.064
	dPF-M	const.	13.1	34.7	-	3.408	15.2	40.8	-	**0.279**	27.7	52.9	-	0.745
		hetero.	**10.3**	**19.8**	-0.7	**3.361**	**11.3**	**23.1**	-0.67	0.424	**18.0**	**36.1**	-0.74	**0.147**
30 distractors	dEKF	const.	14.5	13.9	-	2.543	16.2	14.0	-	0.121	28.6	15.6	-	0.010
		hetero.	**7.8**	**10.4**	-0.72	**1.429**	**8.8**	**10.7**	-0.78	**0.002**	**22.4**	**14.7**	-0.75	**0.008**
	dUKF	const.	15.3	13.9	-	2.047	16.8	14.1	-	0.161	30.2	15.7	-	0.024
		hetero.	**7.8**	**10.4**	-0.71	**1.565**	**8.8**	**10.7**	-0.78	**0.013**	**20.6**	**14.8**	-0.85	**0.010**
	dMCUKF	const.	14.8	13.9	-	2.955	16.7	14.1	-	0.152	29.8	15.7	-	0.022
		hetero.	**7.8**	**10.4**	-0.71	**1.533**	**9.0**	**10.9**	-0.78	**0.006**	**22.1**	**15.1**	-0.78	**0.016**
	dPF-G	const.	15.2	13.8	-	3.433	17.3	14.1	-	**0.224**	29.1	15.6	-	**0.047**
		hetero.	**10.4**	**11.9**	-0.71	**3.103**	**12.1**	**12.6**	-0.53	0.277	**20.8**	**14.4**	-0.86	0.090
	dPF-M	const.	14.1	33.6	-	3.396	16.1	34.3	-	0.435	27.7	49.9	-	1.240
		hetero.	**8.7**	**14.4**	-0.70	**3.223**	**9.6**	**20.8**	-0.77	**0.280**	**13.9**	**21.8**	-0.81	**0.084**

Table S5: Results on disc tracking: End-to-end learning of constant or heteroscedastic process noise $\mathbf{Q}$ on datasets with 30 distractors and different heteroscedastic or constant ($\sigma_{q_p} = 3.0$, $\sigma_{q_v} = 2.0$) process noise. $D_{\mathbf{Q}}$ is the Bhattacharyya distance between true and learned process noise.

		hetero. $\sigma_{q_v}, \sigma_{q_p} = 0.1$			hetero. $\sigma_{q_v}, \sigma_{q_p} = 3.0$			$\sigma_{q_p} = 3.0$, $\sigma_{q_v} = 2.0$			hetero. $\sigma_{q_v}, \sigma_{q_p} = 9.0$		
	Q	RMSE	NLL	$D_{\mathbf{Q}}$	RMSE	NLL	$D_{\mathbf{Q}}$	RMSE	NLL	$D_{\mathbf{Q}}$	RMSE	NLL	$D_{\mathbf{Q}}$
dEKF	const.	4.3	8.2	1.335	8.1	11.6	0.879	8.8	10.7	**0.002**	19.6	14.1	1.108
	hetero.	**3.8**	**7.2**	**0.479**	**7.4**	**11.3**	**0.402**	8.8	10.7	0.033	**18.5**	**13.7**	**0.805**
dUKF	const.	4.23	8.3	1.288	7.8	11.3	0.874	8.8	10.7	**0.013**	20.3	14.2	1.061
	hetero.	**3.8**	**7.2**	**1.008**	**7.6**	**11.2**	**0.391**	**8.7**	10.7	0.030	**20.1**	**14.0**	**0.900**
dMCUKF	const.	4.2	8.3	1.184	8.1	11.5	0.891	9.0	10.9	**0.006**	20.7	14.2	1.057
	hetero.	**3.8**	**7.2**	**0.932**	**7.5**	**11.3**	**0.464**	**8.7**	**10.7**	0.044	**20.3**	**13.9**	**0.904**
dPF-G	const.	7.2	10.9	**3.888**	9.0	11.9	1.104	12.1	12.6	**0.277**	**20.9**	14.3	1.229
	hetero.	**6.8**	**10.8**	3.990	9.0	**11.5**	**0.808**	**11.8**	**12.4**	0.347	21.3	14.3	**1.096**
dPF-M	const.	**5.0**	**10.7**	3.902	8.5	15.2	1.072	**9.6**	20.8	**0.280**	19.7	29.1	**0.799**
	hetero.	5.2	11.1	**3.853**	**8.2**	**14.7**	**0.787**	9.8	**19.8**	0.413	**17.8**	**27.8**	1.074

Table S6: Results on disc tracking: End-to-end learning of independent (*diagonal* covariance matrix) or correlated (*full* covariance matrix) process and observation noise models. We evaluate on one dataset with independent, constant process noise ($\sigma_{q_p} = 3.0$, $\sigma_{q_v} = 2.0$), one with independent heteroscedastic process noise ($\sigma_{q_p} = 3.0$), and one with correlated constant process noise. $D_{\mathbf{Q}}$ is the Bhattacharyya distance between true and learned $\mathbf{Q}$.

	covariance matrix	independent const. noise			independent hetero. noise			correlated const. noise		
		RMSE	NLL	$D_{\mathbf{Q}}$	RMSE	NLL	$D_{\mathbf{Q}}$	RMSE	NLL	$D_{\mathbf{Q}}$
dEKF	diagonal	8.8	10.7	**0.033**	**7.4**	**11.3**	**0.402**	8.9	10.6	1.249
	full	**8.6**	**10.7**	0.089	7.6	12.2	0.591	**8.4**	**10.1**	**1.003**
dUKF	diagonal	8.7	10.7	**0.030**	**7.6**	11.2	**0.391**	8.7	10.6	1.345
	full	**8.6**	**10.7**	0.126	7.6	**10.8**	0.523	**8.7**	**10.4**	**0.994**
dMCUKF	diagonal	8.7	10.7	**0.044**	**7.5**	11.3	**0.464**	8.8	10.6	1.248
	full	**8.6**	**10.7**	0.143	7.6	**10.8**	0.507	**8.7**	**10.3**	**1.026**
dPF-G	diagonal	11.8	12.4	**0.347**	9.0	11.5	**0.808**	11.7	**12.4**	1.646
	full	**11.5**	**12.3**	0.421	**8.8**	11.5	0.942	**11.4**	12.5	**1.565**
dPF-M	diagonal	9.8	19.8	**0.413**	8.2	**14.7**	**0.787**	9.3	22.1	**1.649**
	full	**8.9**	**18.5**	0.693	**7.3**	15.7	1.463	**8.4**	**18.2**	2.005

Table S7: Results on disc tracking: Comparison between the DFs and LSTM models with one or two LSTM layers on two different datasets with 30 distractors and constant process noise with increasing magnitude. Each experiment is repeated two times and we report mean and standard error.

	$\sigma_{q_p} = 0.1$		$\sigma_{q_p} = 0.1$ hetero.		$\sigma_{q_p} = 3.0$		$\sigma_{q_p} = 9.0$	
	RMSE	NLL	RMSE	NLL	RMSE	NLL	RMSE	NLL
dEKF	6.1±0.54	9.1±0.42	4.9±1.04	**8.3±0.91**	6.3±0.12	9.2±0.10	11.8±0.28	11.1±0.20
dUKF	5.8±0.21	8.9±0.25	4.1±0.09	7.6±0.06	6.5±0.20	9.3±0.26	11.5±0.18	10.8±0.16
dMCUKF	5.5±0.30	**8.6±0.23**	5.0±0.86	8.4±0.77	6.5±0.18	**9.2±0.17**	11.6±0.10	**10.8±0.11**
dPF-G	12.9±0.29	12.3±0.05	10.4±0.06	11.4±0.02	13.3±0.45	12.4±0.10	18.7±0.35	13.5±0.07
dPF-M	6.2±0.34	11.7±0.16	5.0±0.40	10.7±0.45	6.7±0.07	12.3±0.09	11.5±0.07	20.5±0.36
dPF-G-lrn	**4.9±0.11**	9.2±0.12	**3.6±0.13**	8.4±0.04	**5.7±0.06**	9.8±0.01	10.6±0.17	11.9±0.03
dPF-M-lrn	5.3±0.17	10.8±0.22	4.4±0.09	9.9±0.10	5.9±0.15	11.4±0.15	**10.0±0.13**	19.2±0.18
LSTM-1	5.9±0.20	9.0±0.21	14.2±9.33	10.5±2.52	9.4±0.77	10.6±0.25	14.6±0.70	11.8±0.22
LSTM-2	5.7±0.40	9.2±0.62	6.3±2.73	8.9±1.51	7.1±0.86	9.8±0.56	13.9±0.51	11.9±0.07

Table S8: Sensor model and heteroscedastic observation noise architecture. Both output layers (for $\mathbf{z}$ and $\text{diag}(\mathbf{R})$) get fc 2's output as input.

Layer	Output Size	Kernel	Stride	Activation	Normalization
Input $\mathbf{D}$	$50 \times 150 \times 6$	-	-	-	-
conv 1	$50 \times 150 \times 16$	7×7	1×1	ReLU	Layer
conv 2	$50 \times 75 \times 16$	5×5	1×2	ReLU	Layer
conv 3	$50 \times 37 \times 16$	5×5	1×2	ReLU	Layer
conv 4	$25 \times 18 \times 16$	5×5	2×2	ReLU	Layer
dropout (0.3)	$25 \times 18 \times 16$	-	-	-	-
fc 1	128	-	-	ReLU	-
fc 2	128	-	-	ReLU	-
$\mathbf{z}$ (fc)	2	-	-	-	-
$\text{diag}(\mathbf{R})$ (fc)	2	-	-	-	-

Table S9: Learned process model architecture. We use a modified version of the previous state $\mathbf{x}$ as input: $\bar{\mathbf{x}} = (v, \dot{\theta}, \cos\theta, \sin\theta)$

Layer	Output Size	Activation
Input $\bar{\mathbf{x}}$	4	-
fc 1	32	ReLU
fc 2	64	ReLU
fc 3	64	ReLU
$\Delta\mathbf{x}$ (fc)	5	-

Table S10: Heteroscedastic process noise model architecture. We use a modified version of the previous state $\mathbf{x}$ as input: $\bar{\mathbf{x}} = (v, \dot{\theta}, \cos\theta, \sin\theta)$

Layer	Output Size	Activation
Input $\left(v, \dot{\theta}\right)$	2	-
fc 1	32	ReLU
fc 2	32	ReLU
$\text{diag}(\mathbf{Q})$ (fc)	5	-

Initialization When creating the noisy initial states, we do not add noise to the absolute position and orientation of the vehicle, since the DFs have no way of correcting them. We use $\text{diag}(\boldsymbol{\Sigma}_\text{init}) = (0.01\ 0.01\ 0.01\ 25\ 25)$ for the initial covariance matrix. When training the DFs from scratch, we initialize the covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ with $\text{diag}(\mathbf{Q}) = (0.01\ 0.01\ 0.01\ 100\ 100)$ and $\mathbf{R} = 100\mathbf{I}_2$. This reflects the high uncertainty of the untrained models, but also the fact that the process noise should be higher for the velocities (to account for the unknown driver actions) than for the absolute pose.

C.2 Training Sequence Length and Filter Parameters

One special feature of the Visual Odometry task is that the the error on the estimated absolute vehicle pose will inevitably grow during filtering. As this could have an effect on the ideal training sequence length, we repeat the experiment from Section 6.4 in the main document.

For the dPF-M, we also evaluate different values of the fixed per-particle covariance $\boldsymbol{\Sigma}$ for calculating the GMM-likelihood. We anticipate that this parameter, too, could be sensitive to the accumulating uncertainty in the problem.

In addition, we also reevaluate different values for parameterizing the sigma point selection and weighting in the dUKF.

C.2.1 Training Sequence Length and dPF-M

Experiment: We only test with the dEKF, dUKF, dPF-M and dPF-M-lrn on *KITTI-10.* The filters learn the sensor and process model as well as constant noise models. We train them using L_{NLL} on sequence lengths $k \in \{2, 5, 10, 25\}$ while keeping the total number of examples per batch (steps × batch size) constant.

For the dPF-M, we also evaluate two different values of the per-particle covariance, $\boldsymbol{\Sigma} = \mathbf{I}$ and $\boldsymbol{\Sigma} = 5^2\mathbf{I}$.

Results: The results shown in Figure S4 largely confirm the results obtained for the simulation dataset in Section 6.4 of the main document. We again see that longer training sequences increase the tracking performance of all DFs up to a sequence length of around $k = 10$.

The dUKF seems to be most sensitive to the sequence length, with the highest tracking error and an extremely bad NLL score for sequences of length 2. Different from the simulation experiment, for both dEKF and dUKF, the NLL keeps decreasing strongly over the full evaluated sequence length range, despite the best RMSE already being reached at $k = 5$. We attribute this to the accumulating uncertainty about the vehicle pose. For the dPFs, in contrast, the likelihood behaves similarly to the RMSE.

In light of the longer training times with higher sequence lengths, we again decide to keep a training-sequence length of 10 when training the DFs from scratch. However, when only the noise models are trained, longer sequences can be used to improved results on the NLL.

For the dPF-M, the experiment also shows that the covariance of the single distributions in the GMM is an important tuning parameter. With $\boldsymbol{\Sigma} = \mathbf{I}$, we achieve the best tracking error, however, the likelihood does not reach the performance of dEKF and dUKF. The NLL values can be drastically improved by using larger $\boldsymbol{\Sigma}$, at the cost of a decreased tracking performance. Visual inspection of the position estimates shows that the particles remain relatively tightly clustered over the complete sequence, such that the likelihood of the GMM is

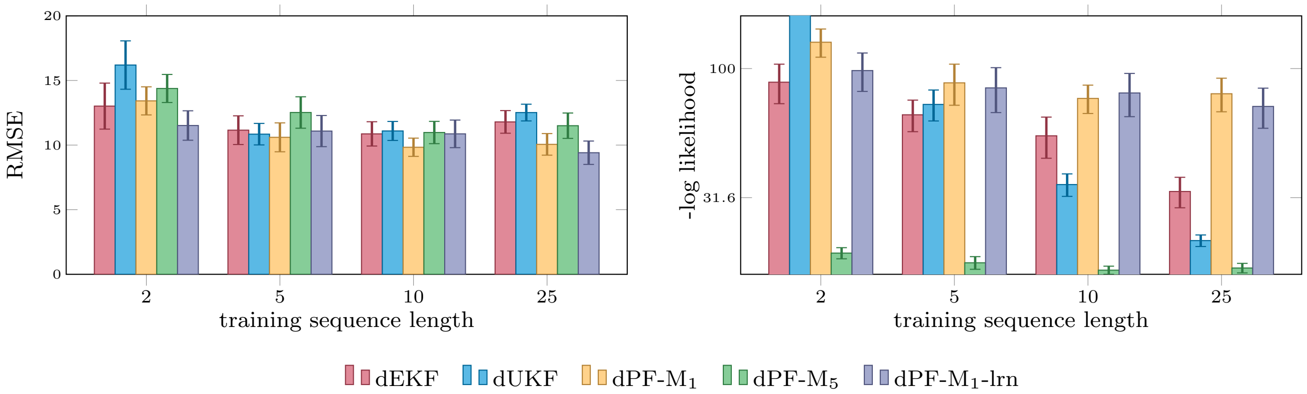

Fig. S4: Results on *KITTI-10*: Tracking error and negative log likelihood (NLL with logarithmic y axis) of dEKF, dUKF, dPF-M and dPF-M-lrn trained with different sequence lengths. For the dPF-M, we show two different values for the covariance $\boldsymbol{\Sigma}$ of the single Gaussians in the mixture model, $\boldsymbol{\Sigma} = \mathbf{I}$ and $\boldsymbol{\Sigma} = 5^2\mathbf{I}$. The cut-off NLL value for the dUKF on sequences of length 2 is 2004.4±518.3.

not so different from the likelihood of the individual Gaussian components.

This clustered particle distribution can be explained by the characteristics of the task: The uncertainty in the system mainly stems from the velocity components that are affected by the unknown actions. However, by applying the observation update and resampling the particles at every step, we keep the variance in the velocity components small and thus prevent a stronger diffusion of the unobserved position components. This also explains why the dPF cannot profit as much as the dUKF and dEKF from seeing longer sequences during training.

The large influence of the tuning parameter $\boldsymbol{\Sigma}$ on the value of the likelihood, independent of the tracking performance, also shows that comparing likelihood scores between different probabilistic models can be difficult. In light of this, we decide to keep using $\boldsymbol{\Sigma} = \mathbf{I}$ for the better tracking error.

C.2.2 dUKF

We also repeat the evaluation of different values of the parameters α, κ and β for the dUKF described in Experiment B.2.1. The experiment confirms our finding from the simulation experiment that the exact choice of the values does not have a significant effect on the filter performance. We thus keep the values at $\alpha = 1$, $\kappa = 0.5$ and $\beta = 0$.

C.3 Learning Noise Models

Experiment: The baseline model with constant, hand-tuned noise uses $\text{diag}(\mathbf{Q}) = (10^{-4}\ 10^{-4}\ 10^{-6}\ 0.01\ 0.16)^T$ and $\text{diag}(\mathbf{R}) = (0.36\ 0.36)^T$.

C.4 Benchmarking

Table S11 extends the results from Table 6 with data for the dPF-G and dPF-G-lrn. Interestingly, we find that the difference in performance between the dPF variants with learned or analytical observation update is not as pronounced as in the results we obtained for the simulation experiment (Section B.2.4). In particular, the dPF-G-lrn performs similarly bad as the dPF-G on this task.

D Extended Experiments: Planar Pushing

D.1 Network Architectures and Initialization

Sensor Network Our architectures for the sensor network is very similar to the one used by Kloss et al. (2020), where only the object position $\mathbf{p}_o$ is estimated from the full image while the contact-related state components ($\mathbf{r}$, $\mathbf{n}$, s) are computed from a smaller glimpse around the pusher location.

For predicting the orientation of the object, we extract a second glimpse from the full image, this time centered on the estimated object position. A small CNN then predicts the change in orientation between the glimpse extracted from the initial image in the sequence and the glimpse at the current time step.

The sensor network predicts object position, contact point and normal in pixel space because predictions in this space can be most directly related to the input image and the predicted feature maps. To this end, we also transform the action into pixel space before using it (together with the glimpse encoding) as input for predicting the contact point and normal. The pixel predictions are then transformed back to to world-coordinates using the depth measurements and camera information. The resulting sensor network including the layers for computing the heteroscedastic observation noise is illustrated in Figure S5.

Process Model Tables S12 and S13 show the architecture for the learned process model and the heteroscedastic process noise. One problem we noticed is that the estimates for l sometimes diverge during filtering if the DFs estimate that the pusher is in contact with the object while it is not. Just as for the absolute position of the vehicle in the KITTI task, we thus found it important for the stability of the dUKF and dMCUKF to not make the heteroscedastic process noise model dependent on l.

Note that in the filter state, we measure $\mathbf{p}_o$ and $\mathbf{r}$ in millimeter and θ and α_m in degree. To avoid having too large

Table S11: Results on KITTI: Comparison between the DFs and LSTM (mean and standard error). Numbers for prior work BKF*, LSTM* taken from Haarnoja et al. (2016) and DPF* taken from Jonschkowski et al. (2018). BKF* and DPF* use a fixed analytical process model while our DFs learn both, sensor and process model. $\frac{\mathrm{m}}{\mathrm{m}}$ and $\frac{\mathrm{deg}}{\mathrm{m}}$ denote the translation and rotation error at the final step of the sequence divided by the overall distance traveled.

		RMSE	NLL	$\frac{\mathrm{m}}{\mathrm{m}}$	$\frac{\mathrm{deg}}{\mathrm{m}}$
KITTI-11	dEKF	15.8±5.8	338.8±277.1	0.24±0.04	0.080±0.005
	dUKF	14.9±5.7	326.7±267.5	**0.21 ± 0.04**	0.079±0.008
	dMCUKF	15.2±5.5	266.3±216.1	0.23±0.04	0.083±0.012
	dPF-M	16.3±6.1	115.2±34.6	0.24±0.04	**0.078 ± 0.006**
	dPF-G	21.1±5.7	121.9±80.5	0.33±0.04	0.175±0.036
	dPF-M-lrn	**14.3 ± 5.2**	**94.2 ± 33.3**	0.22±0.04	0.088±0.013
	dPF-G-lrn	19.1±5.3	197.8±125.3	0.31±0.06	0.168±0.049
	LSTM	25.7±5.7	3970.6±2227.4	0.55±0.05	0.081±0.008
	LSTM*	-	-	0.26	0.29
	BKF*	-	-	0.21	0.08
	DPF*	-	-	0.15±0.015	0.06±0.009
KITTI-10	dEKF	10.1±0.8	61.8±7.7	0.21±0.03	0.079±0.006
	dUKF	9.3±0.6	59.3±7.2	**0.18 ± 0.02**	0.080±0.008
	dMCUKF	9.7±0.6	**50.3 ± 8.1**	0.2 ±0.03	0.082±0.013
	dPF-M	10.2±0.9	82.4±12.2	0.21±0.02	**0.077 ± 0.007**
	dPF-G	15.5±1.4	41.7±6.6	0.3 ±0.04	0.182±0.038
	dPF-M-lrn	**9.2 ± 0.7**	61.3±6.1	0.19±0.03	0.090±0.014
	dPF-G-lrn	14.4±2.4	73.6±17.9	0.29±0.06	0.179±0.053
	LSTM	20.2±2.0	1764.6±340.4	0.54±0.06	0.079±0.008

Table S12: Learned process model architecture.

Layer	Output Size	Activation
Input $(\mathbf{x}, \mathbf{v}_u)$	12	-
fc 1	256	ReLU
fc 2	128	ReLU
fc 3	128	ReLU
$\Delta\mathbf{x}$ (fc)	10	-

Table S13: Heteroscedastic process noise model architecture. We use a modified version of the previous state $\mathbf{x}$ as input: $\bar{\mathbf{x}}$ does not include the latent parameter l.

Layer	Output Size	Activation
Input $(\bar{\mathbf{x}}, \mathbf{v}_u)$	11	-
fc 1	128	ReLU
fc 2	64	ReLU
$\mathrm{diag}(\mathbf{Q})$ (fc)	10	-

differences between the magnitudes of the state components, we downscale l by a factor of 100. $\mathbf{n}$ is a dimensionless unit vector and s should take values between 0 and 1.

To keep the filters stable during training, we found it necessary to enforce maximum and minimum values for α_m and l. Both α_m and l cannot become negative. The opening angle of the friction cone, α_m, should also not be larger than 90°, while we limit l to be in the range of $[0.1, 5000]$ to ensure that the computations in the analytical model remain numerically stable.

Initialization For the initial covariance matrix, we use $\sqrt{\mathrm{diag}(\boldsymbol{\Sigma}_{\mathrm{init}})} = (50\ 50\ 10^{-3}\ 5\ 5\ 50\ 50\ 0.5\ 0.5\ 0.5)^T$. When training the noise models, we initialize $\mathbf{Q}$ and $\mathbf{R}$ with $\mathrm{diag}(\mathbf{Q}) = \mathbf{I}_{10}$ and $\mathbf{R} = \mathbf{I}_8$.

D.2 Learning Noise Models

Experiment: The diagonals of the hand-tuned models are $\sqrt{\mathrm{diag}(\mathbf{Q})} = (0.23\ 0.23\ 0.37\ 0.01\ 0.01\ 0.7\ 0.7\ 0.1\ 0.1\ 0.13)^T$ and $\sqrt{\mathrm{diag}(\mathbf{R})} = (3.0\ 2.5\ 8.8\ 3.3\ 1.0\ 0.1\ 0.1\ 0.3)^T$.

Results: Table S14 extends the results from Table 7 with data for the dPF-G. In contrast to the other DF variants, learning complex noise models for the pushing task is not successful for the dPF-G. While the NLL can be further decreased when the noise models are heteroscedastic instead of constant, this comes at the cost of a significantly decreased tracking performance.

D.3 Benchmarking

Table S15 extends the results from Table 8 with data for the dPF-G and dPF-G-lrn. Note that their difference in performance to the dPF-M variants is smaller here than for the previous tasks because training on L_{mix} instead of L_{NLL} reduces the effect of how the belief is represented on the loss.

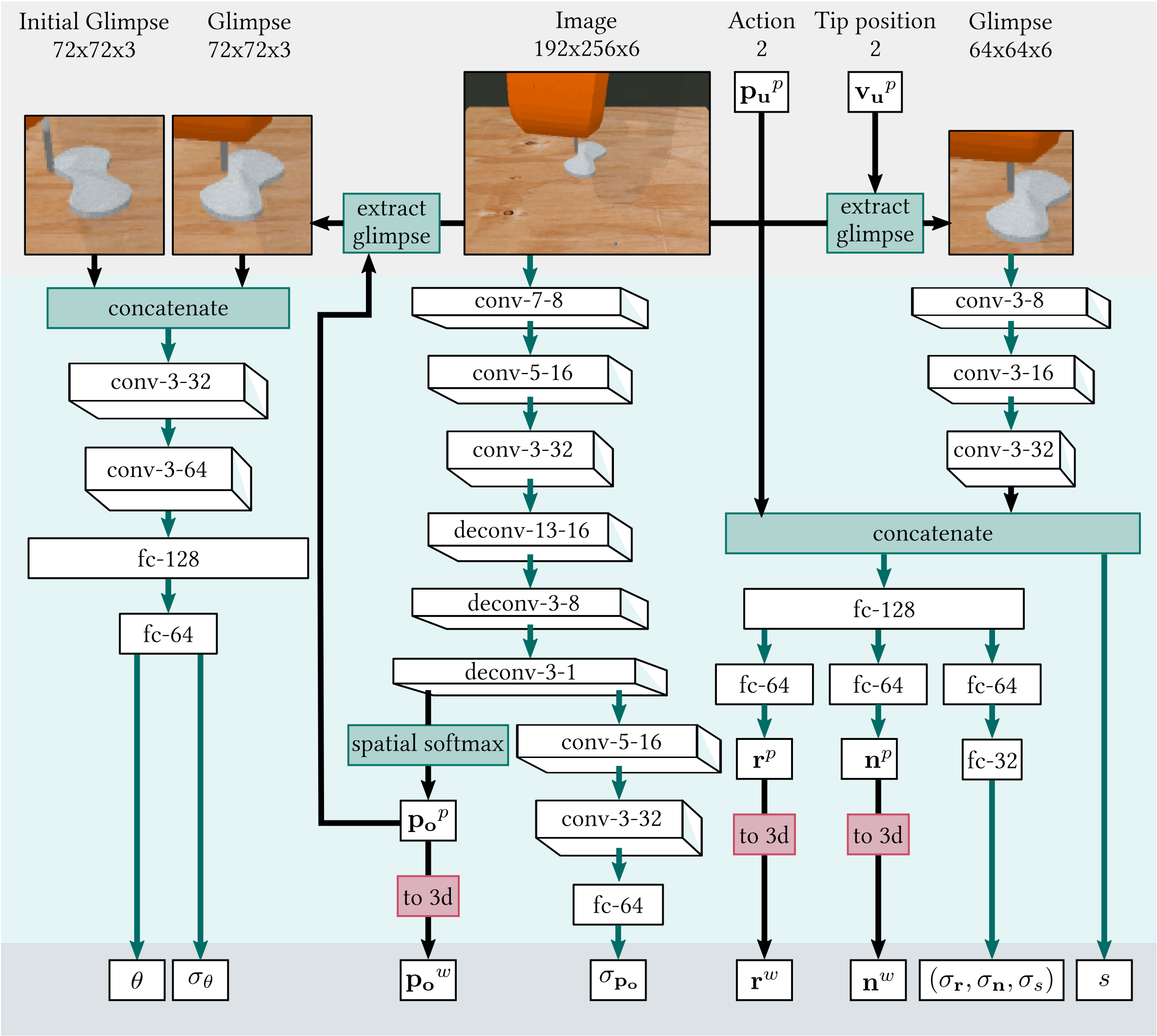

Fig. S5: Architecture of the sensor network and heteroscedastic observation noise model for planar pushing. We use 6-channel RGBXYZ images as input for computing the object position and contact related state components. The object orientation is estimated relative to the initial orientation by comparing the RGB glimpse centered on the current estimated object position to the initial one.
White boxes represent tensors, green arrows and boxes indicate network layers, whereas black arrows represent dataflow without processing. For convolution (conv) and deconvolution (deconv) layers, the numbers in each tensor are the kernel size and number of output channels of the layer that produced it. For fully connected layers (fc), the number corresponds to the number of output channels.
With the exception of the output layers, all convolution, deconvolution and fully connected layers are followed by ReLU non-linearities. The (de)convolution layers also use layer normalization.

References

Gordon NJ, Salmond DJ, Smith AF (1993) Novel approach to nonlinear/non-gaussian bayesian state estimation. In: IEE Proceedings F (Radar and Signal Processing), IET, vol 140, pp 107–113

Haarnoja T, Ajay A, Levine S, Abbeel P (2016) Backprop KF: Learning discriminative deterministic state estimators. In: Advances in Neural Information Processing Systems, pp 4376–4384

Jonschkowski R, Rastogi D, Brock O (2018) Differentiable particle filters: End-to-end learning with algorithmic priors. In: Robotics: Science and Systems, Pittsburgh, USA

Julier S, Uhlmann J, Durrant-Whyte HF (2000) A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Transactions on Automatic Control 45(3):477–482

Julier SJ (2002) The scaled unscented transformation. In: American Control Conference, vol 6, pp 4555–4559 vol.6

Julier SJ, Uhlmann JK (1997) New extension of the kalman filter to nonlinear systems. Proc SPIE 3068:3068 – 3068 –

Table S14: Results for planar pushing: Translation (tr) and rotation (rot) error and negative log likelihood for the DFs with different noise models (mean and standard error). The hand-tuned DFs use fixed noise models whereas for the other variants, the noise models are trained end-to-end through the DFs. $\mathbf{R}_c$ indicates a constant observation noise model and $\mathbf{R}_h$ a heteroscedastic one (same for $\mathbf{Q}$). The best result per DF and metric is highlighted in bold.

		Hand-tuned $\mathbf{R}_c\mathbf{Q}_c$	$\mathbf{R}_c\mathbf{Q}_c$	$\mathbf{R}_h\mathbf{Q}_c$	$\mathbf{R}_c\mathbf{Q}_h$	$\mathbf{R}_h\mathbf{Q}_h$
tr [mm]	dEKF	6.22	4.45	4.61	4.44	**4.38**
	dUKF	4.87	4.44	5.25	**4.43**	4.45
	dMCUKF	4.73	4.42	4.8	4.39	**4.35**
	dPF-M	18.13	5.07	4.92	5.32	**4.64**
	dPF-G	17.95	**5.48**	35.57	210.45	10.92
rot [°]	dEKF	10.49	10.00	**9.71**	10.15	9.97
	dUKF	9.87	9.91	**9.73**	10.05	10.00
	dMCUKF	**9.78**	9.95	9.93	10.04	9.85
	dPF-M	16.18	10.18	**9.92**	10.39	10.06
	dPF-G	16.56	10.27	11.27	43.41	**10.25**
NLL	dEKF	265.17	126.69	33.09	79.24	**26.48**
	dUKF	378.08	84.12	33.06	81.55	**27.61**
	dMCUKF	130.22	78.53	30.43	64.12	**30.1**
	dPF-M	353.25	128.15	104.40	103.21	**82.46**
	dPF-G	> 16m	12,089.71	34.18	5,789.83	**31.60**

Table S15: Results on pushing: Comparison between the DFs and LSTM. Process and sensor model are pretrained and get finetuned end-to-end. The DFs learn heteroscedastic noise models. Each experiment is repeated three times and we report mean and standard errors.

	RMSE	NLL	tr [mm]	rot [°]
dEKF	14.9±0.46	33.9±3.86	**3.5 ± 0.02**	8.8±0.22
dUKF	**13.7 ± 0.15**	**31.1 ± 1.90**	3.7±0.06	8.8±0.14
dMCUKF	13.8±0.10	34.1±3.57	3.7±0.06	**8.8 ± 0.06**
dPF-M	18.3±0.38	120.4±5.70	5.7±0.16	10.5±0.36
dPF-G	23.2±3.60	35.8±1.86	6.9±1.43	11.9±0.67
dPF-M-lrn	29.0±0.73	486.0±3.27	12.0±0.78	18.9±0.04
dPF-G-lrn	29.2±0.67	40.8±0.82	10.9±0.27	19.9±0.52
LSTM	27.36±0.2	35.4±0.24	8.8±0.17	19.0±0.001

12, DOI 10.1117/12.280797

Kalman RE (1960) A new approach to linear filtering and prediction problems. Journal of Basic Engineering 82(1):35–45

Karkus P, Hsu D, Lee WS (2018) Particle filter networks with application to visual localization. In: Conference on Robot Learning, pp 169–178

Kloss A, Schaal S, Bohg J (2020) Combining learned and analytical models for predicting action effects from sensory data. The International Journal of Robotics Research DOI 10.1177/0278364920954896

Sorenson H (1985) Kalman Filtering: Theory and Application. IEEE Press selected reprint series, IEEE Press

Thrun S, Burgard W, Fox D (2005) Probabilistic robotics. MIT press

Van Der Merwe R (2004) Sigma-point kalman filters for probabilistic inference in dynamic state-space models

Wu Y, Hu D, Wu M, Hu X (2006) A numerical-integration perspective on gaussian filters. IEEE Transactions on Signal Processing 54:2910–2921, DOI 10.1109/TSP.2006.875389

Wüthrich M, Garcia Cifuentes C, Trimpe S, Meier F, Bohg J, Issac J, Schaal S (2016) Robust gaussian filtering using a pseudo measurement. In: Proceedings of the American Control Conference, Boston, MA, USA