TEXAS
The University of Texas at Austin

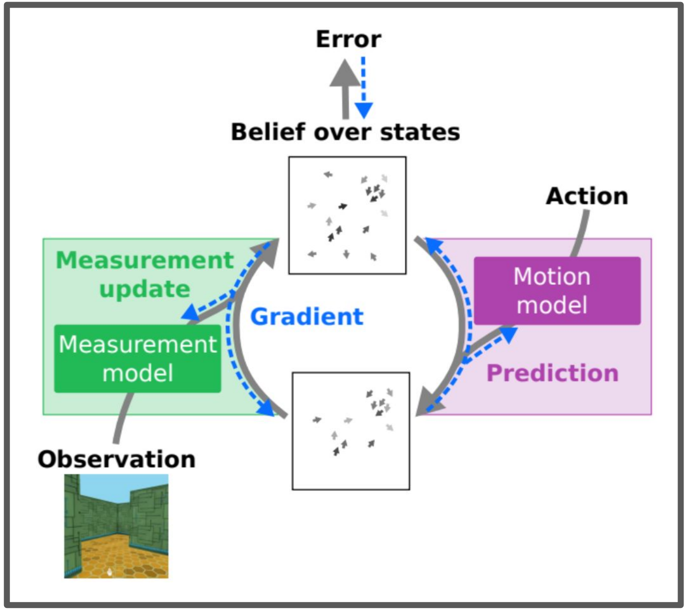
Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors

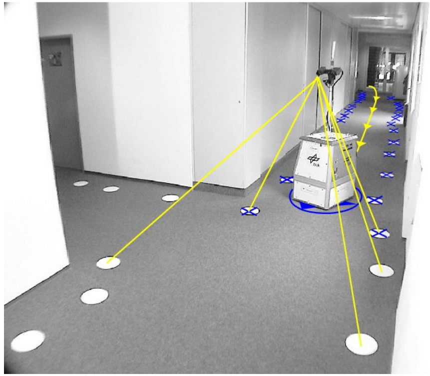
Presenter: Gerardo Riaño

9-28-2021

Differentiable Particle Filters

Robots take noisy **measurements** (from cameras, LiDAR, radar), while **exploring** their environment to estimate **system states** (e.g., position) and accomplish goals (e.g., planning, mapping)

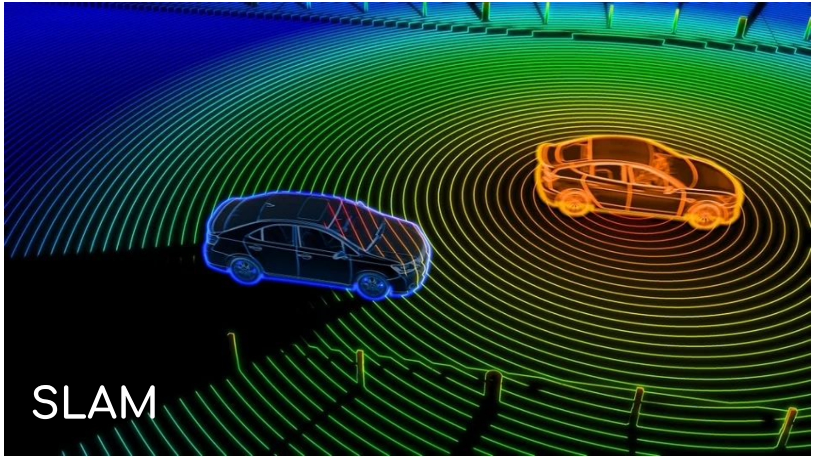


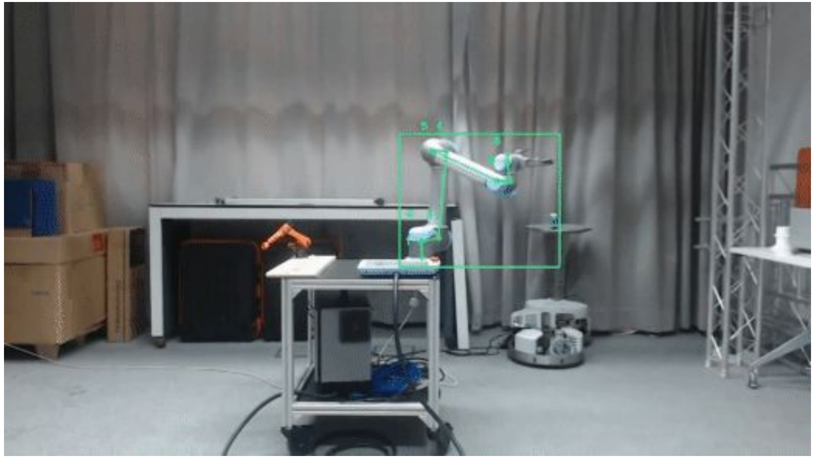


Jonschkowski, R., Rastogi, D., & Brock, O. (2018). Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv preprint arXiv:1805.11122*.

Motivation and Main Problem

State Estimation and Planning - robot localization, pose estimation, sensor fusion





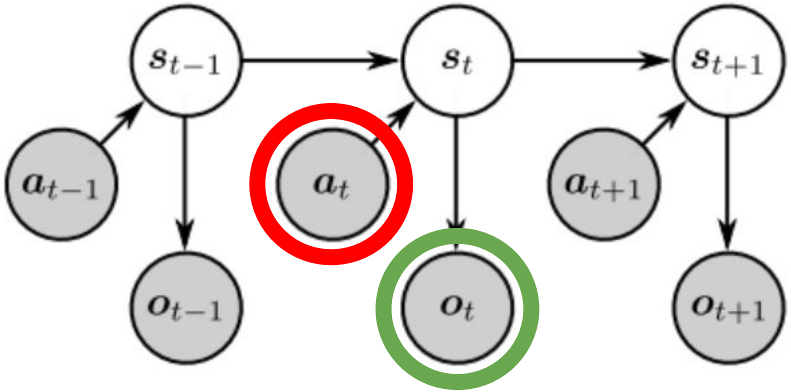
Tamar, A., Wu, Y., Thomas, G., Levine, S., & Abbeel, P. (2016). Value iteration networks. *arXiv preprint arXiv:1602.02867*.
Deng, X., Mousavian, A., Xiang, Y., Xia, F., Bretl, T., & Fox, D. (2021). PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking. *IEEE Transactions on Robotics*.

State Estimation

2016-2018	**Learning and Particle Filters (NN)**	**DPF embeds Bayesian filter priors in an end-to-end differentiable neural network** 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). Karl, M., Soelch, M., Bayer, J., & Van der Smagt, P. (2016). Deep variational bayes filters: Unsupervised learning of state space models from raw data. arXiv preprint arXiv:1605.06432.
1993 - 2018	Particle Filtering	Nonlinear dynamics, arbitrary noise distribution, intractable problem (High-computational cost)
1990s	Extended Unscented KF	Nonlinear dynamics, Gaussian noise
1960	Kalman Filter (KF)	Assumes linear dynamics

Problem Setting

From Bayesian Inference to Particle Filtering



Belief of being in s_t

$$\mathrm{bel}(s_t) = p(s_t | a_{1:t}, o_{1:t})$$

s: states
a: actions
o: observations

Prediction Step

$$\overline{\mathrm{bel}}(s_t) = \int \underbrace{p(s_t \mid s_{t-1}, a_t)}_{\text{Motion Model}} \mathrm{bel}(s_{t-1})\, ds_{t-1}$$

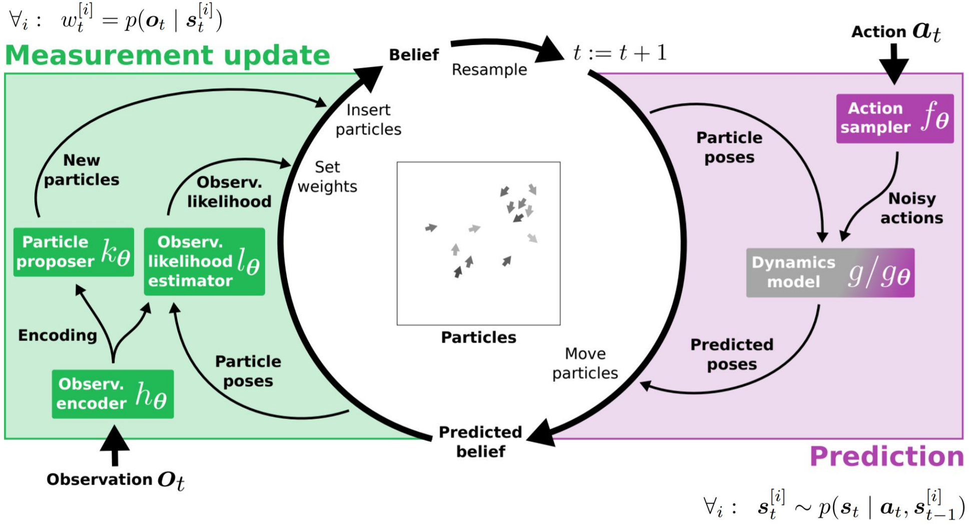
Measurement Update

Particles (Samples) $\longrightarrow$ $\mathrm{bel}(s_t) = \eta\, p(o_t \mid s_t)\, \overline{\mathrm{bel}}(s_t)$ Monte Carlo Simulation

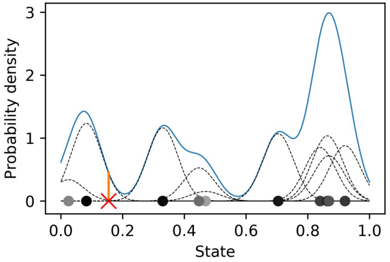
Particle Filtering

Models can be nonlinear and noise is not necessarily Gaussian

requires

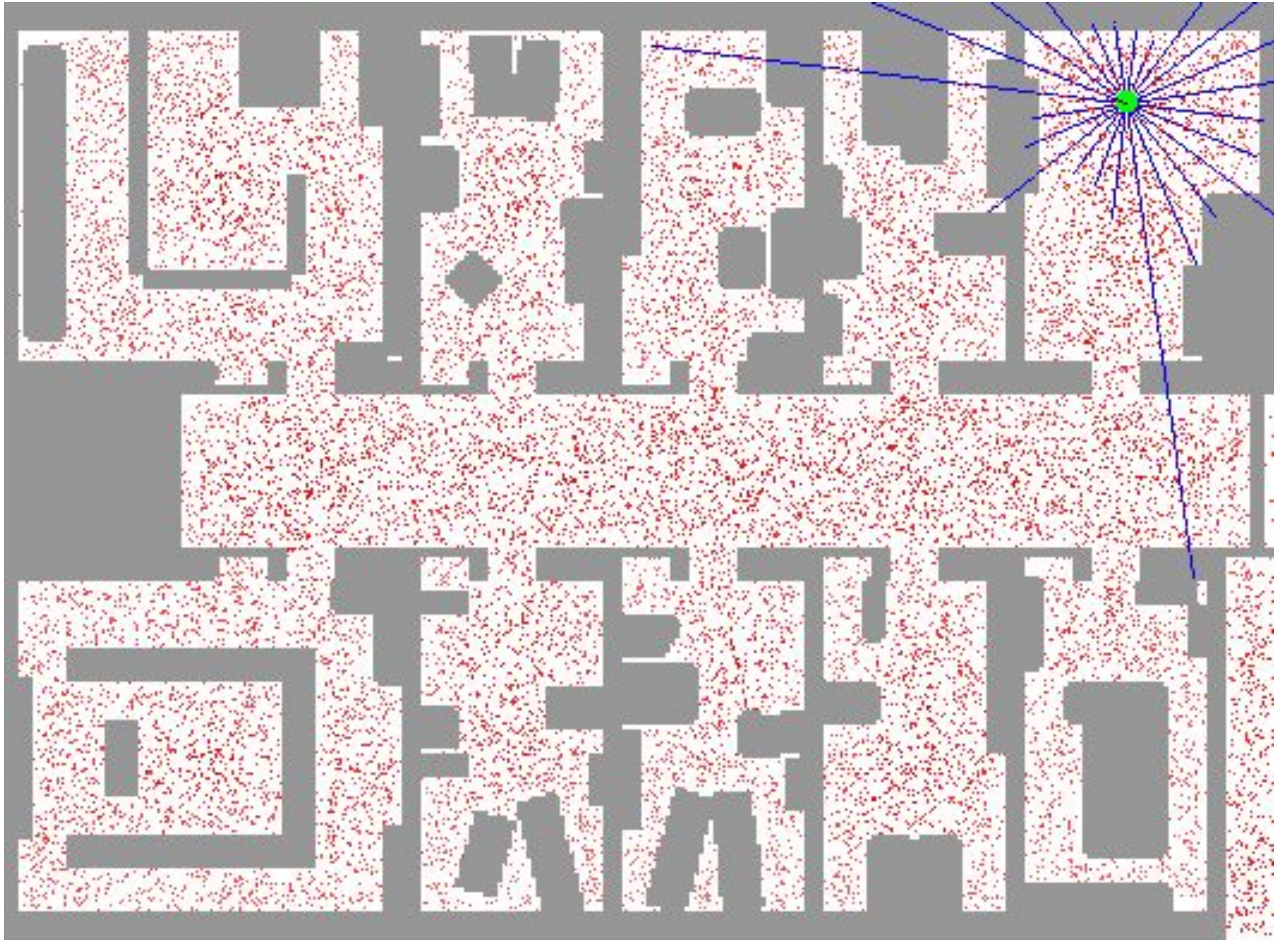


Importance Sampling & Resampling



$$\mathcal{S}_t = \boldsymbol{s}_t^{[1]}, \boldsymbol{s}_t^{[2]}, \ldots, \boldsymbol{s}_t^{[n]}$$

$$w_t^{[1]}, w_t^{[2]}, \ldots, w_t^{[n]}$$



1. Sample action
2. Prediction
3. Measurement Update
4. Resampling

Differentiable Particle Filtering (DPF)

1. Belief: modeled by n particles in d-dimensional state space

$$\mathrm{bel}(\boldsymbol{s}_t) = (S_t, \boldsymbol{w}_t)$$
$$S \in \mathbb{R}^{n \times d}$$

2. Prediction step
   - $f_{\boldsymbol{\theta}}$ action sampler
   - g dynamics model
   - $\boldsymbol{\epsilon}^{[i]}$ noise vector

$$\hat{\boldsymbol{a}}_t^{[i]} = \boldsymbol{a}_t + f_{\boldsymbol{\theta}}(\boldsymbol{a}_t, \boldsymbol{\epsilon}^{[i]} \sim \mathcal{N}),$$
$$\boldsymbol{s}_t^{[i]} = \boldsymbol{s}_{t-1}^{[i]} + g(\boldsymbol{s}_{t-1}^{[i]}, \hat{\boldsymbol{a}}_t^{[i]}),$$

3. Measurement Update
   - $h_{\boldsymbol{\theta}}$ observation encoder
   - $k_{\boldsymbol{\theta}}$ particle proposer
   - $l_{\boldsymbol{\theta}}$ likelihood estimator

$$\boldsymbol{e}_t = h_{\boldsymbol{\theta}}(\boldsymbol{o}_t),$$
$$\boldsymbol{s}_t^{[i]} = k_{\boldsymbol{\theta}}(\boldsymbol{e}_t, \boldsymbol{\delta}^{[i]} \sim B),$$
$$w_t^{[i]} = l_{\boldsymbol{\theta}}(\boldsymbol{e}_t, \boldsymbol{s}_t^{[i]}),$$

4. Particle Proposal & Resampling (Exponential samples around the current observation)
   Stochastic Universal Sampling

Supervised Learning

$f_{\boldsymbol{\theta}}$:	2 x fc(32, relu), fc(3) + mean centering across particles
$g_{\boldsymbol{\theta}}$:	3 x fc(128, relu), fc(3) + scaled by $\mathrm{E}_t[\mathrm{abs}(\boldsymbol{s}_t - \boldsymbol{s}_{t-1})]$
$h_{\boldsymbol{\theta}}$:	conv(3x3, 16, stride 2, relu), conv(3x3, 32, stride 2, relu), conv(3x3, 64, stride 2, relu), dropout(keep 0.3), fc(128, relu)
$k_{\boldsymbol{\theta}}$:	fc(128, relu), dropout*(keep 0.15), 3 x fc(128, relu), fc(4, tanh)
$l_{\boldsymbol{\theta}}$:	2 x fc(128, relu), fc(1, sigmoid scaled to range [0.004, 1.0])

fc: fully connected, conv: convolution, *: applied at training and test time

$\boldsymbol{\theta}$: model parameters

1. Individual learning of the motion model $\quad \boldsymbol{\theta}_f^* = \operatorname{argmin}_{\boldsymbol{\theta}_f} - \log p(\boldsymbol{s}_t^* \mid \boldsymbol{s}_{t-1}^*, \boldsymbol{a}_t; \boldsymbol{\theta}_f)$

2. Individual learning of the measurement model

$$\boldsymbol{\theta}_{h,l}^* = \operatorname{argmin}_{\boldsymbol{\theta}_{h,l}} - \log(\mathrm{E}_t[l_{\boldsymbol{\theta}}(h_{\boldsymbol{\theta}}(\boldsymbol{o}_t), \boldsymbol{s}_t^*)]) - \log(1 - \mathrm{E}_{t_1,t_2}[l_{\boldsymbol{\theta}}(h_{\boldsymbol{\theta}}(\boldsymbol{o}_{t_1}), \boldsymbol{s}_{t_2}^*)])$$

3. End-to-end learning $\quad \boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} - \log \mathrm{E}_t[\mathrm{bel}(\boldsymbol{s}_t^*; \boldsymbol{\theta})]$

Limitations and Future Work

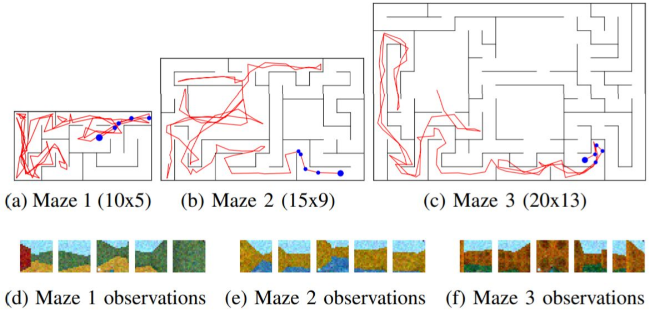
- End-to-end gradient by backpropagation from the DPF output through the filtering loop only
- The gradient neglects the effects of previous prediction and update steps on the current belief (resampling is not differentiable)

The authors propose alternative differentiable resampling methodologies:

- **Partial resampling:**
  keep n-m particles from the previous time step, operate backpropagation with those
- **Proxy gradients:**
  proxy gradient for the weight of a resampled particle that is tied to the particle it was sampled from

Experiments

- **Baseline:** LSTMs, and Backprop Kalman Filters (BKF)
- **Description:** the authors evaluated a) the effect of end-to-end learning **(e2e)** compared to individual learning **(ind)** and b) the influence of algorithmic priors encoded in DPFs (comparison with generic LSTM, policy generalization)
- **Metrics:** error rates



GLOBAL LOCALIZATION

DeepMind Lab

RGB-D (with noise)

Hand-coded policies (10% random)

1000 trajectories

VISUAL ODOMETRY

KITTI dataset

Ground Truth provided

RGB camera

1st order dynamics

Global Localization

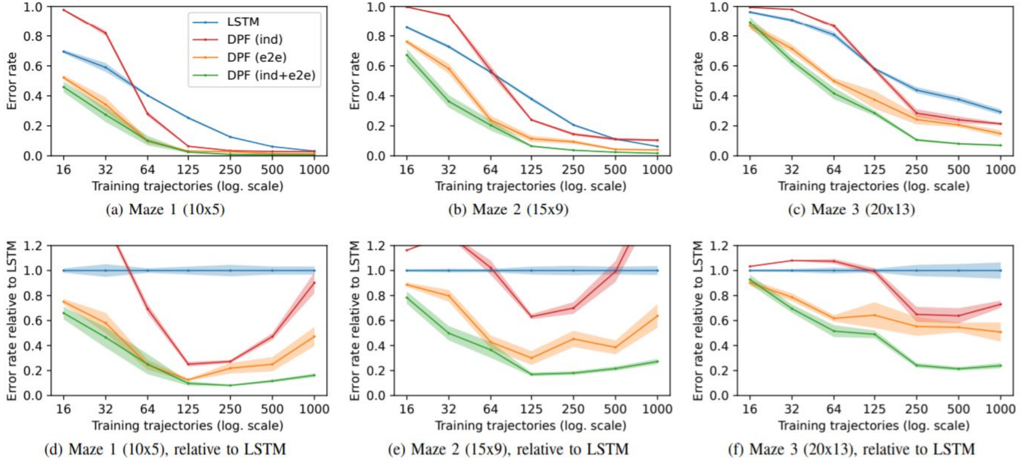


Fig. 8: **Learning curves** in all mazes (a-c), also relative to LSTM baseline (d-f). ind: individual learning, e2e: end-to-end learning. Shaded areas denote standard errors.

- End-to-end learned DPFs (orange line) consistently outperform individually trained DPFs (red line) across all mazes
- Performance improves even more when we sequence individual and end-to-end learning (green line in Fig. 8a-c)
- The error rate of DPF (ind+e2e) is lower than for LSTM for all mazes and all amounts of training data
- Knowing the dynamics model is helpful but not essential for DPF's performance

Global Localization

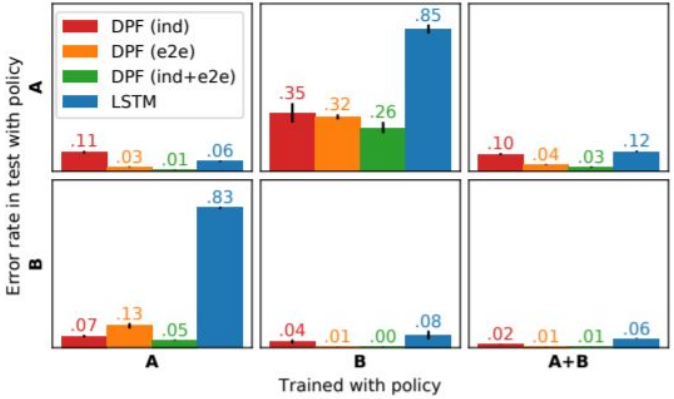


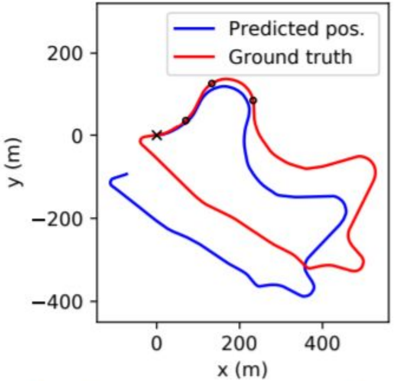
Fig. 9: **Generalization between policies** in maze 2. A: heuristic exploration policy, B: shortest path policy. Methods were trained using 1000 trajectories from A, B, or an equal mix of A and B, and then tested with policy A or B.

- DPFs relative performance to LSTMs improves with more data and converges to about 1/10 to 1/3
- The priors from the Bayes filter algorithm reduce variance without adding bias
- All methods have low error rates when tested on their training policy
- The LSTM baseline is not able to generalize to new policies

Visual Odometry

Infer location based on camera observations

(a) Visual input (image and difference image) at time steps 100, 200, and 300 (indicated in (b) by black circles)



(b) Trajectory 9; starts at (0,0)

	Test 100	Test 100/200/400/800
Translational error (m/m)		
BKF*	0.2062	0.1804
DPF (ind)	0.1901 ± 0.0229	0.2246 ± 0.0371
DPF (e2e)	**0.1467 ± 0.0149**	0.1748 ± 0.0468
DPF (ind+e2e)	0.1559 ± 0.0280	**0.1666 ± 0.0379**
Rotational error (deg/m)		
BKF*	0.0801	0.0556
DPF (ind)	0.1074 ± 0.0199	0.0806 ± 0.0153
DPF (e2e)	0.0645 ± 0.0086	0.0524 ± 0.0068
DPF (ind+e2e)	**0.0499 ± 0.0082**	**0.0409 ± 0.0060**

Means ± standard errors; * results from [7]

DPFs outperform BKFs, in particular for short sequences where they reduce the error by ~30%
Authors claim that improvement over BKFs is due to particles being able to model long-tailed distributions

Critique / Limitations / Open Issues

- Even though authors claim that knowing the dynamics of the model is not essential for performance, other works in robotics and safe learning show the opposite

- DPFs might outperform the proposed baselines but results show the methodology might be impractical and requires further research

- It would be interesting to see this approach operating with sensor fusion to further reduce error rates as it is the strength of particle filtering

Further Reading

- Tamar, A., Wu, Y., Thomas, G., Levine, S., & Abbeel, P. (2016). Value iteration networks. *arXiv preprint arXiv:1602.02867*.
- Deng, X., Mousavian, A., Xiang, Y., Xia, F., Bretl, T., & Fox, D. (2021). PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking. IEEE Transactions on Robotics.
- Ma, X., Karkus, P., Hsu, D., & Lee, W. S. (2020, April). Particle filter recurrent neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 04, pp. 5101-5108)
- ICRA18 Keynote: Machine Learning for Safe High-performance Control of Mobile Robots (https://www.youtube.com/watch?v=-Lp_ckvzhrk)

Summary

- Differentiable particle filters are introduced to demonstrate the advantages of combining end-to-end learning with algorithmic priors
- End-to-end learning optimizes models for performance while algorithmic priors enable explainability and regularize learning, which improves data-efficiency and generalization
- The use of algorithms as algorithmic priors will help to realize the potential of deep learning in robotics
- DPFs outperform baseline network architectures such as LSTMs and BKFs in terms of error reduction