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

Presenter: Haroon Mushtaq

September 14, 2023

Overview

Motivation	Problem	Related Work
Method	Experiments/Results	Discussion

Motivation

End-to-End learning enables individual parts of a system to mutually adapt for optimal end-to-end performance.[1]

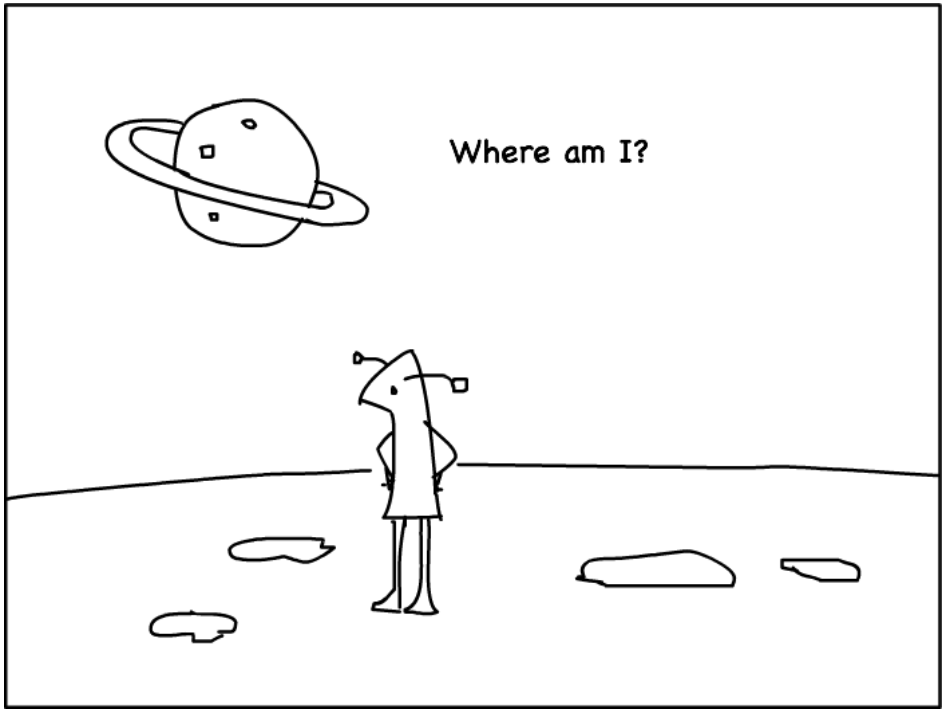
Robotics problems have problem structure represented in algorithms that can be exploited.

Turn robotics algorithms into network architectures to enable optimal end-to-end performance. *Algorithmic prior.*

1 Only if given suitable priors

Main Problem - State Estimation/Localization

- Robots use sensors to collect information about the environment.
- Sensors are noisy.
- Noisy data produces noisy motion.
- Where am I?
- How can I know where I am from uncertain observations and actions?



Related Work

- ➔ Particle Filter Networks with Application to Visual Localization. Karkus,P., Hsu, D., & Lee, W.S. (2018)
  - ◆ Similar to current study, but important differences:
    - differentiable approximation of resampling
    - apply PF-net to visual localization in new and unseen environments after learning (current study deals with fixed environment)

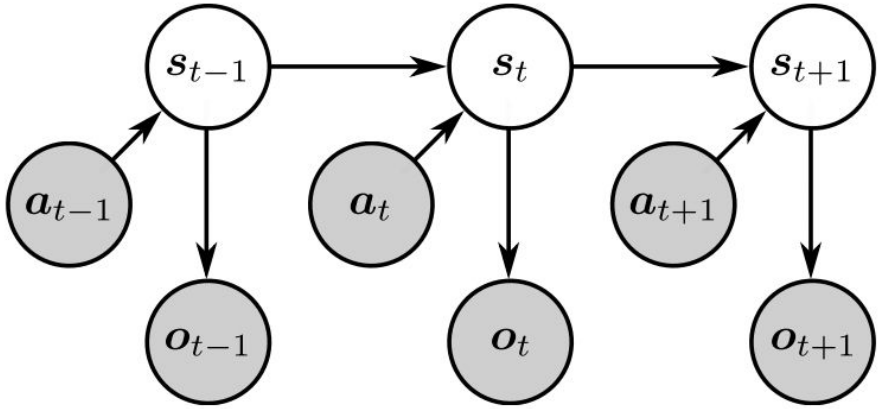
- ➔ Value iteration networks. Tamar, A., Wu, Y., Thomas, G., Levine, S., & Abbeel, P. (2016)
  - ◆ Value iteration in a grid based state space can be represented by CNNs.

- ➔ End-To-End Learnable Histogram Filters. Jonschkowski, R. & Brock, O. (2016)
  - ◆ End-to-end learnable histogram filter

- ➔ Backprop KF: Learning Discriminative Deterministic State Estimators. Haarnoja, T. ,Ajay, A.,Levine, S., & Abbeel,P. (2016)
  - ◆ Differentiable Kalman filter with a Gaussian belief and an end-to-end learnable measurement model from visual input.

Background - Bayes Filters

- Problem:
  - Estimate a latent *state* s_t from *observation* o_t and *actions* a_t.
- How to deal with uncertainty?
  - Estimate a probability distribution over current state conditioned on history of observations and actions
  - This estimate is called the *Belief*.
    - $bel(s_t) = p(s_t|a_{1:t},o_{1:t})$



Background - Bayes Filters

Two steps in Bayes Filters to update belief:

1. Prediction:
   Based on a motion model $p(s_t|s_{t-1},a_t)$

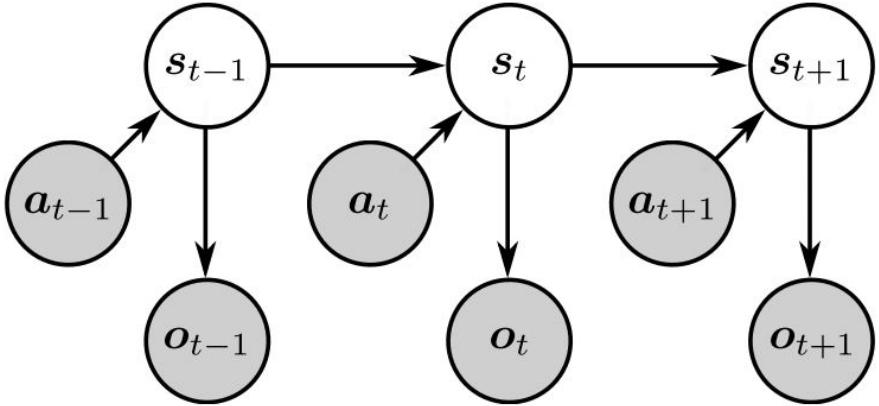
   Compute *predicted belief*

   $$\overline{bel(s_t)} = \int p(s_t|s_{t-1},a_t)bel(s_t)ds_{t-1}$$

2. Measurement:
   Based on a measurement model $p(o_t|s_t)$

   Update the belief using Bayes' rule

   $$bel(s_t) = \eta p(o_t|s_t)\overline{bel(s_t)}$$



Background - Particle Filters

- Approximate belief with a set of particles with weights

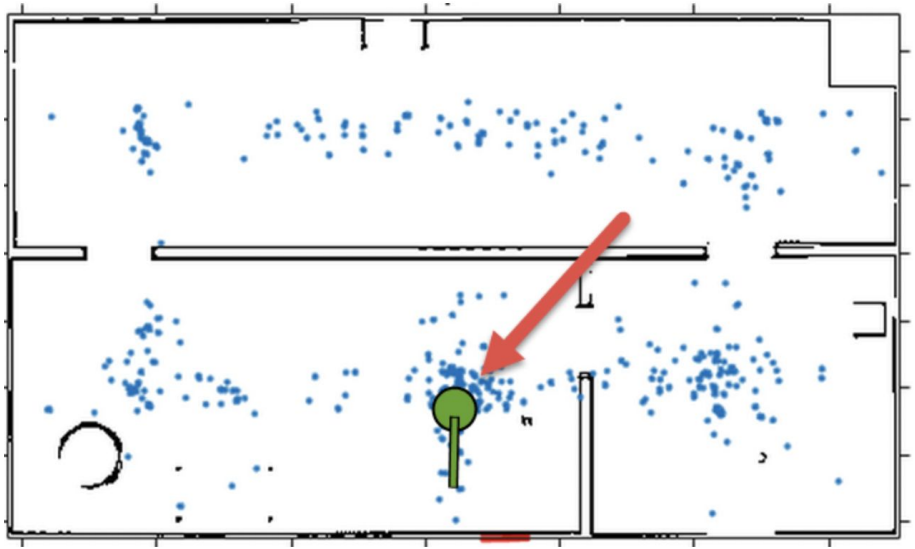
  $S_t = \{s_t^{[1]}, s_t^{[2]}, s_t^{[3]}, \square, s_t^{[n]}\}$; $w_t^{[1]}, w_t^{[2]}, \square, w_t^{[n]}$

- Distribution is updated by:
  - moving particles
  - changing weights
  - resampling*

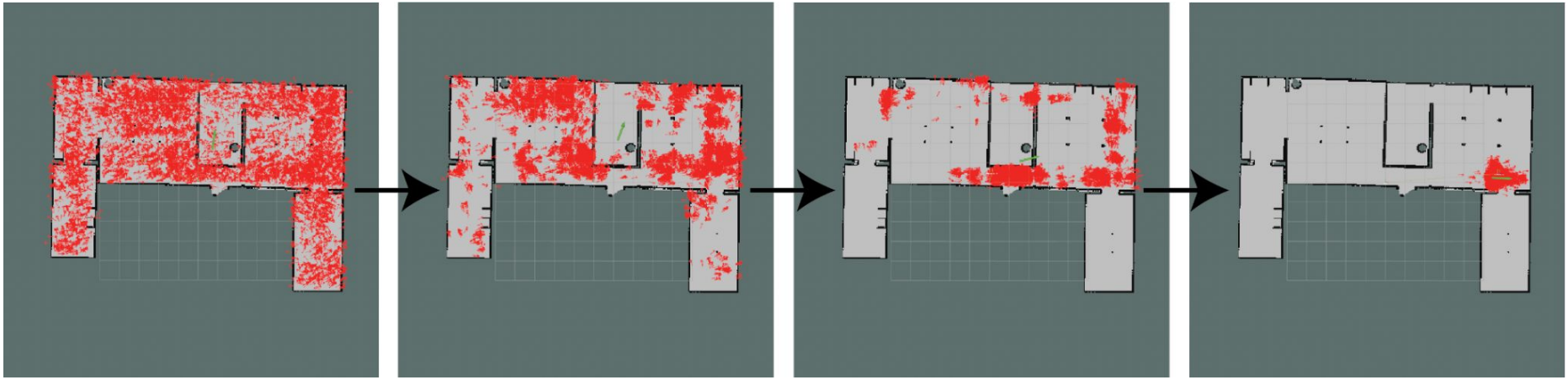
- Prediction step implementation:
  - move each particle stochastically by sampling from the following motion model:

    $\forall i: s_t^{[i]} \sim p(s_t | a_t, s^{[i]}_{t-1})$

- Measurement step implementation:
  - set weights of each particle to observation likelihood: $\forall i: w_t^{[i]} \sim p(o_t | s^{[i]}_t)$
- Resample

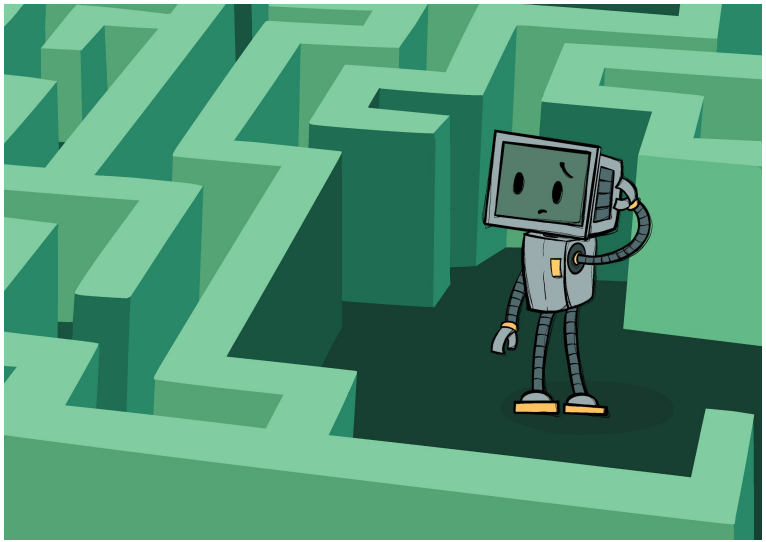


Background - Particle Filters



Differentiable Particle Filter

- What is it?
    - Differentiable implementation of Particle Filter algorithm w/ end-to-end learnable models.
- Need the ability to backpropagate gradient through Particle Filter algorithm to change models and ultimately, improve performance.
- But…how is this implemented?



Differentiable Particle Filter

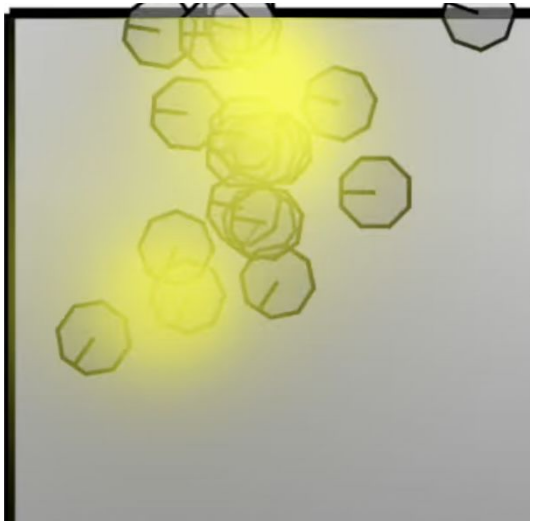
- Belief:
  - Representing belief at time *t* with set of weighted particles:

    $$bel(s_t) = (S_t, w_t) \text{ where } S \in \mathbb{R}^{nxd} \text{ \& } w \in \mathbb{R}^n$$

  - Update previous belief $bel(s_{t-1})$ at every time step to get $bel(s_t)$
- Prediction:
  - Move each particle by sampling from $\forall i: s_t^{[i]} \sim p(s_t|a_t, s^{[i]}_{t-1})$
  - Motion model split into an action sampler *f* and dynamics model *g* :
    - *f* creates a noisy action and *g* moves each particle accordingly

    $$\hat{a}_t^{[i]} = a_t + f_\theta(a_t, \epsilon^{[i]} \sim N) \ ;$$

    $$s^{[i]}_t = s^{[i]}_{t-1} + g(s^{[i]}_{t-1}, \hat{a}_t^{[i]})$$



Differentiable Particle Filter

- Measurement Update:
  - Use observation to compute particle weight
  - Three components:
    i. Observation encoder $h_\theta(o_t) = e_t$
    ii. Particle Proposer $k_\theta(e_t, \boldsymbol{\delta}^{[i]} \sim B) = s_t^{[i]}$
    iii. Observation Likelihood Estimator $l_\theta(s_t^{[i]}, e_t) = w_t^{[i]}$
  - h_θ, k_θ, l_θ are feedforward networks and $\boldsymbol{\delta}^{[i]}$ is a dropout vector.
- Particle Proposal and Resampling:
  - Proposing particles from current observation
  - Resample to eliminate particles (stochastic universal sampling)
  - RESAMPLING NOT DIFFERENTIABLE!!!

TABLE I: **Feedforward networks for learnable DPF models**

$f_{\boldsymbol{\theta}}$:	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])

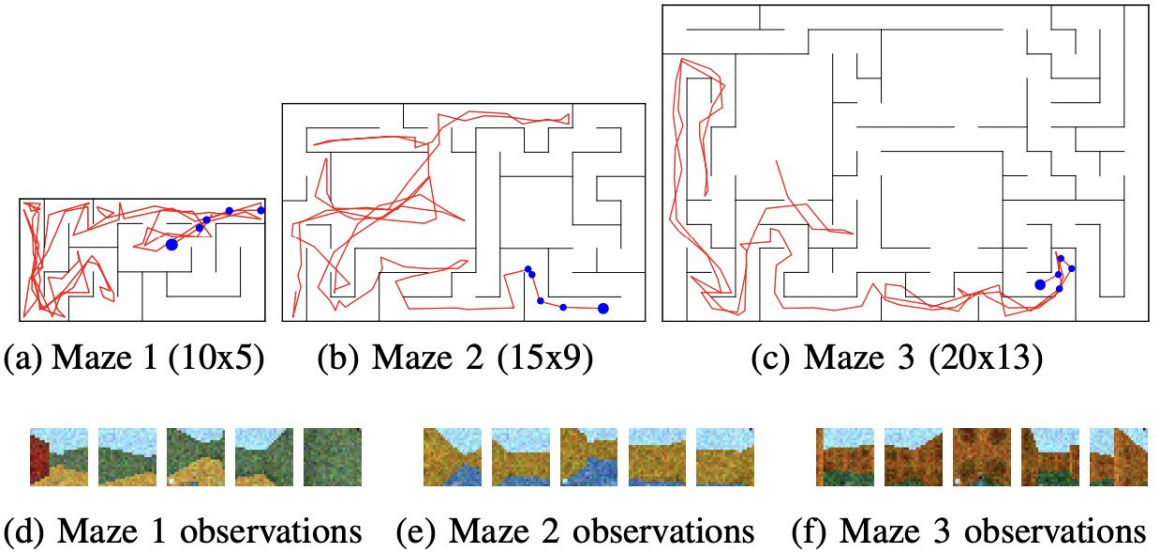
fc: fully connected, conv: convolution, *: applied at training and test time

INTERMISSION

Experiment Setup & Results

Global Localization

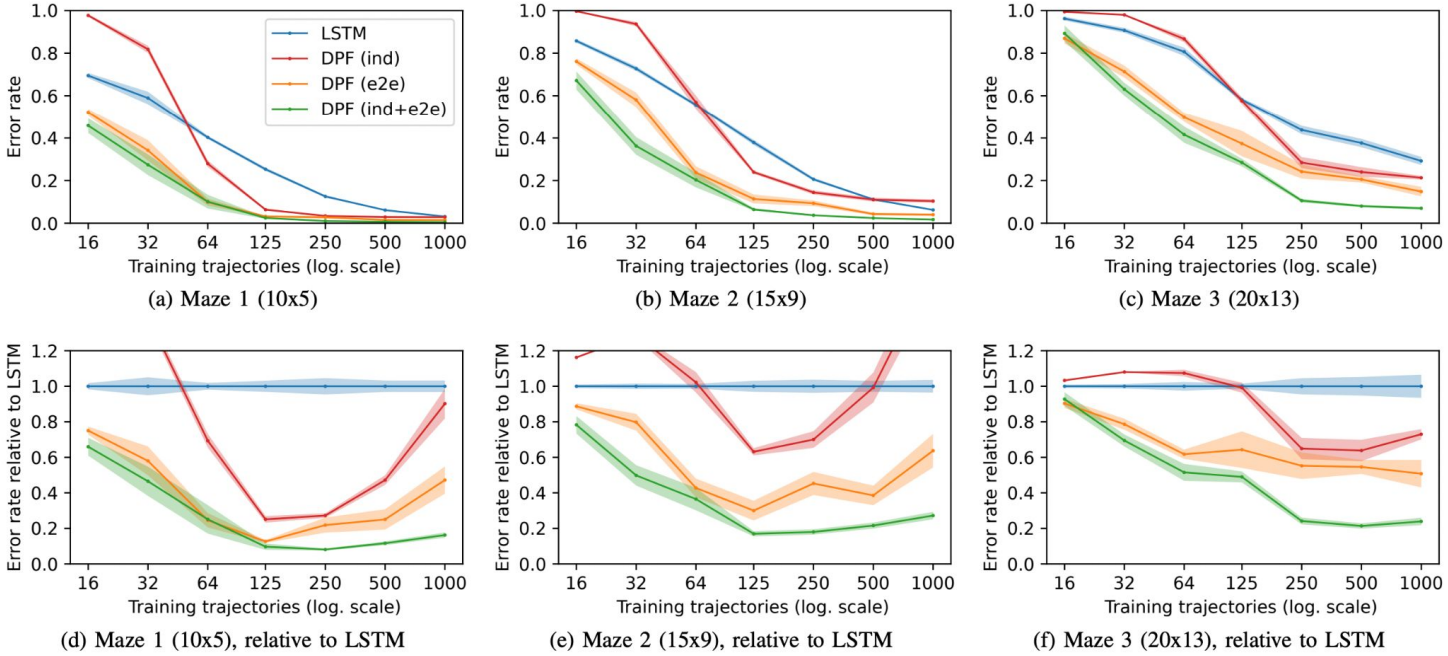
- Estimate the pose of a robot based on visual and odometry data.
- Navigation environment from DeepMind Lab.
- Robot followed hand-coded policy.
- Collected 1000 trajectories for each maze.



(a) Maze 1 (10x5) (b) Maze 2 (15x9) (c) Maze 3 (20x13)

(d) Maze 1 observations (e) Maze 2 observations (f) Maze 3 observations

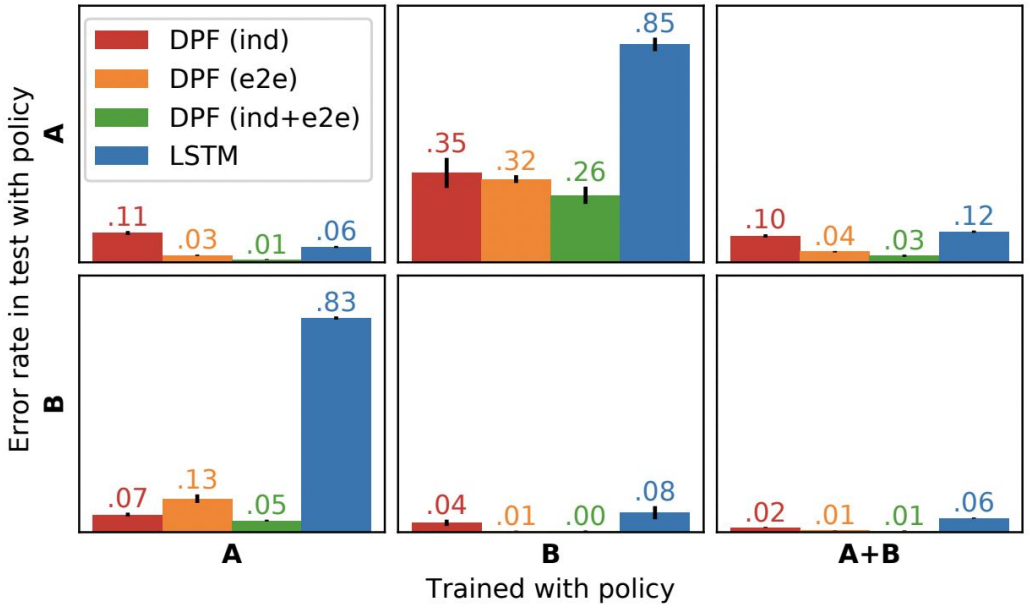
Global Localization



End-to-end DPFs outperform
across all environments

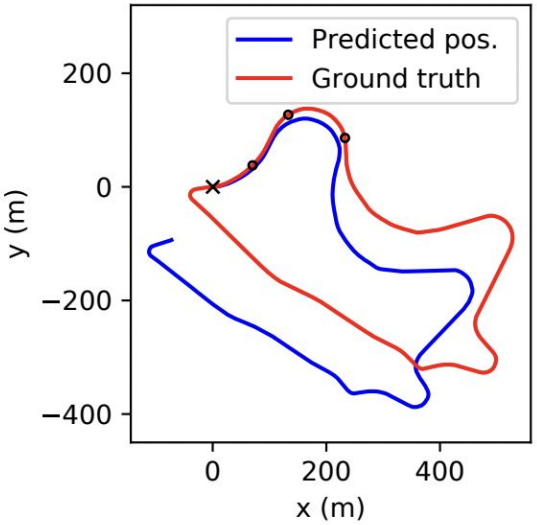
Global Localization

- Algorithmic priors lead to policy invariance.
  - Desirable b/c useful in variety of tasks
- LSTM baseline not able to generalize
- LSTM error rate vs DPF error rate



Visual Odometry

(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)

- KITTI visual odometry data set.
- Track the car's position and orientation by estimating its translational and angular velocity.
- DPFs outperform BKFs:
  - Reason? Particles represent long-tail probability distributions.
- DPFs generalize to different tasks well.

Visual Odometry

TABLE II: **KITTI visual odometry results**

	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]

Limitations

- ➔ **Since resampling is not differentiable, the gradient neglects the effects of previous prediction and update steps on the current belief.**
- ➔ **Does not seem very practical/robust as it stands.**
- ➔ **Future work should address differentiable resampling.**

Extended Readings

- Particle Filter Networks with Application to Visual Localization. Karkus,P., Hsu, D., & Lee, W.S. (2018)
- Value iteration networks. Tamar, A., Wu, Y., Thomas, G., Levine, S., & Abbeel, P. (2016)
- End-To-End Learnable Histogram Filters. Jonschkowski, R. & Brock, O. (2016)
- Backprop KF: Learning Discriminative Deterministic State Estimators. Haarnoja, T. ,Ajay, A.,Levine, S., & Abbeel,P. (2016)
- Probabilistic Robotics.Thrun S., Burgard, W., & Fox, D. (2005)

Summary

- End-to-End learning enables individual parts of a system to mutually adapt for optimal end-to-end performance.
- DPFs remain explainable through algorithmic prior.
- Algorithmic prior regularizes learning and improves generalization.
- DPFs reduce error rate by ~80% and require ~87% less training data for same error rate.
- Resampling is not differentiable; limits scope.
- Algorithmic priors improve performance and lead to policy invariance