Particle Filter Networks with Application to Visual Localization

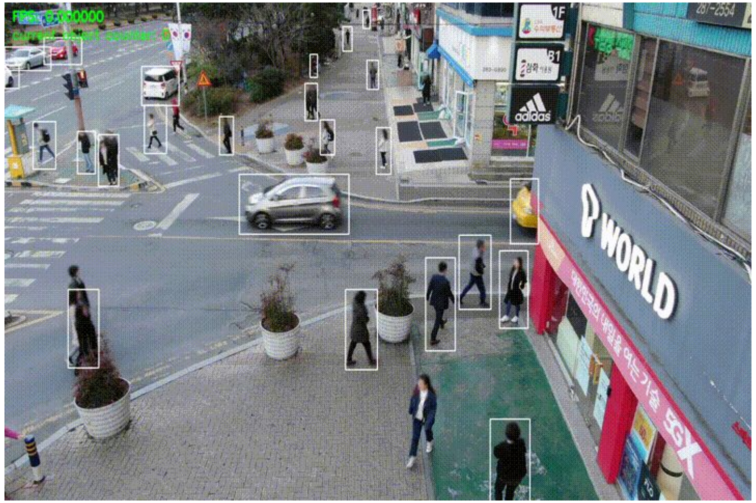
Peter Karkus, David Hsu, Wee Sun Lee

Presenter: Amanda Adkins

09/28/2021

State Estimation

Object Tracking



Robot Localization

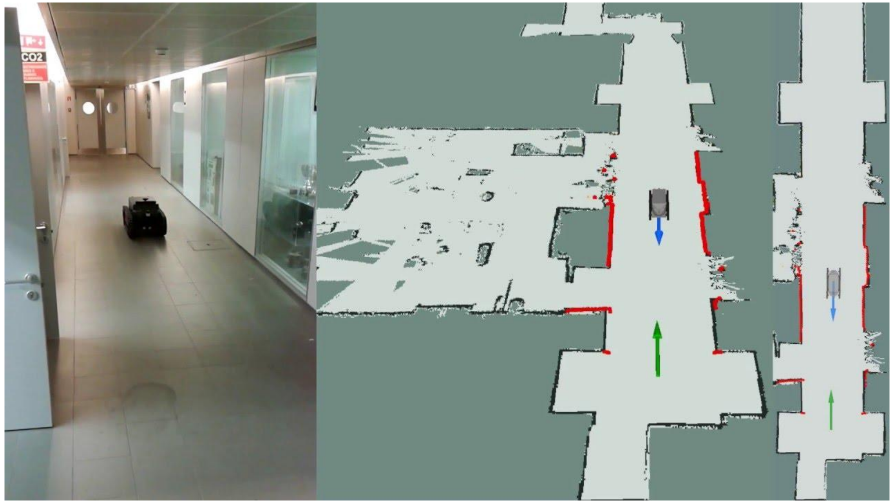


Image Sources:
https://github.com/yehengchen/Object-Detection-and-Tracking
https://github.com/carlosmccosta/dynamic_robot_localization

State Estimation: Traditional Approaches

- Bayes Filter
  - Broad class of approaches for sequential state estimation
  - Belief at time t computed recursively from belief at t-1
  - Information from control data and sensor observations
- Kalman Filter
  - Bayes filter with Gaussian belief
- Particle Filter
  - Bayes filter represented by samples
  - Need observation model $Z(o_t|s_t^k;\mathbb{M})$ and transition model $T(s_t|u_t, s_{t-1}^k)$
    - Observation model hard to craft for complex systems

State Estimation: Machine Learning

- Often, individual components of a system use an learned component
  - Learned Keypoints: SuperPoint
  - Feature Matching Noise: IV-SLAM
  - Learned representations: PoseRBPF
- Growing interest in training systems end-to-end
  - Atlas: End-to-End 3D Scene Reconstruction from Posed Images
  - DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks

Tang, Jiexiong, et al. "Self-supervised 3d keypoint learning for ego-motion estimation." *arXiv preprint arXiv:1912.03426*(2019).
Rabiee, Sadegh, and Joydeep Biswas. "IV-SLAM: Introspective vision for simultaneous localization and mapping." *CoRL* (2020).
Deng, Xinke, et al. "PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking." *IEEE Transactions on Robotics* (2021).
Wang, Sen, et al. "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks." *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.
Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End- to-End 3D Scene Reconstruction from Posed Images. In European Conference on Computer Vision (ECCV), 2020.

Particle Filter Networks with Application to Visual Localization

- Neural network architecture with:
    - Embedded particle filter structure
    - Observation and transition models
- Differentiable approximation for sampling
- Enables end-to-end learning

Related Work

- Histogram filter network
  - Discrete state space
  - Doesn’t scale with large environments or high-dimensional state spaces
- Kalman filter network
  - Limited application to problems where belief can be approximated by a Gaussian

R. Jonschkowski and O. Brock. End-to-end learnable histogram filters. In Workshop on Deep Learning for Action and Interaction at NIPS, 2016.
T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In Advances in Neural Information Processing Systems, pages 4376–4384, 2016.

Related Work

- Conventional particle filters
  - Observation model hard to construct for sensors more complicated than 2D LiDAR
- Differentiable particle filters
  - Don’t have differentiable resampling step
- Learned generative models for particle filter proposal
  - Observation space is complex for many robotics applications → learning a generative model is hard

R. Jonschkowski, D. Rastogi, and O. Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. arXiv preprint arXiv:1805.11122, 2018.
C. A. Naesseth, S. W. Linderman, R. Ranganath, and D. M. Blei. Variational sequential monte carlo. arXiv preprint arXiv:1705.11140, 2017.
C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. Filtering variational objectives. In Advances in Neural Information Processing Systems, pages 6576–6586, 2017.
T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-encoding sequential monte carlo. In Proceedings of the 6th International Conference on Learning Representations (ICLR), 2018.
S. Gu, Z. Ghahramani, and R. E. Turner. Neural adaptive sequential monte carlo. In Advances in Neural Information Processing Systems, pages 2629–2637, 2015.

Problem Setting

- Sequential state estimation should provide
  - Belief $b(s)$
  - Maximum likelihood state estimate $\bar{s}_t$
- Particle filter approach
  - $b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$ - Belief represented by set of particles (weighted samples)
    - s_k - particle state
    - w_k - particle weight
    - Transition model: $T(s_t | u_t, s_{t-1}^k)$
    - Observation model: $Z(o_t | s_t^k; \mathbb{M})$
  - $\bar{s}_t = \sum_k w_t^k s_t^k$ Maximum likelihood state estimate given by weighted mean

Problem Setting: Visual Localization

- States s_k are robot poses
- Observation o_t is a camera or lidar observation
- u_t is odometry
- $\mathbb{M}$ is 2D floor map

Approach: PF-net

Particle Filter

1.	Belief represented by set of particles	$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2.	Particle states updated by sampling from transition model	$s_t^k \sim T(s_t \mid u_t, s_{t-1}^k)$
3.	Compute likelihood from observation model	$f_t^k = Z(o_t \mid s_t^k; \mathbb{M})$
4.	Particle weights updated with likelihood	$w_t^k = \eta f_t^k w_{t-1}^k$
5.	Particles resampled with probability proportional to weight	$p(k) = w_t^k$
6.	Weights updated according to uniform distribution	$w_t'^k = 1/K$

PF-net

- Idea: Embed particle filter steps in neural network
  - Allows end to end training of observation and transition models in context of estimation
  - *Requires all steps to be differentiable*

PF-net

1. Belief represented by set of particles $b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2. Particle states updated by sampling from transition model $s_t^k \sim T(s_t | u_t, s_{t-1}^k)$
3. Compute likelihood from observation model $f_t^k = Z(o_t | s_t^k; \mathbb{M})$
4. Particle weights updated with likelihood $w_t^k = \eta f_t^k w_{t-1}^k$
5. Particles resampled with probability proportional to weight $p(k) = w_t^k$
6. Weights updated according to uniform distribution $w_t'^k = 1/K$

PF-net

1.	Belief represented by set of particles	$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2.	Particle states updated by sampling from transition model	Neural network
3.	Compute likelihood from observation model	Neural network
4.	Particle weights updated with likelihood	$w_t^k = \eta f_t^k w_{t-1}^k$
5.	Particles resampled with probability proportional to weight	$p(k) = w_t^k$
6.	Weights updated according to uniform distribution	$w_t'^k = 1/K$

Observation model and transition model represented by neural networks

Still need differentiable way to sample from transition model

PF-net

1.	Belief represented by set of particles	$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2.	Particle states updated by sampling from transition model	Neural network
3.	Compute likelihood from observation model	Neural network
4.	Particle weights updated with likelihood	$w_t^k = \eta f_t^k w_{t-1}^k$
5.	Particles resampled with probability proportional to weight	$p(k) = w_t^k$
6.	Weights updated according to uniform distribution	$w_t'^k = 1/K$

Take noise vector as input and express transition model as function of noise vector

PF-net

1.	Belief represented by set of particles	$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2.	Particle states updated by sampling from transition model	Neural network
3.	Compute likelihood from observation model	Neural network
4.	Particle weights updated with likelihood	$w_t^k = \eta f_t^k w_{t-1}^k$
5.	Particles resampled with probability proportional to weight	$p(k) = w_t^k$
6.	Weights updated according to uniform distribution	$w_t'^k = 1/K$

Uniform particle weights produce zero gradients → non-differentiable

Replace with soft-resampling

PF-net

1.	Belief represented by set of particles	$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2.	Particle states updated by sampling from transition model	Neural network
3.	Compute likelihood from observation model	Neural network
4.	Particle weights updated with likelihood	$w_t^k = \eta f_t^k w_{t-1}^k$
5.	Particles resampled with probability proportional to weight	$q(k) = \alpha w_t^k + (1 - \alpha)1/K$
6.	Weights updated according to uniform distribution	$w_t'^k = 1/K$

Sample from a combination of uniform distribution and $p(k)$

PF-net

1. Belief represented by set of particles	$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2. Particle states updated by sampling from transition model	Neural network
3. Compute likelihood from observation model	Neural network
4. Particle weights updated with likelihood	$w_t^k = \eta f_t^k w_{t-1}^k$
5. Particles resampled with probability proportional to weight	$q(k) = \alpha w_t^k + (1-\alpha)1/K$
6. Weights updated according to uniform distribution	$w_t'^k = \frac{p(k)}{q(k)} = \frac{w_t^k}{\alpha w_t^k + (1-\alpha)1/K}$

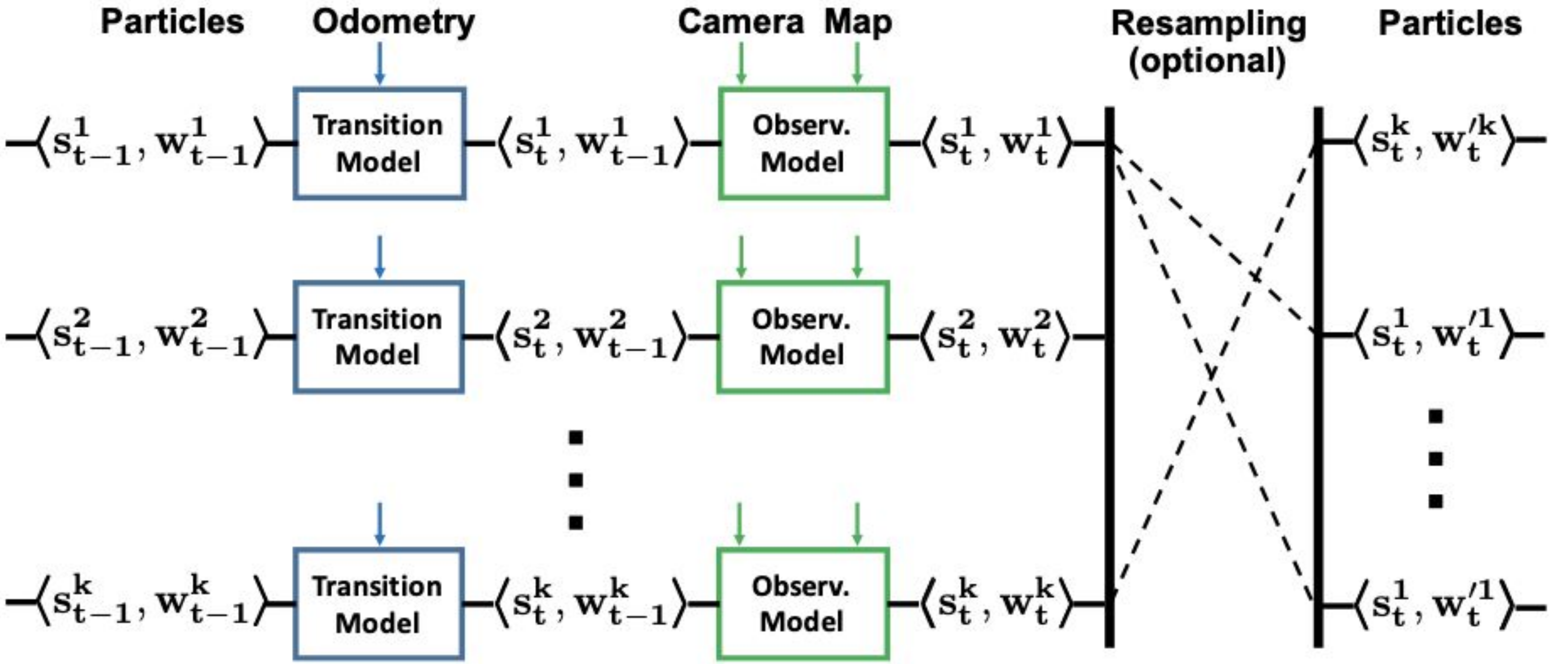
New weights computed by based on resampling likelihood and particle weight

PF-net

1.	Belief represented by set of particles	$b_t(s) \approx \langle s_t^k, w_t^k \rangle_{k=1:K}$
2.	Particle states updated by sampling from transition model	Neural network
3.	Compute likelihood from observation model	Neural network
4.	Particle weights updated with likelihood	$w_t^k = \eta f_t^k w_{t-1}^k$
~~5.~~	~~Particles resampled with probability proportional to weight~~	~~$q(k) = \alpha w_t^k + (1-\alpha)1/K$~~
~~6.~~	~~Weights updated according to uniform distribution~~	~~$w_t'^k = \frac{p(k)}{q(k)} = \frac{w_t^k}{\alpha w_t^k + (1-\alpha)1/K}$~~

Resampling not helpful in low uncertainty settings

PF-net


Particles
Odometry
Camera
Map
Resampling (optional)
Particles
$\langle s_{t-1}^1, w_{t-1}^1 \rangle$
Transition Model
$\langle s_t^1, w_{t-1}^1 \rangle$
Observ. Model
$\langle s_t^1, w_t^1 \rangle$
$\langle s_{t-1}^2, w_{t-1}^2 \rangle$
Transition Model
$\langle s_t^2, w_{t-1}^2 \rangle$
Observ. Model
$\langle s_t^2, w_t^2 \rangle$
$\langle s_{t-1}^k, w_{t-1}^k \rangle$
Transition Model
$\langle s_t^k, w_{t-1}^k \rangle$
Observ. Model
$\langle s_t^k, w_t^k \rangle$
$\langle s_t^k, w_t'^k \rangle$
$\langle s_t^1, w_t'^1 \rangle$
$\langle s_t^1, w_t'^1 \rangle$

PF-net Specialization

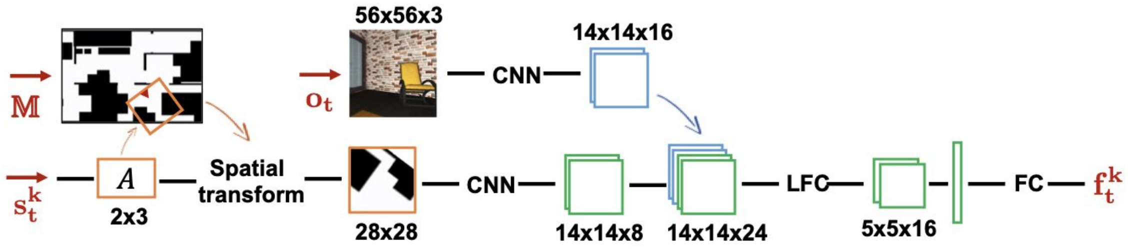
- State representation
- Loss function
- Network architecture for transition model and observation model

PF-net: Visual Localization

- State representation $s_t = x_t, y_t, \phi_t$
- Loss function: mean squared error $\mathcal{L} = \sum_t (\overline{x}_t - x_t^*)^2 + (\overline{y}_t - y_t^*)^2 + \beta(\overline{\phi}_t - \phi_t^*)^2$
  - $\overline{x}_t, \overline{y}_t, \overline{\phi}_t$ Estimated robot pose
  - x_t^*, y_t^*, ϕ_t^* True robot pose

PF-net: Visual Localization

- Network architecture for transition model and observation model
  - Transition model: Combines previous particle, odometry, and Gaussian noise
  - Observation model:
    - Takes map, particle state, and observations as input, produces particle likelihood
    - Obtain local map through affine transformation network
    - Extract features from map and observation using CNNs and combine using fully connected layers



Experiments

Experimental Setup

- House3D Simulator
  - Simulates multi-room residential buildings with furniture
  - Known 2D schematic known for each building
- Tasks
  - Tracking - approximate initial robot pose is known
  - Global localization - initial belief is spread over entire environment
  - Semi-global localization - initial belief spread over 1 or more rooms
- Sensors: Monocular RGB, Depth, RGB-D, Simulated LIDAR, LIDAR-W

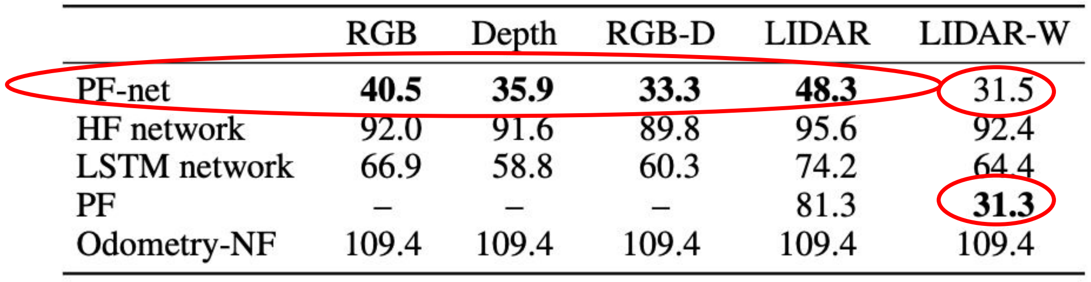
Experimental Setup

- Training
  - Tracking only
  - 45,000 randomly generated trajectories from 200 buildings
  - Backpropagation limited to 4 timesteps
- Comparison approaches
  - Histogram filter network
  - LSTM network
  - Conventional particle filter
  - Odometry only

Experimental Setup

- Evaluation
  - 820 trajectories in 47 unseen buildings
  - Tracking, semi-global localization, global localization
    - Varying numbers of particles
  - Resampling only for semi-global/global localization
- Metrics
  - Tracking - average root mean squared error
  - Semi-global/global localization - success rate
    - Estimation error below 1m for last 25 steps of 100 step trajectory

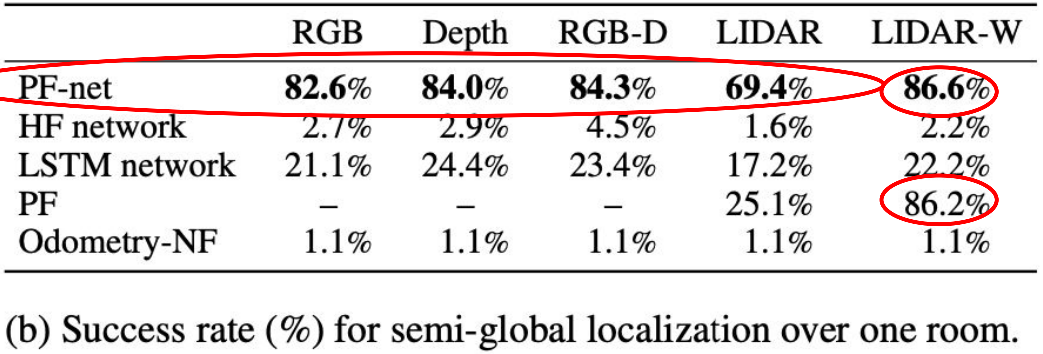
Experimental Results: Tracking



	RGB	Depth	RGB-D	LIDAR	LIDAR-W
PF-net	**40.5**	**35.9**	**33.3**	**48.3**	31.5
HF network	92.0	91.6	89.8	95.6	92.4
LSTM network	66.9	58.8	60.3	74.2	64.4
PF	–	–	–	81.3	**31.3**
Odometry-NF	109.4	109.4	109.4	109.4	109.4

(a) RMSE (cm) for tracking.

Experimental Results: Semi-global Localization



	RGB	Depth	RGB-D	LIDAR	LIDAR-W
PF-net	**82.6%**	**84.0%**	**84.3%**	**69.4%**	**86.6%**
HF network	2.7%	2.9%	4.5%	1.6%	2.2%
LSTM network	21.1%	24.4%	23.4%	17.2%	22.2%
PF	–	–	–	25.1%	86.2%
Odometry-NF	1.1%	1.1%	1.1%	1.1%	1.1%

(b) Success rate (%) for semi-global localization over one room.

Experimental Results: Global Localization

K	$N = 1$	$N = 2$	All
500	80.0%	70.5%	46.1%
1,000	84.3%	80.1%	57.9%
2,000	87.3%	84.8%	68.5%
3,000	**89.0%**	**85.9%**	**76.3%**

(c) PF-net success rate (%) for semi-global and global localization, with K particles. The initial belief is uniform over $N = 1$, $N = 2$, or all rooms.

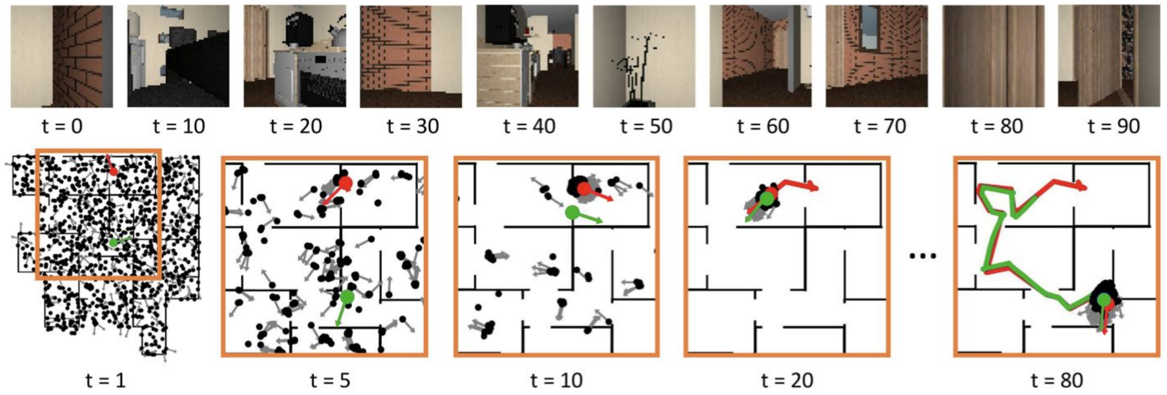
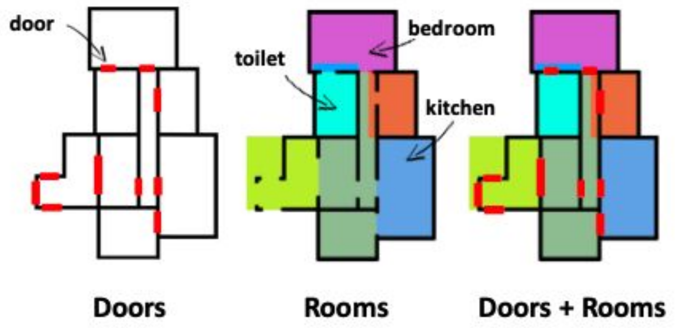


Figure 6: Global localization with PF-net ($K = 1000$) and RGB camera input. Red indicates ground-truth poses. Black indicates PF-net particles. Green indicates the weighted mean of particles.

Experimental Results: Semantic Labels



Labels	RGB	Depth	RGB-D
None	82.6%	84.0%	84.3%
Doors	84.4%	83.9%	84.5%
Rooms	84.5%	86.2%	86.5%
Both	**84.6%**	**86.7%**	**87.2%**

(d) PF-net success rate (%) for semi-global localization with additional semantic information on doors, rooms types, or both.

Experimental Results: Ablation

- Resampling in training

Percent Success on Semi-global Localization

	No resampling	Resampling	Alternate Step Resampling
Low uncertainty	79%	75%	-
High uncertainty	39%	42%	54%

- Backpropagation steps
- Loss function

Experimental Results: Ablation

- Resampling
- Backpropagation steps

Number of steps	1	2	4
Percent success	73%	79%	79%

- Loss function

Experimental Results: Ablation

- Resampling
- Backpropagation steps
- Loss function

Percent Success on Semi-global Localization

	Original Loss Function	Probabilistic Loss Function $-\log \mathrm{E}_t[\mathrm{bel}(\boldsymbol{s}_t^*; \boldsymbol{\theta})]$
Low Uncertainty	79%	74%
High Uncertainty	39%	67%

Discussion and Conclusion

Discussion of Results

- **Differentiable algorithmic priors are useful**
  - LSTM performs worse due to lack of inductive bias

Discussion of Results

- **Differentiable algorithmic priors are useful**
- **End-to-end learning increases robustness**
    - Test on very similar environments to training - unclear how well it generalizes
    - Additional possible tests: more datasets/domains, real robot

Discussion of Results

- **Differentiable algorithmic priors are useful**
- **End-to-end learning increases robustness**
- **PF-net is effective with various sensors**
  - Computation time
    - Effectiveness includes if computation time is tractable
    - Additional tests: Compare computation time for other approaches
  - Convergence details
    - Different thresholds, time to convergence

Limitations

- Only supports sequential optimization
  - Many modern SLAM/localization systems optimize the full trajectory
- Requires ground truth poses from many trajectories
  - Only feasible to train in simulation
  - Unclear how well this will generalize to different types of environments (other simulations or real world)

Future Work

- Learn noise parameters
- Real-world deployment
  - How to adapt sim-trained approach to real world
  - Study balance of training data needed vs. accuracy
- Self-supervised or sparse training signals
- Test on applications other than visual localization

Extended Readings

- Differentiable SLAM-net: Learning Particle SLAM for Visual Navigation
  - Embeds FastSLAM algorithm (Rao-Blackwellized particle filter) in differentiable computation graph
- End-To-End Semi-supervised Learning for Differentiable Particle Filters
  - Reduces demand for annotated data by enabling end-to-end optimization using sparse ground truth
- Towards Differentiable Resampling
  - Replaces resampling approximation with learned resampler
- Differentiable Mapping Networks: Learning Structured Map Representations for Sparse Visual Localization
  - Integrates differentiable particle filter with learned map representation

Summary

- PF-net proposes an end-to-end learned model for sequential state estimation
- End-to-end learning enables training of observation models that can be hard to hand-craft
- Prior works either lose the benefits of the particle filter structure or fail to make the network fully differentiable
- PF-net combines benefits of particle filter structure and end-to-end learning through algorithmic prior and differentiable approximations to particle filter steps
- Achieved improved localization performance compared to other deep-learned approaches and traditional particle filter

Thank you!

Questions?