Particle Filter Networks: End-to-End Probabilistic Localization From Visual Observations

Peter Karkus[1,2], David Hsu[1,2] and Wee Sun Lee[2]

***Abstract*— Particle filters sequentially approximate posterior distributions by sampling representative points and updating them independently. The idea is applied in various domains, e.g. reasoning with uncertainty in robotics. A remaining challenge is constructing probabilistic models of the system, which can be especially hard for complex sensors, e.g. a camera. We introduce the Particle Filter Networks (PF-nets) that encode both a learned probabilistic system model and the particle filter algorithm in a single neural network architecture. The unified representation allows learning models end-to-end, circumventing the difficulties of conventional model-based methods. We applied PF-nets to a challenging visual localization task that requires matching visual features from camera images with the geometry encoded in a 2-D floor map. In preliminary experiments end-to-end PF-nets consistently outperformed alternative learning architectures, as well as conventional model-based methods.**

I. Introduction

Particle filters, also known as sequential Monte-Carlo methods, build on the powerful idea of representing any probability distribution by a set of particles, i.e. weighted samples from the distribution [1]. They are extensively used in various domains like robotics [2], [3], and computer vision [4], [5], as well as statistics, physics, econometrics and finanance [6], [7], [8]. Sample-based representations are important in robotics as we must deal with uncertainties and thus reason with different classes of probability distributions. Indeed, they are extensively used in algorithms for localization [2], SLAM [9], and planning under partial observability [10]. The fundamental limitation of these methods is that they require a probabilistic model of the system. Unfortunately, constructing probabilistic models manually, or learning them from data remains difficult [11], [12], [13].

An emerging line of recent work integrates algorithmic structures into deep neural networks and learns models in an end-to-end fashion, circumventing the difficulty of traditional model learning [14], [15], [16], [17], [18], [19], [20]. Some of these works address uncertainty and explicitly incorporate filtering in the form of Kalman filters [15] and histogram filters [16], [17], [21]. These filtering methods represent the probability distribution by a Gaussian and a discrete histogram, respectively, and thus they are unsuitable for many problems arising in robotics.

We introduce the Particle Filter Networks (PF-nets), a deep recurrent neural network (RNN) architecture that encodes the particle filter algorithm together with learned probabilistic transition and observation models (Fig. 1). Both the algorithm and the models are captured in a unified network representation, which allows training end-to-end, i.e. optimizing the models jointly for the final task objective. In contrast to previous work, the particle representation can efficiently approximate arbitrary, continuous distributions, making PF-nets suitable for a wide range of problems. Here we apply them to the challenging task of localizing a robot on a map from partial, visual observations (Fig. 2).

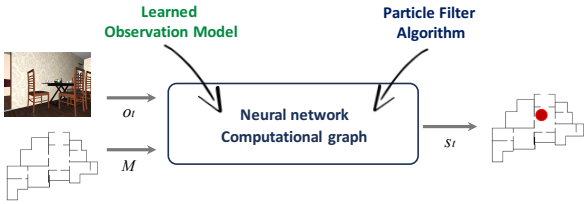

Fig. 1: PF-nets employ *algorithmic priors* for probabilistic state estimation: both the learned model and the particle filter algorithm are encoded in a unified representation.

A robot is navigating in a previously unseen environment, while it does not know its exact location. The task is to periodically estimate the location given a 2-D floor map and the history of observations from onboard sensors, e.g. a camera. Particle filters are considered to be the industry standard for localization with laser rangefinders [2]. Here we consider visual sensors instead of rangefinders. The difficulty now is to obtain a probabilistic observation model that extracts information from rich camera images and match it with the 2-D geometry encoded in the floor map. We may learn such a model from data, but the concern is the consequence of model errors on state estimation, and the lack of suitable training data. Instead of learning the model directly, we propose to encode it in a neural network architecture, together with the transition model and the inference algorithm, and train end-to-end, from sequences of observations to state estimates. The models are now optimized jointly for the final objective; and they are trained on easy-to-obtain trajectory data instead of ground-truth observation probabilities.

Preliminary results on the House3D dataset [22] show that PF-nets are effective for visual localization, and learned models generalize to new environments. Through end-to-end training, PF-nets may learn models that are mode robust against objects in the environment than conventional model-based methods. The unified neural network representation provides an opportunity to fuse information from different sensors; and to integrate geometric and semantic information

[1]NUS Graduate School for Integrative Sciences and Engineering
[2]School of Computing
National University of Singapore, 119077, Singapore.
{karkus, dyhsu, leews}@comp.nus.edu.sg

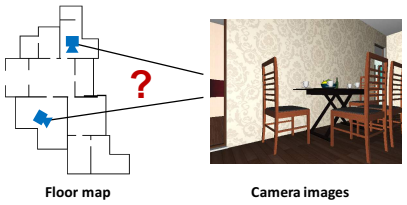

Fig. 2: Visual localization. Localize a robot given a 2D map and images from an onboard camera. The observation model must match visual features to the map geometry, while ignoring objects that are not indicated on the map.

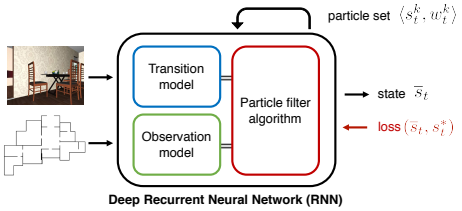

Fig. 3: PF-nets encode the transition and observations models together with the particle filter algorithm in a single RNN, where the particle set corresponds to the RNN hidden state.

for localization. The particle representation in PF-nets, when combined with appropriate algorithmic structures, may also allow scaling partially observable planning to large state spaces in the future.

II. Related work

Integrating algorithmic priors with deep neural networks has been gaining attention recently. The idea was applied to decision making in the context of value iteration in Markov decision processes [14], searching in graphs [18], [19], [23], path integral optimal control [24], quadratic program optimization [20], [25], and planning under partial observability [17], [26]. Probabilistic state estimation was addressed by Haarnoja et al. who encoded Kalman filters in a neural network [15]. The probability distributions are limited to unimodal Gaussian. Jonschkowski et al. introduced end-to-end histogram filters [16], and similar filters were used by [17] and [21] for policy learning under partial observability. Histogram filters use a discrete representations and thus they are inherently limited to small state spaces. In contrast, we use particle filters that can efficiently approximate an unrestricted class of distributions over large, continuous spaces.

The problem of robot localization has been successfully tackled by particle filter methods, often called Monte-Carlo localization in this setting. Monte-Carlo localization typically assumes a laser rangefinder mounted on the robot, for which simple analytic observation models have been developed [27]. There has been attempts to incorporate monocular or depth cameras [28], [29], [30], [31], but obtaining a probabilistic observation model for particle filtering remains difficult. PF-nets encode both the probabilistic models and the particle filter algorithm in a single neural network, thus allowing to learn localization end-to-end, from camera images to state estimates. We do not require direct supervision on the models, but instead expect effective models to emerge through end-to-end training.

III. Particle filter networks

We introduce the Particle Filter Networks (PF-nets) for learning probabilistic state estimation. PF-nets encode the particle filtering algorithm and the transition and observation models in a single, deep RNN architecture, and thus combine the benefits of algorithmic reasoning and deep learning (Fig. 3 and Fig. 4). Encoding the algorithm in the network allows jointly learning the transition and observation models end-to-end, optimized directly for the state estimation objective, and thus circumventing the difficulties of traditional model-based learning. The neural network representation also provides a principled approach to sensor fusion. We may simply add multiple observation model components, or jointly learn an observation model for multiple sensor modalities, end-to-end. Compared to generic architectures such as LSTM networks [38], PF-nets explicitly encode structure for sequential probabilistic estimation, i.e. the particle representation of probability distributions, and Bayesian updates for transitions and observations. In this section we introduce the details of the PF-net architecture. We apply PF-nets to visual localization, but the approach could be used in other state estimation domains in future work.

A. Particle filter algorithm

Particle filters estimate the state of the system by periodically approximating the posterior distribution over states after an observation is received, i.e. the belief $b_t(s)$ at time t. The belief is approximated by a set of *particles*, i.e. weighted samples from the probability distribution,

$$b_t(s) \approx \langle s_t^k, w_t^k \rangle, \tag{1}$$

where $\sum_k w_k = 1$ and $k = 1..K$. Here K denotes the number of particles, s_k and w_k denote the particle state and weight, respectively. The particle representation can approximate an unrestricted class of probability distributions. At any point, the state can be estimated by the weighted mean of the particles,

$$\overline{s}_t = \sum_k w_t^k s_t^k. \tag{2}$$

The particles are periodically updated in a Bayesian manner when a new observation is received. First, the particle states are updated by sampling from a probabilistic transition model,

$$s_t^k \sim T(s|u_t, s_{t-1}^k), \tag{3}$$

where T is the transition model defining the probability of a state given a previous state, s_{t-1}^k, and odometry reading, u_t. The odometry is assumed to have Gaussian noise, and transitions are typically modeled by a Gaussian too.

Second, the observation likelihood, f_t^k, is computed for each particle,

$$f_t^k = Z(o_t|s_t^k; \mathbb{M}), \tag{4}$$

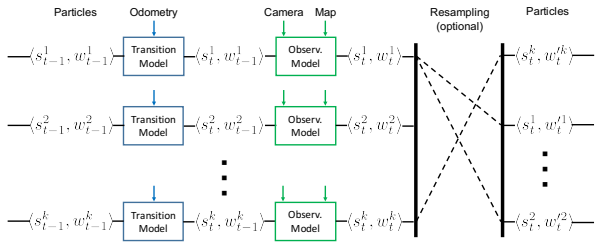

Fig. 4: Update process in PF-nets. The transition and observation models are both learnable neural network components. The weights are shared across particles.

where the observation model, Z, defines the conditional probability of an observation, a camera image, o_t, given a state, s_t^k, and the 2-D floor map, $\mathbb{M}$. Particle weights are then updated according to the likelihoods,

$$w_t^k = f_t^k w_{t-1}^k c^{-1}, \tag{5}$$

where $c = \sum_{j \in K} f_t^j w_{t-1}^j$ is a normalization factor.

One common issue with particle filtering is particle degeneracy, when weights become near-zero for several particles. The problem is typically addressed by *resampling*. We sample a new set of particles from the current set with repetition, where a particle is chosen with the probability defined by its weight,

$$p(k) = w_t^k. \tag{6}$$

The weights are then updated to a uniform distribution,

$$w_t'^k = 1/K. \tag{7}$$

The new particle set approximates the same distribution, but devotes its representation power to the important regions of the space. The set may contain repeated particles, but they will later diverge due to the stochastic transition updates.

B. Neural network implementation

We implement the particle filter algorithm as an RNN, where the hidden state of the RNN corresponds to the particle set. The transition and observation models are represented as neural network component and can be learned from data. See Fig. 3 for the overall architecture and Fig. 4 for the detailed structure of an update step.

The key idea for implementing the particle filter algorithm as an RNN is to view neural networks as computation graphs, or *differentiable programs*, and implement the algorithmic steps (1)-(5) in a differentiable manner, acting on tensors in the computational graph. Most of the operations in particle filtering are differentiable, e.g. multiplication and addition. However, in the general case we must address non-differentiable operations, such as sampling from a learned distribution in (11), and resampling particles in (6)-(7). We may use a differentiable approximation of both of these operations as follows.

We use a Gaussian distribution for the transition model, T, as it is usually done in the robot localization domain. The parameters of the distribution can be learned if we turn sampling from a learned Gaussian to a differentiable operation, e.g. by the "reparameterization trick" used in Variational Autoencoders [32].

The issue with the resampling operation is that new particle weights are set to constant, which in turn produces zero gradients. We address the problem by introducing *soft-resampling*, a differentiable approximation of the resampling operation. Unlike in (6), we sample particles with probability

$$q(k) = \alpha w_t^k + (1 - \alpha) 1/K, \tag{8}$$

a combination of the particle weights and a constant corresponding to a uniform distribution, where α is a trade-off parameter. The new weights are then computed by the importance sampling formula,

$$w_t'^k = w_t^k q(k)^{-1}, \tag{9}$$

which has a non-zero gradient when $\alpha \neq 1$. Soft-resampling trades off the optimal sampling distribution ($\alpha = 1$) with the uniform sampling distribution ($\alpha = 0$), maintaining the dependency on previous particle weights and thus providing gradients for training.

We have now all operations implemented in a differentiable manner. Given a particular task we must choose the representation of states, the network architecture for T and Z, and the number of particles, K. Note that we can choose different K settings for training and evaluation. In the next section we discuss our design choices for the visual localization task.

IV. Visual Localization

Consider a differential drive robot navigating in an indoor environment it has not seen before. The robot is equipped with an onboard camera and odometry sensors, but it is uncertain of its location. The task is to periodically estimate the location from the history of past observations given a 2-D floor map encoding the building geometry. Formally, we seek to minimize the mean squared difference of the 2-D pose estimate and the actual robot pose over the trajectory,

$$\min \sum_t (\overline{x}_t - x_t^*)^2 + (\overline{y}_t - y_t^*)^2 + \beta(\overline{\phi}_t - \phi_t^*)^2 \tag{10}$$

where $\overline{x}_t, \overline{y}_t, \overline{\phi}_t$ and x_t^*, y_t^*, ϕ_t^* are the estimated and true robot pose for time t, respectively; and β is a constant parameter. Challenges are two-fold. First, we must periodically update a posterior over states given ambiguous observations, where the posterior is a non-standard, multimodal distribution over a continuous space. Second, we need to obtain the probability of a camera image given a state and a map. This involves extracting and comparing the building geometry from the 2-D floor map and the 3-D scene captured by the camera.

We apply PF-nets to this visual localization task. PF-nets address both key challenges: they explicitly encode the structure of particle filtering for sequential estimation of a non-standard, continuous distribution; and they represent the observation model as a neural network, which allows learning appropriate features end-to-end from data.

We define the state in the PF-net as the robot pose, $s = \{x, y, \phi\}$, where x and y are the robot coordinates, and ϕ is

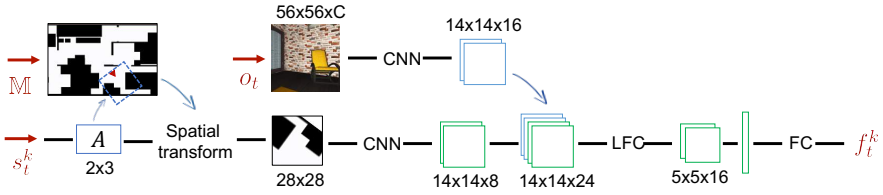

Fig. 5: Observation model in the PF-nets for localization. Inputs are the floor map $\mathbb{M}$, observation o_t at time t, and a particle state s_t^k. Output is the particle likelihood f_t^k. CNN, LFC and FC denote a convolutional, locally-fully-connected, and fully connected component, respectively.

the orientation. The particles are then pairs of a candidate pose and the corresponding weight. We used $K = 30$ particles for training, and $K = 30..100$ particles for evaluation. The transition and observation models are differentiable neural networks with appropriate structure.

The transition model propagates the particle state given the relative motion, u_t, defined by the odometry in the robot coordinate frame. We transform the relative pose to the global coordinate frame, u'_t, and add it to the state along with Gaussian noise. The transition model is then

$$T(s_t^k | u'_t, s_{t-1}^k) = \mathcal{N}(s_{t-1}^k + u'_t; \mathrm{diag}(\sigma_t, \sigma_t, \sigma_r)), \quad (11)$$

where $\mathcal{N}(\mu; \Sigma)$ denotes a multivariate Gaussian distribution, and σ_t and σ_r are the standard deviation for translation and rotation, respectively. In preliminary experiments we manually set σ_t and σ_r, but in the future we may learn them from data, optimizing together with the observation model.

The observation model defines the probability of an observation, o_t, given the particle state, s_t^k, and the floor map $\mathbb{M}$. We use the network shown in Fig. 5. First, a *local map* is obtained from $\mathbb{M}$ and s_t^k through a spatial transformation component. We adopt the Spatial Transformer Network [34], that implements affine image transformation as a neural network. Unlike in [34], we define the affine transformation matrix, A, for each particle such that the transformed image is a local view from the pose s_t^k. We then extract features from the local map and the observation input through two separate convolutional components (CNN). The feature maps are concatenated and fed into a locally fully connected layer, reshaped to a vector, and fed to a final fully connected layer. The output is a scalar, f_t^k, the likelihood of the particle state. The likelihood is obtained for each particle and the particle weights are updated through (5). We note that the observation features only need to be computed once for all particles; and the network applies to floor maps of any size.

The updates are applied to each particle and the particle weights are normalized. We can then resample using (8)-(9); or simply carry over the particles to the next step. In our experiments we did not use resampling during training; however, we believe it can be beneficial when training with long trajectories and high uncertainty in future work.

V. Experiments

We evaluated PF-nets in preliminary simulation experiments where they successfully learned the challenging localization task. Through end-to-end training, learned observation models may incorporate multiple sensor modalities and generalize to new environments. Compared to conventional model-based methods, PF-nets can learn more effective models end-to-end.

A. Experimental setup

We conducted experiments in the House3D virtual environment [22], a 3-D simulator with a large set of human-designed, realistic residential buildings from the SUNCG dataset [35].

We generated $45k$ random, collision-free trajectories of length 24 in 200 buildings for training, and 830 trajectories from a separate set of 47 buildings for evaluation. The robot either moves forward with probability $p = 0.8$, or rotates with probability $p = 0.2$. The distance and rotation angle are sampled uniformly in the range $[20\text{cm}, 80\text{cm}]$ and $[15°, 60°]$, respectively[1]. The initial belief and the underlying initial state are also chosen at random. We pick a point and sample the initial state and initial particles from a multivariate Gaussian centered at this point, using $\Sigma = \mathrm{diag}(30\text{cm}, 30\text{cm}, 30°)$. We rendered camera images along the trajectory and obtained a floor map from the simulator. The floor map is a top-down view indicating architectural elements, i.e. walls. The map does not show other objects in the environments, e.g. furniture. The observation model must recognize and ignore these objects for geometry-based localization.

We considered two variants of the task: a simplified variant where only walls are rendered on the camera image (*walls only* in Table I); and one where all objects are rendered (*all objects*). The latter setting is hard for geometry-based localization, because most of the visible objects are not shown on the map, and thus object distances from the camera do not directly correspond to distances on the map. We tried different sensors in separate experiments: depth camera (*depth*), monocular camera (*RGB*), and the combination of the two, an RGB-D camera (*RGB-D*). For comparing with conventional laser models we used a virtual rangefinder (*laser*), a transformation of the 2-D depth-map to a 1-D virtual laser-scan.

We implemented PF-nets in Tensorflow [36] and trained by backpropagation through time using the RMSProp optimizer [37].

[1]We sample a new action if the forward motion would result in a collision.

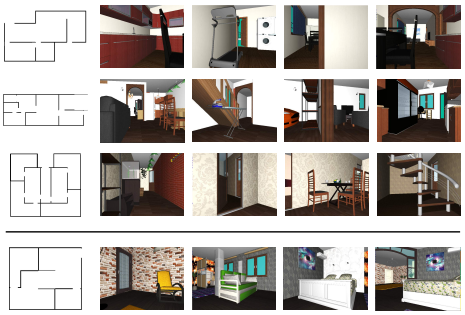

Fig. 6: Example maps and RGB camera images picked from the training (top) and test (bottom) environments. Maps are not to scale. All objects are rendered on the camera images, but not on the floor map.

B. Baselines of comparison

1) Propagated uncertainty: Propagated uncertainty indicates the estimation error due to the propagated initial uncertainty. We only perform transition updates but no observation update, thus the uncertainty is never reduced.

2) Laser-model PF: This model-based baseline performs particle filtering with a manually constructed model for laser rangefinders. There are two models often used in practice: the beam model and the likelihood field model [27]. We used an implementation of the beam model from the AMCL package of ROS [2], which can be considered the industry standard. For a fair comparison, we tuned the model parameters using virtual rangefinder data generated in the *wallsonly* setting.

3) Histogram filter network: To evaluate the importance of the particle representation in a network learning architecture we compared to the end-to-end histogram filters [16], [17], adapted to visual localization. Histogram filter networks employ algorithmic priors for probabilistic state estimation similar to PF-nets, but the belief is represented by a histogram, i.e. table of probability values over discrete states. The histogram representation does not scale well to a large state-space. In our histogram network we discretize the states-space to the size of the floor map scaled down by a factor of 20; and 16 discrete orientations. We could not achieve better results with finer discretization.

4) LSTM network: To further evaluate the importance of belief representation in our network we compared to an LSTM architecture that employs priors for map-based localization but does not employ algorithmic priors specific to probabilistic state estimation. The network receives an image and a local map at each step and outputs a relative state estimate. The relative state is added to the previous state, and the map is transformed to the local view from the updated state for the next step. In addition, we tried multiple variants of vanilla LSTM formulations that directly map from global map, observation and current state to next state, but we did not achieve success with such architectures.

C. Results and discussion

Preliminary results are summarized in Table I. We report the Root Mean Squared Error (RMSE) error computed for the x, y components, averaged over a fixed set of 830 test trajectories in 47 previously unseen environments. The test trajectories are 24 steps long and are generated identically to the training trajectories.

TABLE I: Summary of results comparing PF-nets with alternatives for different sensor modalities. We report RMSE in cm. K denotes the number of particles for particle filtering.

	Walls only		**All objects**			
	Laser	**Depth**	**Laser**	**Depth**	**RGB**	**RGB-D**
Propagated uncertainty	109.37	109.37	109.37	109.37	109.37	109.37
Laser-model PF, $K=300$	**31.28**	–	81.34	–	–	–
Histogram filter network	92.36	85.88	95.62	91.55	91.99	89.82
LSTM network	64.44	46.06	74.24	58.76	66.92	60.31
PF-net (ours), $K=30$	52.12	47.19	62.77	51.34	52.22	51.91
PF-net (ours), $K=300$	31.52	**25.45**	**48.33**	**35.92**	**40.47**	**33.31**

Learned PF-nets successfully reduce the state uncertainty using visual observations, even when the environment is populated with distracting furniture and household objects. The algorithmic priors in PF-nets, i.e. the particle filter algorithm, is beneficial for sequential state estimation: PF-nets consistently outperformed alternative learning architectures with weaker priors (*Histogram filter network*, *LSTM network*).

For laser sensor the learned PF-net models are more robust compared to conventional model-based methods (*laser*). When only walls were rendered in the environment they perform similarly, but learned models are significantly better when additional objects are present. The conventional laser model decouples the laser-scan to individual laser beams, and thus it has no principled way to distinguish walls from objects other than treating them as outlier detections. In contrast, PF-nets learn a model end-to-end. The laser-scan is processed jointly and thus the learned model may use relationship between laser-beams to detect objects. The end-to-end approach also enables learning to compensating for imperfect maps, e.g. missing walls, glass windows, or artifacts at the edge of the map, which we observed in the House3D dataset.

Learned depth and RGB models performed better than laser models, indicating that PF-nets learned to incorporate the richer information encoded in depth and RGB images. RGB models perform worse than depth models, but the gap is not large. This suggests that the deep neural network learned to extract geometry from RGB images, a challenging task on its own. For the combined RGB-D sensor we simply concatenated the depth and RGB inputs. The performance did not improve significantly compared to depth-only images; nevertheless, we believe the end-to-end approach may learn to integrate information from multiple sensor with a better network architecture for sensor-fusion.

For now we only considered localization in a tracking setting where the initial state uncertainty is relatively low. Future work should investigate more difficult localization settings with large initial uncertainty. One limitation of our approach is that we need to compute observation likelihoods for each particle. This can be computationally expensive for complex models and many particles, while large uncertainty usually requires 1,000 - 10,000 particles for indoor robot localization [2]. Note that the computational complexity

comes from the nature of particle filtering rather than the neural network representation.

VI. Conclusion

We proposed to integrate algorithmic reasoning and deep learning for probabilistic state estimation, by encoding the particle filter algorithm and a probabilistic model in a unified network learning architecture. Preliminary results in a visual localization domain are promising.

Future work may extend our experiments with PF-nets to real-world visual localization, a problem of great interest for mobile robot applications. PF-nets could be applied to other domains as well, e.g. visual object tracking or simultaneous localization and mapping. Finally, we believe the particle representation will be important for encoding more sophisticated algorithms in neural networks, which may enable truly large-scale reasoning, e.g. for planning under partial observability.

References

[1] A. Doucet, N. De Freitas, and N. Gordon, "An introduction to sequential monte carlo methods," in *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 3–14.

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[3] S. Thrun, "Particle filters in robotics," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 511–518.

[4] A. Blake and M. Isard, "The condensation algorithm-conditional density propagation and applications to visual tracking," in *Advances in Neural Information Processing Systems*, 1997, pp. 361–367.

[5] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[6] P. Del Moral, A. Doucet, and A. Jasra, "Sequential monte carlo samplers," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 3, pp. 411–436, 2006.

[7] J. S. Liu, *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[8] A. Doucet, N. De Freitas, and N. Gordon, "Sequential monte carlo methods in practice. series statistics for engineering and information science," 2001.

[9] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[10] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online POMDP planning with regularization," *Journal of Artificial Intelligence Research*, vol. 58, pp. 231–266, 2017.

[11] G. Shani, R. I. Brafman, and S. E. Shimony, "Model-based online learning of POMDPs," in *European Conference on Machine Learning*, 2005, pp. 353–364.

[12] B. Boots, S. M. Siddiqi, and G. J. Gordon, "Closing the learning-planning loop with predictive state representations," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 954–966, 2011.

[13] L. Getoor, N. Friedman, D. Koller, and B. Taskar, "Learning probabilistic models of link structure," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 679–707, 2002.

[14] A. Tamar, S. Levine, P. Abbeel, Y. Wu, and G. Thomas, "Value iteration networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2146–2154.

[15] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," in *Advances in Neural Information Processing Systems*, 2016, pp. 4376–4384.

[16] R. Jonschkowski and O. Brock, "End-to-end learnable histogram filters," in *Workshop on Deep Learning for Action and Interaction at NIPS*, 2016. [Online]. Available: http://www.robotics.tu-berlin.de/fileadmin/fg170/Publikationen_pdf/Jonschkowski-16-NIPS-WS.pdf

[17] P. Karkus, D. Hsu, and W. S. Lee, "QMDP-net: Deep learning for planning under partial observability," in *Advances in Neural Information Processing Systems*, 2017, pp. 4697–4707.

[18] J. Oh, S. Singh, and H. Lee, "Value prediction network," in *Advances in Neural Information Processing Systems*, 2017, pp. 6120–6130.

[19] G. Farquhar, T. Rocktäschel, M. Igl, and S. Whiteson, "TreeQN and ATreeC: Differentiable tree planning for deep reinforcement learning," *arXiv preprint*, 2017. [Online]. Available: https://arxiv.org/abs/1710.11417

[20] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, 2017, pp. 136–145.

[21] D. S. Chaplot, E. Parisotto, and R. Salakhutdinov, "Active neural localization," *arXiv preprint arXiv:1801.08214*, 2018.

[22] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," *arXiv preprint arXiv:1801.02209*, 2018.

[23] A. Guez, T. Weber, I. Antonoglou, K. Simonyan, O. Vinyals, D. Wierstra, R. Munos, and D. Silver, "Learning to search with MCTSnets," *arXiv preprint*, 2018. [Online]. Available: https://arxiv.org/abs/1802.04697

[24] M. Okada, L. Rigazio, and T. Aoshima, "Path integral networks: End-to-end differentiable optimal control," *arXiv preprint arXiv:1706.09597*, 2017.

[25] P. Donti, B. Amos, and J. Z. Kolter, "Task-based end-to-end model learning in stochastic optimization," in *Advances in Neural Information Processing Systems*, 2017, pp. 5484–5494.

[26] T. Shankar, S. K. Dwivedy, and P. Guha, "Reinforcement learning via recurrent convolutional neural networks," in *International Conference on Pattern Recognition*, 2016, pp. 2592–2597.

[27] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[28] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999, pp. 588–594.

[29] P. Elinas and J. J. Little, "σmcl: Monte-carlo localization for mobile robots with stereo vision," in *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[30] B. Coltin and M. Veloso, "Multi-observation sensor resetting localization with ambiguous landmarks," *Autonomous Robots*, vol. 35, no. 2-3, pp. 221–237, 2013.

[31] O. Mendez, S. Hadfield, N. Pugeault, and R. Bowden, "Sedar-semantic detection and ranging: Humans can localise without lidar, can robots?" *arXiv preprint arXiv:1709.01500*, 2017.

[32] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 573–580.

[34] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[35] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 190–198.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: http://tensorflow.org/

[37] T. Tieleman and G. Hinton, "Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, pp. 26–31, 2012.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.