Differentiable SLAM-nets: Learning Task-Oriented SLAM for Visual Navigation

Peter Karkus, Shaojun Cai, David Hsu
National University of Singapore
karkus@comp.nus.edu.sg

Abstract

This paper introduces the Differentiable SLAM Network (SLAM-net) that encodes a particle filter based SLAM algorithm in a differentiable computation graph, and learns task-oriented neural network models by backpropagating through the SLAM algorithm. Because of its ability to learn task-oriented models, differentiable SLAM can potentially overcome a number of important limitations of traditional SLAM methods, such as dynamic environments, sparse visual features, maps that allow downstream path planning or semantic tasks. Preliminary results in simulated indoor environments show a strong performance for SLAM-net, both for visual localization and downstream visual navigation. Our navigation architecture with SLAM-net improves the state-of-the-art for the Habitat PointNav Challenge 2020 task by a large margin (37% to 64% success). For an extension of this work visit the project website: http://sites.google.com/view/slamnet

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a crucial component of an autonomous navigation system. However, traditional SLAM methods have limitations in a number of aspects that may impact downstream navigation [5; 24; 7]. For example, feature extraction and association may fail due to featureless walls, noisy sensors or low frame rate; relocalization and loop closure could be difficult due to environmental changes and repetitive features; or the map representation may not be appropriate for the downstream planning task.

This paper introduces the Differentiable SLAM Network (SLAM-net). The key idea is to encode a SLAM algorithm in a differentiable computation graph and then learn neural network model components for the SLAM algorithm end-to-end, by backpropagating gradients through the algorithm. By learning task-oriented models, differentiable SLAM may overcome the limitations of traditional SLAM, e.g., we may optimize feature extractors for localization or optimize the map representation for downstream navigation. In this work we instantiate the idea for a particle filter based SLAM algorithm with occupancy grid map representation, and apply it to indoor localization and navigation. Our SLAM-net architecture benefits from end-to-end training to learn a robust discriminative observation model for low quality visual input, for which direct supervision would not be available.

A general framework for robot learning with differentiable algorithms has been proposed in prior work [20]. Differentiable algorithms have been developed for state estimation [12; 17; 19], mapping [11; 21], planning [33; 18; 9; 29; 10; 38] and control [1; 30; 8; 3]. Our differentiable SLAM-net fills a gap in this literature. Differentiating through SLAM has been proposed by Jatavallabhula et al. [16] and Tang and Tan [34]; however, the former does not utilize gradients for learning, and the latter focuses on learning the feature metric representation instead of the whole pipeline.

Learning appears in modern approaches for both SLAM [7] and visual navigation [24]. In the case of SLAM, most approaches only learn specific modules for specific decoupled learning objectives [35;

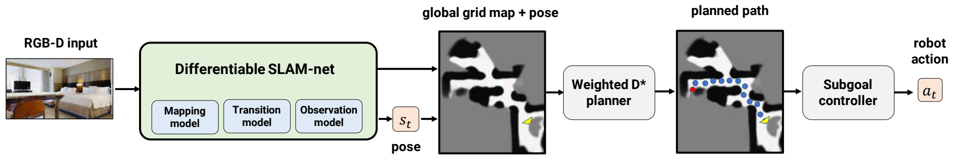

Figure 1: Visual navigation pipeline with the differentiable SLAM-net.

4; 39]. In the case of visual navigation, one line of work replaces the entire navigation pipeline with a deep neural network [36]. Others propose modular architectures with learned components, however, they either assume known location [11], known map [20], or they rely on simple visual odometry [6; 31], which unavoidably accumulates errors over time.

We validate SLAM-nets in preliminary experiments for both visual SLAM and downstream navigation in previously unseen indoor environments. We use the Habitat simulator [32] with the Gibson dataset [37], a physics simulator built on scans of real-world apartments. When evaluated for the localization task, SLAM-net significantly outperforms learned visual odometry as well as the the popular ORB-SLAM [28]. For downstream visual navigation we adopt a standard navigation pipeline, similar to to Neural SLAM [6], but using our differentiable SLAM-net module instead of visual odometry (Fig. 1). Our approach significantly improves the state-of-the-art for the CVPR Habitat 2020 PointNav challenge [13].

2 Differentiable SLAM-nets

The Differentiable SLAM-net architecture is shown in Fig. 2. Inputs are RGB(D) observations o_t, outputs are pose s_t and optional global map M_t. Internally SLAM-net represents the global map as a collection of local maps. Local maps are 2D occupancy grids associated with a local-to-global transformation. We add a local map for each observation, but without knowing the robot pose we do not know the local-to-global map transformation. Instead, the algorithm maintains a distribution over the unknown robot trajectory using particle filtering. This algorithm is similar to FastSLAM [25; 26]. Our differentiable implementation is built on PF-nets [19].

Algorithm. A particle filter maintains K weighted particle, where each particle represents a past trajectory $s_{0:t}$ of 2D poses and 1D orientations. At $t = 0$ all particle trajectories are set to the origin; particle weights are constant, and the local map collection is empty. In each time step a *mapping model* predicts a local occupancy map from the input observation, $m_t = f_\theta^{\text{map}}(o_t)$, and it is added to the collection. A probabilistic *transition model* estimates the relative motion given the last two inputs, $\Delta s_t^k \sim f_\theta^{\text{trans}}(o_t, o_{t-1}, a_{t-1})$. Each particle trajectory k is extended using samples from the transition model. Particle weights are then updated using an *observation model* that estimate the observation likelihood, $\log w_t^k = \log w_{t-1}^k + \log p(o_t|o_{1:t-1}, s_{1:t}^k)$. We approximate the likelihood by learned pairwise compatibility estimates that intuitively measure if two local maps were compatible if the particle trajectory was correct. Finally, a single pose estimate is output by taking the weighted sum of trajectory particles; and a global map is output by combining the collection of local grid maps along the mean trajectory estimation using simple 2D image transformations.

Differentiability. The key feature of SLAM-net is that it is differentiable so that its models can be trained end-to-end with gradient descent. Specifically, the mapping, transition and observation models of SLAM-net are neural networks, which can be learned together optimizing for the end-objective of localization accuracy and/or global map quality. To make the algorithm differentiable we use the reparameterization trick [22] to differentiate through samples from the transition model; and we use spatial transformers [15] for differentiable map transformations. The rest of the operations of the algorithm, as presented, are already differentiable. While not used in our experiments, differentiable particle resampling could be incorporated from prior work [19; 40]. Further, to make use of the differentiability of the SLAM algorithm for learning with limited GPU memory, the design choices on the map representation and the formulation of the observation model are important.

Transition model. The transition model is a CNN that takes in the concatenated current and last depth observations (and optionally the last robot action), and outputs parameters of a Gaussian Mixture Model with separate learned means and variances for the relative 2D pose and 1D orientation. The model is trained to maximize the log-likelihood of true relative poses along the training trajectories.

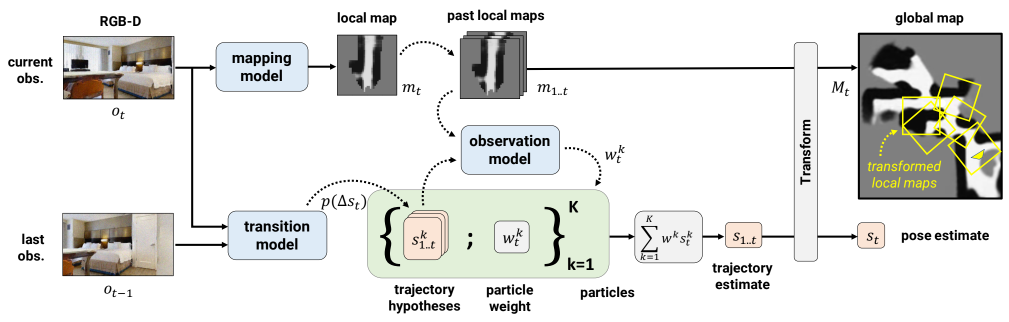

Figure 2: Differentiable SLAM-net. A global grid map is maintained by a collection of learned local grid maps. The trajectory is tracked by a particle filter. Particles represent trajectories and they are updated with learned neural network models (mapping, transition, and observation models).

Mapping model. The mapping model is a structured CNN with depth input and local map output. Local maps are $N \times N$ grids with 16+1 channels. Optionally, one channel can be supervised to predict local occupancy, i.e., the probability of the area in front of the robot being occupied. Other channels are latent, with no associated meaning, and are trained jointly with the observation model. For the occupancy supervision we use ground-truth maps and a pixel-wise classification loss.

Observation model. We learn a discriminative observation model that estimates the observation likelihood by comparing pairs of local maps, $\log p(o_t|o_{1:t-1}, s_{1:t}^k) \approx \sum_{\tau \in T} f_\theta^{\text{obs}}(m_t, s_t^k, m_{t-\tau}, s_{t-\tau}^k)$. Here f_θ^{obs} is a CNN that takes in two local maps and outputs a "compatibility" value, intuitively, measuring the extent to which the maps capture the same area. For each comparison $m_{t-\tau}$ is first transformed to the viewpoint of the current local map m_t given their relative pose in the particle trajectory, $s_{t-\tau}^k$ and s_t^k. The two maps are then concatenated and fed to the CNN. We sum the pairwise map compatibly values along the trajectory for past steps T. To limit computation we only consider the "most relevant" pairs in T. During training we pick the last 4–8 steps; during inference we dynamically choose 8 steps based on the largest overlapping view area (estimated using simple geometry). The algorithm estimates observation likelihoods for all particles this way, and updates the particle weights after normalization across particles.

Training. The training data consists of image-pose pair trajectories, and optionally, ground-truth occupancy maps for the mapping model. The end-to-end training loss is the sum of Huber losses for the 2D pose error and 1D orientation error. We train in multiple stages: pre-train a mapping module to predict occupancy; train an observation model in a low-noise setting (ground-truth transitions plus a small additive Gaussian noise, training for the end-objective); train a transition model separately; finally fine-tune all components jointly end-to-end. During training we use short trajectories (4-8 steps) and $K = 32$ particles without resampling. During evaluation we use long trajectories (up to 500 steps) and $K = 128..256$ particles with resampling.

3 Experiments

We evaluate SLAM-net for both visual localization and navigation in the Habitat environment [32]. We use the Habitat PointNav Challenge 2020 task where a robot navigates to goals in previously unseen apartments using noisy RGB-D observations. The goal is defined by coordinates relative to the initial pose, but the robot location is unknown thereafter, and discrete robot actions generate noisy motion. Navigation is successful if the robot takes a dedicated *stop* action less than 0.36 meters from the goal. Note that this success criteria places high importance on state estimation accuracy.

Localization. We split the Gibson data to 72 scenes for training (500 randomized trajectories each), 7 scenes for validation, 7 scenes for testing. For evaluation we generate three sets of 105 trajectories: using a shortest-path expert (**traj_expert**), the shortest-path expert mixed with random actions (**traj_exp_rand**); and our navigation pipeline described later (**traj_nav**). We report success rate (SR) that measures the percentage of the episodes where the final pose error is below 0.36 meters; and root-mean-square-error (RMSE) which measures the absolute trajectory error as defined in [14]. Estimated trajectories are aligned with the ground-truth at the beginning of each episode.

Method	traj_expert SR↑	traj_expert RMSE↓	traj_exp_rand SR↑	traj_exp_rand RMSE↓	traj_nav SR↑	traj_nav RMSE↓
SLAM-net (ours)	**83.8%**	**0.16m**	**62.9%**	**0.28m**	**77.1%**	0.19m
Learned Visual Odometry	60.0%	0.26m	24.8%	0.63m	30.5%	0.47m
ORB-SLAM [27]	3.8%	1.39m	0.0%	3.59m	0.0%	3.54m
Blind baseline	16.2%	0.80m	1.0%	4.13m	3.8%	1.5m

Table 1: Localization results.

Pose est. method	SR↑	SPL↑
SLAM-net (ours)	**65.7%**	**0.38**
Visual odometry	32.4%	0.19
Ground-truth	**90.7%**	**0.56**

Table 2: Navigation results.

Rank	Method	SR↑	SPL↑
1	**SLAM-net (ours)**	**64.5%**	**0.38**
2	VO	37.3%	0.27
3	OccupancyAnticipation	29.0%	0.22

Table 3: Habitat Challenge 2020 leaderboard [13].

Results are in Table 1. We compare with learned visual odometry, i.e., the transition model of SLAM-net used in isolation; the classic ORB-SLAM [27] with RGB-D input; and a blind baseline that accumulates the nominal motion for actions but ignores observations. We find that SLAM-net outperforms the other methods across all datasets by a large margin. ORB-SLAM performs particularly poorly. We believe this is due to the combined effects of rapid turns, sparse visual features, observation noise, and low frame rate. Indeed, if we remove noise and increase the frame rate ORB-SLAM performs well (up to 90.4%). Detailed results are in the Appendix. In an additional ablation study we found that joint finetuning is useful (77.1% vs. 66.7% without finetuning); it is possible to use only latent or only occupancy channels in local maps (70.5% latent only, 75.2% occupancy only, 77.1% both); and increasing the number of particles has diminishing benefit (60.0% for $K=8$, 72.4% for $K=32$, 77.1% for $K=128$, 82.9% for $K=512$).

Navigation. We integrate SLAM-net into the navigation pipeline shown in Fig. 1. SLAM-net predicts the robot pose and a global occupancy map in each time step. The map and pose are fed to a 2D path planner that plans a path to the goal. We choose a weighted variant of the D* algorithm [23] with costs that penalize moving near obstacles. The planned path is tracked by a simple controller that chooses to turn or move forward aiming for the furthest straight-line traversable point along the path. In addition we hard-code an initial $370°+$ turn and a collision recovery behavior, where an obstacle is added to the map and the robot turns around and takes a step back.

Results are in Table 2. We report success rates (SR) and success weighted path length (SPL) defined in [2]. We test different methods for estimating the robot pose, while keeping the rest of the architecture, including the learned map prediction model, fixed. Performance with ground-truth localization is strong, which validates the rest of our navigation architecture and serves as an upper bound for SLAM. Performance drops significantly for visual odometry. Note that our navigation architecture with visual odometry is similar to that of Neural SLAM [6] and Occupancy Anticipation [31], the winning entries to the 2019 and 2020 PointNav challenges. In comparison, SLAM-net achieves a much stronger performance.

We have also submitted our approach to be evaluated on a closed test set. Leaderboard results are in Table 3. SLAM-net achieves 64.5%success, significantly improving over the SOTA (VO, 37.3%) and the 2020 challenge winner (OccupancyAnticipation [31], 29.0%).

4 Conclusion

Differentiable SLAM opens up opportunity for a wide range of interesting applications beyond point-goal navigation. For example, one could learn to relocalize in dynamic environments with large difference between the time of mapping and localization; learn more robust features for low quality observations; or learn task oriented semantic map representations for downstream task, such as semantic navigation or visual question answering. In ongoing work we successfully trained SLAM-net with RGB only input, and transferred trained models to new datasets. Extended results and an Appendix is available on the project website: `http://sites.google.com/view/slamnet`

References

[1] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter. Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, pages 8299–8310, 2018.

[2] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[3] M. Bhardwaj, B. Boots, and M. Mukadam. Differentiable gaussian process motion planning. In *International Conference on Robotics and Automation (ICRA)*, pages 10598–10604, 2020.

[4] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2560–2568, 2018.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[6] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations*, 2020.

[7] C. Chen, B. Wang, C. X. Lu, N. Trigoni, and A. Markham. A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence. *arXiv preprint arXiv:2006.12567*, 2020.

[8] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon. Infinite-horizon differentiable model predictive control. In *International Conference on Learning Representations*, 2019.

[9] G. Farquhar, T. Rocktäschel, M. Igl, and S. Whiteson. TreeQN and ATreeC: Differentiable tree planning for deep reinforcement learning. In *International Conference on Learning Representations*, 2018.

[10] A. Guez, T. Weber, I. Antonoglou, K. Simonyan, O. Vinyals, D. Wierstra, et al. Learning to search with MCTSnets. In *International Conference on Machine Learning*, 2018.

[11] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 2017.

[12] 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.

[13] Habitat. Leaderboard for the CVPR Habitat 2020 PointNav challenge, 2020. URL `https://evalai.cloudcv.org/web/challenges/challenge-page/580/leaderboard/1631`. Accessed: 2020 November 16.

[14] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *International Conference on Robotics and Automation (ICRA)*, pages 1524–1531, 2014.

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

[16] K. M. Jatavallabhula, G. Iyer, and L. Paull. gradslam: Dense slam meets automatic differentiation. *arXiv preprint arXiv:1910.10672*, 2019.

[17] R. Jonschkowski, D. Rastogi, and O. Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *Proceedings of Robotics: Science and Systems*, 2018.

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

[19] P. Karkus, D. Hsu, and W. S. Lee. Particle filter networks with application to visual localization. In *Proceedings of the Conference on Robot Learning*, pages 169–178, 2018.

[20] P. Karkus, X. Ma, D. Hsu, L. P. Kaelbling, W. S. Lee, and T. Lozano-Pérez. Differentiable algorithm networks for composable robot learning. In *Robotics: Science and Systems (RSS)*, 2019.

[21] P. Karkus, A. Angelova, V. Vanhoucke, and R. Jonschkowski. Differentiable mapping networks: Learning structured map representations for sparse visual localization. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[22] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[24] D. Mishkin, A. Dosovitskiy, and V. Koltun. Benchmarking classic and learned navigation in complex 3d environments. *arXiv preprint arXiv:1901.10915*, 2019.

[25] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *AAAI Conference on Artificial Intelligence*, 593598, 2002.

[26] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *IJCAI*, pages 1151–1156, 2003.

[27] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[28] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[29] J. Oh, S. Singh, and H. Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, 2017.

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

[31] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman. Occupancy anticipation for efficient exploration and navigation. *arXiv preprint arXiv:2008.09285*, 2020.

[32] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, et al. Habitat: A platform for embodied AI research. In *International Conference on Computer Vision (ICCV)*, 2019.

[33] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, 2016.

[34] C. Tang and P. Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.

[35] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6243–6252, 2017.

[36] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations*, 2020.

[37] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. Gibson dataset license agreement available at: `http://svl.stanford.edu/gibson2/assets/GDS_agreement.pdf`.

[38] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki. Path planning using neural a* search. *arXiv preprint arXiv:2009.07476*, 2020.

[39] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11776–11785, 2019.

[40] M. Zhu, K. Murphy, and R. Jonschkowski. Towards differentiable resampling. *arXiv preprint arXiv:2004.11938*, 2020.