Fast Differentiable Particle Filters for Visual-Lidar Localization

Raphael Alluson[†]
Robotics Department
University of Michigan
Ann Arbor, USA
allusson@umich.edu

Saipranav Janyavula[†]
Robotics Department
University of Michigan
Ann Arbor, USA
sjanyavu@umich.edu

Siddharth Kotapati[†]
Robotics Department
University of Michigan
Ann Arbor, USA
sidkot@umich.edu

***Abstract*—We investigate the practical deployment of Differentiable Particle Filters (DPFs) for visual-lidar localization on low-compute robotic platforms. While DPFs have demonstrated promise in simulation, their real-world feasibility remains underexplored. Using the MBot platform, we extend the original DPF framework to incorporate LiDAR observations alongside vision data, enabling more robust state estimation. A traditional particle filter serves as both a training reference and evaluation baseline, highlighting DPFs' ability to capture multi-modal, non-Gaussian uncertainty. Despite encouraging localization performance, we identify sensor data processing, particularly image handling, as a major runtime bottleneck. Our findings emphasize the importance of optimizing sensor fusion strategies and leveraging onboard GPU acceleration for efficient real-world deployment. Future work will focus on improving ground truth quality via motion capture integration and enhancing inference speed through hardware optimizations. The source code is available at: https://github.com/Sai503/differentiable-particle-filters The project page is available at: https://sites.google.com/umich.edu/differentiable-particle-filter/projects.**

I. Introduction

Accurate state estimation is essential for autonomous robotics, enabling agents to localize themselves and navigate complex environments based on noisy and incomplete sensor inputs. While classical filtering approaches such as the Extended Kalman Filter and Particle Filter have provided reliable solutions for decades, these methods often depend on hand-crafted motion and sensor models that can be brittle in real-world conditions.

Recent research has introduced Differentiable Particle Filters (DPFs), a promising hybrid between classical particle filtering and end-to-end trainable neural networks. DPFs retain the algorithmic structure of traditional particle filters while learning motion and measurement models directly from data, allowing them to capture multimodal and non-Gaussian uncertainties critical for robust state estimation.

However, despite their potential, prior evaluations of DPFs have largely been limited to simulation environments. Their real-world viability, particularly on resource-constrained robotic platforms, remains underexplored.

In this work, we extend the DPF framework to a real-world embedded setting using the MBot platform. The MBot platform is a differentiable drive mobile robot equipped with a 2D LiDAR, RGB camera, and inertial measurement unit (IMU) used for teaching robotics at the University of Michigan [1]. Our contributions are threefold:

- We design and implement a multimodal observation encoder that fuses visual and LiDAR data, providing richer input features to the DPF.
- We create a custom real-world dataset by collecting synchronized RGB images, LiDAR scans, and ground-truth pose estimates from the MBot's onboard particle filter.
- We evaluate the DPF's localization performance and runtime efficiency on the MBot platform, benchmarking against baseline odometry and analyzing its feasibility for real-time deployment.

Our experiments highlight the benefits and challenges of deploying DPFs in practical robotics systems, and provide insights into future optimizations for embedded state estimation.

II. Related Work

Traditional state estimation techniques in robotics have largely been dominated by Kalman Filters (KFs) and their extensions, such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). These methods assume unimodal, often Gaussian distributions over the system's state and typically require hand-tuned, model-based representations of motion and observation processes. While effective under mild nonlinearities and modest noise, KFs can struggle in highly nonlinear, non-Gaussian environments often encountered in real-world robotics [2].

Particle Filters (PFs) address these limitations by representing the belief state as a set of weighted samples, or particles, allowing them to approximate arbitrary, multimodal distributions [3]. In a PF, the prediction step samples from a motion model to propagate the particles, while the measurement update adjusts particle weights based on observation likelihoods. This flexibility enables PFs to operate effectively in complex environments where parametric assumptions break

down. However, traditional PFs still rely on manually designed models for motion and sensing, making them susceptible to poor performance if the underlying models are inaccurate or poorly tuned. Additionally, the resampling process, necessary to prevent particle degeneracy, can introduce additional computational overhead and sample impoverishment.

To overcome these challenges, Jonschkowski et al. introduced Differentiable Particle Filters (DPFs) [4], which replace hand-crafted motion and measurement models with end-to-end learnable neural networks. DPFs retain the algorithmic structure of particle filtering—critical for interpretability and uncertainty representation—while allowing components such as the motion model, observation encoder, and likelihood estimator to be optimized directly through gradient descent. This shift from hand-tuning to data-driven learning has shown to enhance generalization, especially in complex or previously unseen environments [5].

Building on this foundation, Janne et al. [6] proposed an early sensor-fusion extension to DPFs, demonstrating that fusing stereo vision and IMU yaw rate measurements at the observation level could significantly improve trajectory tracking performance on the KITTI Visual Odometry dataset. Their results highlight the potential of multi-modal learning within differentiable filtering frameworks, motivating further exploration of integrating additional sensing modalities such as LiDAR.

In parallel, Peñarroya et al. [7] demonstrated that convolutional neural networks (CNNs) can be effectively employed to process raw LiDAR range data for rapid and efficient scene understanding. Their findings show that lightweight 1D CNNs are capable of extracting meaningful geometric features from sequential LiDAR scans, even under tight computational constraints. Drawing inspiration from this, we incorporated a dedicated 1D CNN LiDAR encoder into our observation pipeline, enabling richer environmental perception when fused with vision data.

To test these extensions in real-world settings, we utilized the MBot platform, a compact robotic system equipped with an RGB camera, a 2D LiDAR, and an inertial measurement unit (IMU) [8]. The MBot's onboard particle filter provides a reliable ground truth pose estimate, allowing us to create a supervised dataset without the need for external motion capture systems. Its open-source architecture and computational constraints make it an ideal testbed for evaluating the feasibility of deploying DPFs on embedded systems [1].

Together, these developments underscore the growing emphasis on replacing hand-crafted models with learned, data-driven components while maintaining the benefits of probabilistic state estimation. Our work builds on these insights to deploy a real-world visual-lidar DPF capable of running on low-compute hardware, bridging the gap between simulation successes and embedded robotic deployment.

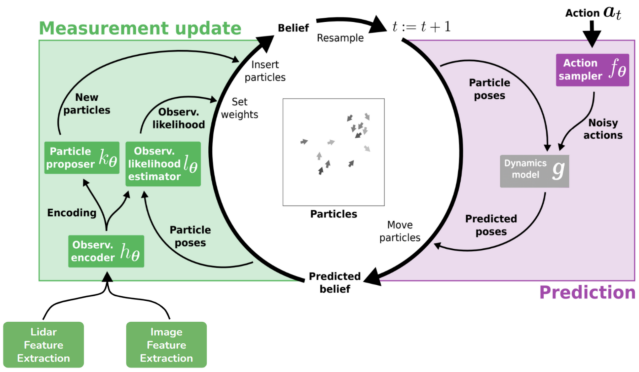

Fig. 1: This diagram describes the algorithmic prior that our model was based on. The gray box represents models that weren't learnable (as in the dynamics model was known prior), and the rest of the models were learned allowing for a diverse set of particles to be proposed and propagated.

III. Paper Reproduction

Our original Differentiable Particle Filter (DPF) model was based on the implementation outlined in [4], which provided TensorFlow 1 code for an end-to-end differentiable particle filter. However, TensorFlow 1 is now deprecated and less efficient compared to modern machine learning frameworks. We therefore ported the codebase to PyTorch to improve compatibility and operational efficiency.

During the porting process, we identified several inefficiencies in the original implementation, particularly in the manual computation of statistical quantities such as means and standard deviations using explicit for-loops. We addressed these inefficiencies by replacing the manual operations with PyTorch's built-in vectorized functions, resulting in significant improvements in run-time performance and scalability.

IV. Algorithmic Extension

To adapt the Differentiable Particle Filter (DPF) architecture for real-world visual-lidar localization on the MBot platform, we introduced several key extensions targeting the observation encoding process and the particle proposing process.

A. Sensor Embedding

The original DPF framework was designed for vision-only input. To enhance robustness, particularly in environments where visual features may be sparse or ambiguous, we extended the observation encoder to jointly process LiDAR and camera data. Specifically, we developed two separate feature extraction networks: a one-dimensional convolutional neural network (1D CNN) for LiDAR range data and a two-dimensional convolutional neural network (2D CNN) for RGB camera images. The outputs of these networks were concatenated to form a unified observation embedding, which was passed into the DPF modules, including the particle proposer (k_θ) and observation likelihood estimator (l_θ), as depicted in Figure

1. This multimodal encoding provided richer, complementary information for particle proposal and measurement updates.

TABLE I: Lidar Encoder Architecture

Layer Type	Input Channels	Output Channels	Details
Conv1D	2	32	Kernel size = 11, Stride = 1
BatchNorm1D	32	32	-
ReLU	-	-	-
Conv1D	32	64	Kernel size = 5, Stride = 1
BatchNorm1D	64	64	-
ReLU	-	-	-
MaxPool1D	-	-	Kernel size = 2, Stride = 2
Conv1D	64	256	Kernel size = 5, Stride = 1
BatchNorm1D	256	256	-
ReLU	-	-	-
MaxPool1D	-	-	Kernel size = 2, Stride = 2
Flatten	-	-	-
Linear	21504	1024	-
BatchNorm1D	1024	1024	-
ReLU	-	-	-
Linear	1024	256	-
BatchNorm1D	256	256	-
ReLU	-	-	-
Dropout	-	-	-
Linear	256	128	-

TABLE II: Image Encoder Architecture

Layer Type	Input Channels	Output Channels	Details
Conv2D	3	16	Kernel size = 3, Stride = 2, Padding = 1
ReLU	-	-	-
Conv2D	16	32	Kernel size = 3, Stride = 2, Padding = 1
ReLU	-	-	-
Conv2D	32	64	Kernel size = 3, Stride = 2, Padding = 1
ReLU	-	-	-
Flatten	-	-	-
Dropout	-	-	-
Linear	1024	128	-
ReLU	-	-	-

The **LiDAR encoder** consisted of a 1d conv-net followed by a 3-layer MLP to generate an embedding as show in Table I. The **Image encoder** retained the original structure used in the original DPF paper as shown in Table II. This network design ensured that both modalities were embedded into low-dimensional, information-rich representations suitable for downstream particle proposal and measurement likelihood estimation.

B. Temporal Context for Observation Encoding

Initial experiments using all available observations as input to the DPF revealed significant instability over long trajectories. In particular, we observed that early predictions remained accurate, but the model's belief degraded progressively as sequences continued, leading to substantial localization drift. We hypothesized that the lack of temporal context prevented the DPF from correctly modeling dynamic state transitions that require memory over multiple time steps.

To address this, we modified the observation encoder (h_θ) to consume a sequence of only the four most recent observations at each time step rather than all available observations. This is in contrast to the experiments conducted by Jonschkowski et al. [4] where they used sequences of 20 observations. However, because of the Mbot's limited sensing and compute resources we ran our implementation at a relatively low frequency meaning that the total time interval represented by the 4 observations is likely similar to if not greater than the interval represented by the 20 observations from the original paper.

Empirically, this adjustment led to a notable reduction in accumulated drift and improved the stability of the belief state, particularly in challenging environments with perceptual aliasing or sparse landmarks. By leveraging short histories of sensor data, the DPF achieved more consistent and accurate localization without requiring recurrent architectures or external memory modules. This simplification has a secondary benefit of reducing computational requirements, reducing the time required to perform each iteration.

C. Fixed offsets for θ

In early experiments we found that the DPF position estimate would experience significant drift relative to the slam and odometry estimates. We discovered that this was due to the Particle proposer (k_θ) consistently proposing particles with an average yaw of 1 radian rather than 0. We didn't investigate the source of this error which we hypothesize could either come from poor training of k_θ or a bad translation of k_θ from the original paper. Instead we simply applied a fixed offset by calculating $\bar{\theta}$ - the mean proposed theta from the first iteration of the proposer, then subtracting $\bar{\theta}$ from all future proposed particles (including the first iteration).

V. Experiments and Results

To evaluate the effectiveness of our Differentiable Particle Filter (DPF) architecture with visual-lidar sensor fusion, we conducted a series of localization experiments using the MBot robotic platform. Our experiments were designed to assess both qualitative trajectory tracking performance and quantitative localization accuracy under real-world conditions. In this section, we describe the experimental setup and present the results comparing DPF-based localization against baseline odometry methods.

A. Experimental Setup

To evaluate the performance of our Differentiable Particle Filter (DPF) with visual-lidar sensor fusion, we designed a series of localization experiments using the MBot robotic platform. Specifically, we constructed a set of indoor maze environments, varying in layout complexity, to serve as test

arenas. These environments were chosen to include a combination of open spaces and narrow corridors to challenge both perception and motion estimation capabilities.

Each MBot was equipped with an RGB camera and a LiDAR sensor, providing the multimodal inputs to the DPF system. During each trial, the MBot was manually driven through the maze while recording sequences of observations, actions, and estimated poses.

To establish a ground truth reference, we used the MBot's built-in particle filter, which fuses sensor data using traditional probabilistic methods, to approximate the robot's true trajectory. Although this method is not perfect, it provides a reasonable benchmark for localization performance in the absence of external motion capture systems.

1) Dataset Collection: To train and evaluate our Differentiable Particle Filter (DPF) model, we created a custom dataset by manually driving the MBot around each maze environment. During data collection, we continuously recorded sensor observations and corresponding ground truth poses.

At each time step, we capture:

- An RGB camera image.
- A LiDAR scan.
- The MBot's estimated pose obtained via its onboard traditional particle filter.
- The Mbot's odometry estimate

The particle filter estimates served as a proxy for ground truth, providing supervisory signals for training the DPF's observation likelihood estimator and particle proposer networks. Each sample in the dataset thus consisted of a fused observation (camera image and LiDAR scan) paired with a ground truth pose and an action (derived from the difference between odometry measurements), enabling supervised learning of the DPF modules. Our model's structure requires a constant size of 360 (for every degree) for the LiDAR scan data, however the built-in LiDAR scanner on the Mbot was not consistent in its scan rate, resulting in mismatched sizes for each individual scan. In order to rectify this, our data loader iterates through each scan and for each degree missing a ray, we insert an average of the previous and subsequent ray.

2) Model Training: We trained the model using a multistage approach in which each individual module (Image Encoding, Lidar encoding, h_θ, k_θ, h_θ, l_θ, and f_θ) is trained independently followed by an end to end training stage as described in [4]. The model was trained on 287 observations collected on 9 individual paths. Then these were resampled into batches of size 32 consisting of observation sequences of length 10. Utilizing the same hyperparameters as the original paper, we trained the model on an RTX 4090 in under 1 hour.

3) Performance Benchmarking: In addition to evaluating localization accuracy, we measured the runtime performance of the DPF on the MBot's onboard processor. During each experiment, we recorded the inference time required to generate particle proposals and perform the measurement update

Fig. 2: Maze used to evaluate dpf

Fig. 3: Sample Path 1 with single turn

at each time step. These measurements allowed us to benchmark whether the DPF was capable of producing localization estimates at a rate compatible with real-time operation requirements. The runtime analysis also informed design trade-offs between model complexity and deployability on low-compute robotic platforms.

B. Results

For each experimental run, we simultaneously tracked three localization outputs: The MBot native SLAM estimate, the native odometry estimate, and our DPF pose estimate. The MBot driven with teleoperation inside a maze seen in Figure 2.

In the result visualizations, the DPF localization outputs are plotted in blue and the odometry trajectories are plotted in red, overlaid against the SLAM path (treated as ground truth) plotted in green. This visualization enables a qualitative comparison of trajectory tracking performance between the DPF and the baseline odometry method. These results are depicted in Figures 3, 4, and 5 which visually indicate strong localization performance from our DPF when compared to traditional SLAM.

Quantitative evaluation is performed by measuring the localization error between each method's predicted pose and the

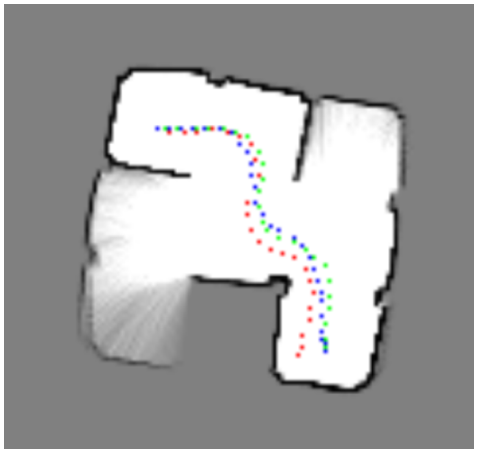

Fig. 4: Sample Path 2 with multiple turns

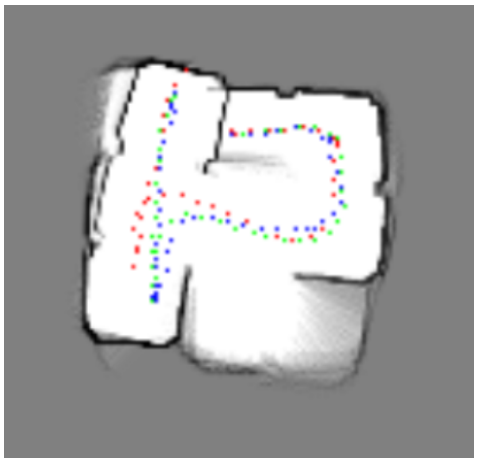

Fig. 5: Sample Path 3 with backtracking

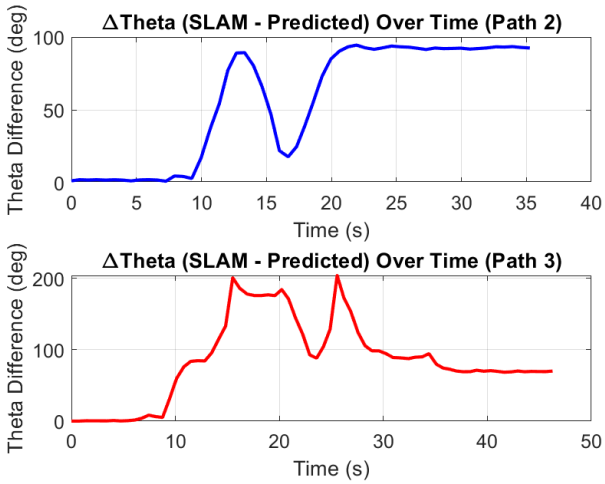

Fig. 6: Predicted Theta Error Over Time

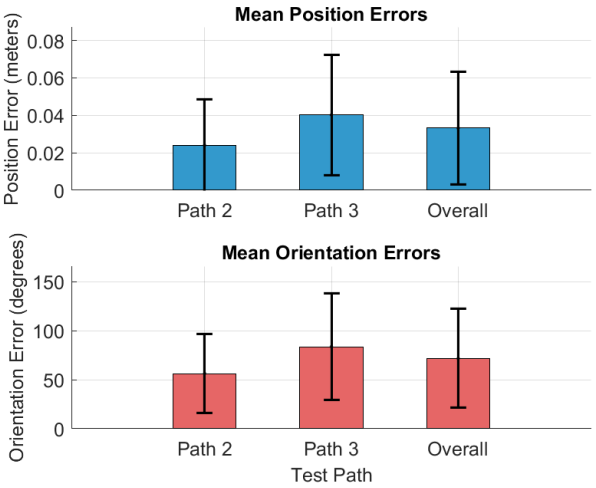

Fig. 7: Averaged Theta and Position Error

ground truth pose over time.

1) Accuracy Evaluation: To evaluate the position and orientation estimation accuracy, we collected SLAM poses and predicted poses from inference while manually driving the Mbot through a maze environment. The average position and orientation errors were computed using SLAM pose as a reference. As shown in Fig. 7, the mean position error was approximately 0.03 meters, indicating highly accurate position updates. In contrast, the mean orientation (θ) error was approximately 70°.

To better understand the high orientation error, we analyzed the θ error evolution over time for each maze, as shown in Fig. 6. We observed that orientation errors initially remained low but grew over time, particularly during turns. This trend suggests that accumulated heading error during rotations is a significant contributor to the overall orientation drift. Interestingly, despite large orientation errors, the positional estimates remained accurate, implying that the Mbot's localization was relatively robust to heading drift in the maze environment.

It is important to note that all accuracy measurements were based on the SLAM pose, which may itself contain drift and therefore does not constitute a true ground truth. Future work will focus on using a high-precision motion tracking system to obtain accurate ground truth data for both training and evaluation purposes.

2) Runtime Evaluation: We evaluated the runtime performance of the Differentiable Particle Filter (DPF) on the MBot platform by measuring the computation time per filter iteration across varying numbers of particles. For our primary experiments, we operated with 100 particles, which we determined to offer a balance between localization accuracy and computational efficiency.

Under this configuration, each DPF iteration required approximately 0.16 seconds, of which 0.03 seconds were spent on sensing operations (capturing and preprocessing camera and LiDAR data) and 0.13 seconds were attributed to inference (particle proposal, update, and resampling). This runtime represents a significant improvement over earlier measurements.

The speedup is primarily due to three factors:

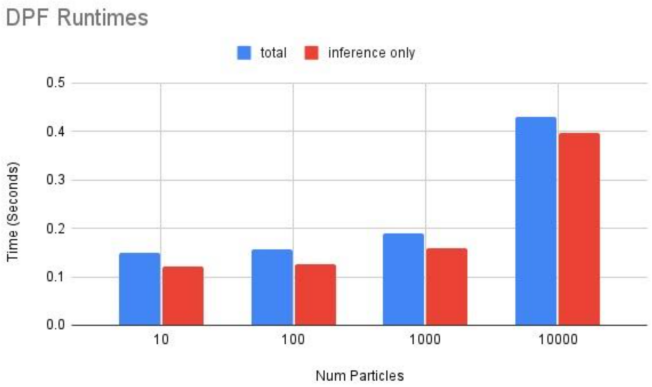

Fig. 8: DPF runtimes as a function of the number of particles. "Total" time includes sensing and inference; "Inference only" isolates the computational cost of the DPF.

A summary of DPF runtime performance across different particle counts is shown in Fig. 8, demonstrating the expected scaling behavior as the number of particles increases.

VI. Conclusions & Future Works

Throughout this project, we were able to achieve highly accurate results in the controlled maze environment used for data collection and testing. However, several limitations in our methodology could be addressed in future work. In particular, the limited dataset size and the absence of accurate ground truth data prevented a fully rigorous evaluation of our model's performance and may have contributed to overfitting to the maze-specific data.

Despite these challenges, we demonstrated that our Differentiable Particle Filter (DPF) framework successfully achieves accurate inference on a low-compute CPU, aligning with our original project goals. Future work will focus on expanding the dataset to include more diverse environments, incorporating motion capture systems for precise ground truth measurements, and further analyzing the causes of orientation error accumulation to improve robustness across different navigation scenarios.

References

[1] P. Gaskell, J. Pavlasek, T. Gao, A. Narula, S. Lewis, and O. C. Jenkins, "Mbot: A modular ecosystem for scalable robotics education," 2023. [Online]. Available: https://arxiv.org/abs/2312.00962

[2] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, 2015, pp. 74–77.

[3] H. R. Künsch, "Particle filters," *Bernoulli*, vol. 19, no. 4, Sep. 2013. [Online]. Available: http://dx.doi.org/10.3150/12-BEJSP07

[4] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," 2018. [Online]. Available: https://arxiv.org/abs/1805.11122

[5] J. B. Alina Kloss, Georg Martius, "How to train your differentiable filter," Jul. 2020.

[6] N. Janne, C. Elwell, and M. Badgett, "Differentiable particle filter reproduction & early sensor-fusion extension: Visual gyrodometry," Robotics Department, University of Michigan, Ann Arbor, Term Project Report, Apr. 2024, project report, 29 April 2024. Reproduction and extension code available at https://github.com/celwell20/deep_rob_term_project_w24 and https://github.com/njanne19/kittifilter.

[7] B. Kaleci, K. Turgut, and H. Dutagaci, "2dlasernet: A deep learning architecture on 2d laser scans for semantic classification of mobile robot locations," *Engineering Science and Technology, an International Journal*, vol. 28, p. 101027, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098621001397

[8] The MBot Ecosystem, "The MBot ecosystem: A low-cost, adaptable ecosystem for teaching robotics with real robots," https://mbot.robotics.umich.edu/, 2025, accessed: 28 April 2025.