Learnable Spatio-Temporal Map Embeddings for Deep Inertial Localization

Dennis Melamed

CMU-RI-TR-21-22

June 25, 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Prof. Kris Kitani, *chair*
Prof. Michael Kaess
Sudharshan Suresh

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Pedestrian localization systems often fuse inertial odometry with map information via hand-defined methods to reduce odometry drift, but such methods are sensitive to noise and struggle to generalize across odometry sources. To address the robustness problem in map utilization, we propose a system that forms a data-driven prior on possible user locations in a map by combining learned map and learned odometry embeddings. Our prior learns to encode which map regions are feasible locations for a user more accurately than previous hand-defined methods. This prior leads to a 49% improvement in inertial-only localization accuracy when used in a particle filter, approaching the performance of bluetooth beacon-based absolute positioning. To show the generalizability of our method, we also show similar improvements using wheel encoder odometry instead of inertial odometry.

Acknowledgments

I'd first like to thank my advisor, Professor Kris Kitani, for being an excellent mentor throughout my time at CMU and for being so giving with his time to help me grow as a researcher. Without his insightful advice, guidance, and support, this work would not have been possible.

I am also extremely grateful to all the members of KLab, both past and present. Their willingness to help with experiments and provide a sounding board for ideas was indispensable. I'd like to particularly thank Scott Sun for all his work getting us to the point where this project was possible, and Vivek Roy for his help in key components of this project. I would also like to thank my thesis committee: Kris Kitani, Michael Kaess, and Sudharshan Suresh for their generous offers of time and feedback on my work.

Finally, I would like to thank my family and friends for their constant support and encouragement as I made my way through graduate school. A special thanks to my partner Melanie for all her help, level-headed advice in tough situations, and patience during these two years. Without any of these people I would not have been able to complete this work, and I am grateful beyond words.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Existing approaches to using occupancy map information to improve odometry-only localization are hand-defined, leading to sensitivity to odometry errors and implicit assumptions which do not always generalize across odometry sources. Methods like SLAM systems [30], map-matching approaches [18, 27], and heuristic map constraints on motion [12, 26] show that maps are a rich information source to improve localization, but new methods are needed to more robustly utilize map information. We seek to extract more robust map information via a learned model to improve localization in odometry-only scenarios, particularly for pedestrian inertial localization.

Inertial odometry, or the estimation of a human user's location from a worn inertial measurement unit (IMU), has seen significant advances in accuracy from various deep methods [9, 22, 28] but still suffers from drift due to integration of small errors in relative motion estimates. This is unavoidable in the real world without introducing absolute constraints on user location, like those provided by a map. Introducing occupancy map information allows for drift correction without

requiring additional sensors to provide absolute constraints, like cameras, LiDARs, or fingerprinting systems.

Existing indoor localization systems using vision, LiDAR, and fingerprinting suffer from downsides that odometry-only methods avoid. Image or LiDAR-based localization via Simultaneous Localization and Mapping (SLAM) can be highly accurate [29, 30], but can require well-lit and highly featured environments, be power-hungry, and be constraining for human activities since these sensors always have to be able to see the world. WiFi and Bluetooth fingerprinting systems require instrumentation of environments, which can be costly and require significant upkeep [2, 16]. The deep inertial odometry methods proposed in RoNIN [28], TLIO [9], and IDOL [22] avoid the above drawbacks by making use of IMUs. IMUs are lightweight, use only milliAmps of current, do not require line-of-sight, and are present in most modern smartphones and robots.

Map information has been utilized to reduce inertial odometry drift [12, 26, 27], but prior methods are hand-defined and make assumptions which generalize poorly, like discretizations of the map for human localization which are too coarse for slower-moving robots, or use heuristics on location feasibility. In our experimentation, these heuristics often lead to localization errors. To reduce odometry drift without hand-defined methods, our approach incorporates map information via an efficient learnable map prior. Our deep network uses a convolutional model to encode map environmental information and a recurrent model to encode recent odometry measurements. These encodings are combined into a map prior to determine areas of the map where the recent odometry measurements would have been feasible with higher accuracy than prior methods. When used as the sensor model of a simple particle filter, our prior is able to utilize highly noisy inertial odometry to provide approximately 49%

improvements in localization accuracy over the original odometry. This level of improvement is also seen when our system is applied to wheel encoder odometry for robots without requiring retraining of any system components.

Chapter 2

Related Work

Indoor agent localization has been explored from several angles including pure odometry approaches, combining maps with odometry, and utilizing other sensors combined with odometry.

2.1 Pure Odometry Approaches

Inertial odometry for pedestrians has used both traditional and data-driven learning approaches. Traditional approaches like Pedestrian Dead Reckoning (PDR) generally begin by estimating device heading via filters like Magwick's [13] or closed source estimators like the iOS CoreMotion API which utilize gyroscope, accelerometer, and magentometer signals. Step-detection is then usually performed to find where in the IMU signals the user has taken a step. PDR methods make assumptions about the size of the user's stride and, when a step is detected, add this stride length in the direction of the estimated heading to the last estimated position to update

the estimate [3]. Such approaches experience significant drift, and the stride length assumption in particular is easily broken.

Chen et al. [4] used a deep network to learn to consume IMU measurements and output user polar displacements like PDR but without making assumptions about user stride and other tunable parameters from PDR. By avoiding these hand-set values, Chen et al. [4] was able to achieve significant improvements in inertial odometry through a learned model. RoNIN [28] explored ResNets, Temporal Convolutional Networks, and recurrent networks to determine the best performing architectures for the deep inertial localization task. TLIO [9] added a Kalman filtering element to the ResNet RoNIN architecture to improve accuracy. The Kalman filter combined RoNIN estimates with the original IMU signals to improve heading estimation and achieve even more accurate localization. IDOL [22] reduced inertial heading errors even further by using a learned deep orientation estimation system.

Functioning similarly to IMUs for human localization, wheel encoders are a common source of odometry measurements for robotic platforms, and recent applications of deep networks have provided better uncertainty estimation to reduce encoder odometry drift [14]. Despite improvements, odometry-only approaches still suffer from drift over the long term since they do not have access to any means of correcting accumulating small errors or biases.

2.2 Map & Odometry Fusion

Map information has been introduced into odometry localization via several different techniques. Graph-based systems turn a map into a series of connected nodes which lie in the space the user might travel through. The user's trajectory is then constructed

out of physically feasible transitions between these nodes which best match the user's odometry. Luo et al. [10] implements a hidden Markov model (HMM) to localize agents with unstable GPS signals on a graph of outdoor road segments. GPS signals are matched to road segments, and based on known factors about the maximum travel speed of the user and other constraints, a series of maximally likely transitions are estimated via the Viterbi algorithm. Deep attention networks have been shown to outperform HMMs in some scenarios for matching user travel to road segments [31].

Indoors, the localization problem in a graph becomes slightly more complex without the constraints of a road network. There are fewer limitations on agent motion indoors and smaller distance errors become more significant, meaning that an algorithm has to track in almost full 2D space instead of the more constraining road network space. Conditional random fields (CRFs) have seen high pedestrian localization accuracy by forming a graph consisting of nodes spaced apart by human step lengths, fully covering the physically accessible space [27]. Transitions in this graph are predicted based on inertial odometry. Graph approaches are successful but require tuning the underlying graph for a specific source of odometry or sensor measurements. The state space of these algorithms is discretized by the nodes. The algorithm will never be able to localize more precisely than the node spacing parameter value. For slower-moving systems, a graph spaced for human users means that the slow agent will spend long periods of time at a given node before transitioning to the next, despite moving the entire time. To resolve this, the graph must be regenerated completely to be denser. This is a barrier to generalization to new odometry sources, can lead to rapid graph size growth, and can increase inference time.

Bayesian filtering systems improve localization accuracy by combining information sources like odometry and map information directly without an underlying set of

possible states. Extended Kalman filtering (EKF) is an efficient approach to estimating user state from both odometry and external information [20], but it suffers from only maintaining one estimate of state and assumes Gaussian uncertainty. For the indoor localization task, maintaining multiple estimates of state until one is proven most likely can be very useful, such as when it is uncertain which side of a narrow obstacle the user has traveled on.

Particle filters [8] are often used when state distributions are unknown, change over time, and could be multi-modal. Such filters maintain an estimated state distribution by storing a set of samples, or particles, which discretely represent the distribution. Multiple estimates of location (the particles) are propagated using odometry. A small amount of noise is artificially added to the odometry to simulate the uncertainty in the odometry measurement. Particles are then weighted based on external information, *e.g.* how well their state matches the output from a LiDAR scan. A resampling process then keeps particles with probability proportional to their weight, maintaining a discrete distribution which best combines both odometry and external information. Particle filters are simple and adaptable, lending themselves well to localization on low compute devices.

For particle filters, map constraints are usually invoked by heuristics which heavily downweight the probability of particles whose last odometry propagation is infeasible according to the map. Such a system has been proven useful in practical scenarios [3, 12, 26], but down-weighting particles in this way is extremely sensitive to certain types of odometry noise in our experiments and can lead to particle weight degeneracy. The particle filter presented by Rechy Rormero et al. [18] calculates overlap between map obstacles and a computed trajectory found by integrating a few previous odometry measurements backwards from a particle location. Particles are given more weight if

there are fewer overlaps. In our experiments, this type of multi-step heuristic is less sensitive to odometry noise than a single-step heuristic but still suffers from the high levels of error in inertial odometry. Our work suggests that a deep network may be able to capture more general shape characteristics of the odometry to better match map locations even when odometry error is high.

Peng and Weikersdorfer [17] utilizes a unique method of fusing map information with odometry which tracks a belief tensor containing all possible user states instead of individual particles. States are propagated in much the same way as in a particle filter, and map information is used to downweight the probability of states which enter obstacles. This method does not noticeably outperform a particle filter in their experiments and shares the sensitivity to odometry noise of particle filter methods which use hand-defined map heuristics.

2.3 Localization with Other Sensors

Other sensors are often available indoors and are used for localization. Vision and LiDAR systems like V-LOAM [30] are highly accurate, but they require line-of-sight to the environment and have significant power and computation needs. For a human user who wishes to localize based on their smartphone, these requirements would mean they are not able to place their smartphone in their pocket and must hold it in a somewhat unnatural position for common usage. Bluetooth beacon-based localization allows a user to hold their phone naturally or put it in their pocket. Bluetooth beacons are used to generate a pattern of radio frequency signals which can fingerprint a location fairly precisely [1]. Beacons emit occasional broadcasts which include a unique ID per beacon. By recording signal strength from each ID, the user can be

triangulated in absolute space [23], or a data-driven model can be used to output more accurate position estimates [1]. Palaskar et al. [16] performs a similar operation but with WiFi access points instead of Bluetooth beacons. Fingerprinting systems using Bluetooth or WiFi are even more effective when combined with inertial odometry to smooth the noisy fingerprinting location estimates [2]. Fingerprinting often requires expensive infrastructure installation and maintenance, and building materials can block signal propagation [21]. Utilizing an odometry-only approach resolves many of these problems, particularly if the sensors are low-power and non-line-of-sight like the IMUs and wheel encoders used in our work.

Chapter 3

Methods

3.1 Systems

We aim to develop a system which consumes high drift odometry measurements and uses indoor occupancy map information to reduce drift in the final estimated user location. Map information is integrated via the sensor model of a particle filter, but prior methods of doing so are sensitive to high levels of error in inertial odometry. Instead, we learn a data-driven map prior from odometry and building maps which we hypothesize will indicate likely user locations with higher accuracy than traditional heuristic methods.

3.1.1 Model Architecture

Our Learnable Map Prior network is shown in Figure 3.1. The network aims to generate the likelihood that a given series of odometry measurements ends at a given location $\mathbf{x}$ in the map. The likelihood can be used directly as the particle filter weight

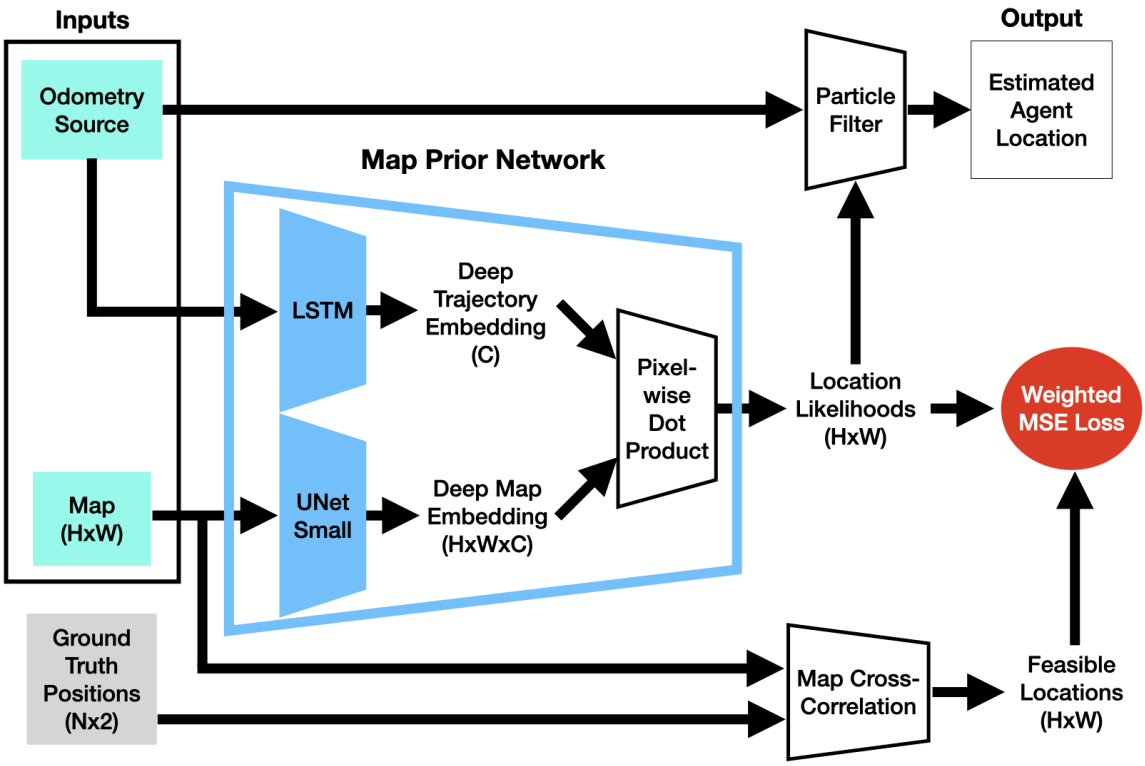


Figure 3.1: System Diagram. N is the amount of odometry history used to calculate the map prior.

for a particle at $\mathbf{x}$. We set up this network with two branches to separate map and odometry processing, whose deep encodings are later combined. This setup is useful because maps are generally static, so at runtime a specific indoor map only needs to be processed by the network once to generate its encoding. The stored map encoding can later be combined with the constantly changing odometry encoding, reducing computational burden. The inputs to our method are a 2D indoor occupancy map M and a short history of the agent's most recent odometry from time $t - N$ to t, O. N is selected experimentally to be 5 seconds for human walking and 20 seconds for slower robot driving. Our map network $f(M, \Theta_f)$ is a small version of U-Net [15, 19] parametrized by weights Θ_f which has been modified to output a "Deep Map

Tensor" of the same spatial dimensions as the input map but with more channels. The number of channels $C = 32$ is chosen experimentally as a good balance between speed of computation and sufficient space to store relevant environmental details. The odometry network $g(O, \Theta_g)$ is a two-layer Long-Short-Term Memory (LSTM) network parametrized by weights Θ_g operating on the time dimension of the odometry history. The LSTM hidden state is of size C, and the last hidden state output is used as the "Deep Trajectory Vector". A dot product is taken between the Deep Map Tensor and the Deep Trajectory Vector at each pixel location $\mathbf{x}$, giving the likelihood of $\mathbf{x}$ given O and M:

$$\mathcal{L}(\mathbf{x}|M, O) = f(M, \Theta_f)_{\mathbf{x}} \cdot g(O, \Theta_g) \tag{3.1}$$

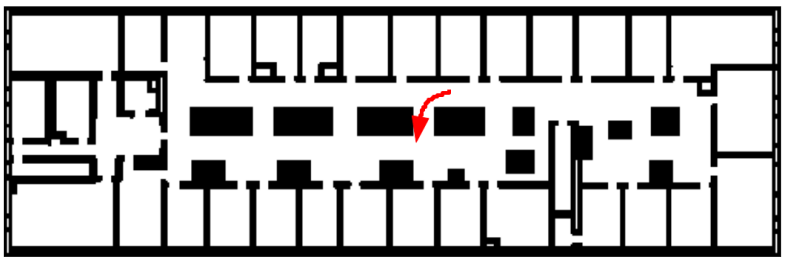
In order to improve learning, the outputs of f and g are normalized to be of 2-norm equal to 1 before this dot product operation is performed. Without this normalization, the network overfits to the most common areas of the map. The norms of the outputs of f and g become very large for the most commonly feasible areas of the map, reducing performance in areas which are less commonly feasible.

3.1.2 Model Training

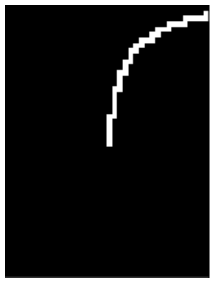
To train our model, we window ground truth position data into windows of N timestamps and subtract the initial position in each window to make this data look like odometry data. We additionally add a velocity bias drawn from a normal distribution with $\sigma = 1$ pixel/second and a random noise drawn from a normal distribution with $\sigma = 0.25$ pixels to each input sample to better mimic noisy odometry data. These windows are used as training input for the odometry branch.

At training time, the map branch of the network does not receive the full map. Instead, the area of the map which includes the ground truth positions in the window being used for the odometry branch plus some additional randomly selected space is given. This has a two-fold purpose. It speeds training, since the entire map does not have to pass through the convolutional layers for each sample. It also theoretically allows the network to generalize to unseen maps, since the network must learn features which are not specific to a given map, but instead encode a general understanding of how obstacles influence motion in different configurations. In this work, the limited size of our datasets does not allow us to test generalization to unseen maps, so at test time a network is used that has been trained with data from the building being tested. For pedestrian inertial localization, the underlying odometry method (IDOL) must currently be trained with data from the specific building regardless, so this does not represent an added data collection burden.

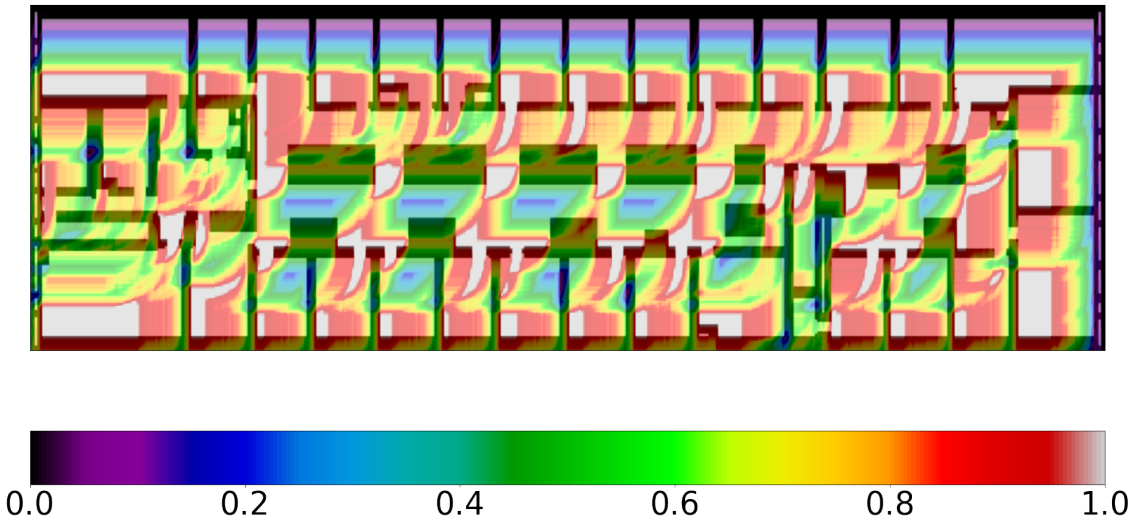
We define a cross-correlation target T (Figure 3.2c) which we train our network to output. A 2D kernel is created with ones where the ground truth "odometry" trajectory is located, as in Figure 3.2b, and zeros elsewhere. The kernel is cross-correlated with the occupancy map to form $\bar{T}$ whose values depend on how much the input trajectory overlaps obstacles. $\bar{T}$ is exponentiated as $T = 10^{-6} * e^{14*\bar{T}}$ to heavily upweight fully free-space trajectory areas and push areas where the trajectory would overlap an obstacle even a little bit almost to zero. We would like our network to output T when given noisy or drifting real odometry as input. We train using Mean Squared Error (MSE) loss: $\mathbf{L} = MSE(T, \mathcal{L}(\boldsymbol{x}|M, O))$. In training our method, we noticed that larger areas of feasible locations in T were being predicted better than small areas. We weight feasible regions by the inverse of their pixel area in our loss to improve small area prediction. Areas where the trajectory is infeasible need to be



(a) Example input ground truth trajectory window (in red). Direction of travel denoted by arrow.



(b) Cross correlation kernel



(c) Resulting training target. Values near 1 are feasible trajectory end locations.

Figure 3.2: Ground truth generation process for training learnable map prior.

weighted by the same amount as the smallest feasible area weight, because where the trajectory is not feasible is equally as useful of a prior as where it is feasible.

3.1.3 Particle Filter

We utilize a particle filter to combine the learned map prior with odometry measurements, as particles filters have relatively low computational burden but are able to handle complex distributions of estimated location. A particle filter maintains

P estimates of the user's state as particles. At each timestep, each particle has its stored state updated using a motion model:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta\mathbf{x} + \mathcal{N}(\mu_{\text{motion}}, \Sigma_{\text{motion}}) \tag{3.2}$$

where $\mathbf{x}_t$ is the user's state (x, y) at timestep t, and $\Delta\mathbf{x}$ is odometry measurements between times t and $t+1$. Noise from the 2D distribution $\mathcal{N}(\mu_{\text{motion}}, \Sigma_{\text{motion}})$ is added to each particle's motion to capture odometry uncertainty.

Due to the significantly higher heading error present in wheel encoder odometry, we extend the state to include robot heading θ for our wheeled robot experiments:

$$s = ||(\Delta x, \Delta y)||_2 \tag{3.3}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + s \begin{bmatrix} cos(\theta) \\ sin(\theta) \end{bmatrix} + \mathcal{N}(\mu_{\text{motion}}\Sigma_{\text{motion}})_{0,1} \tag{3.4}$$

$$\theta_{t+1} = \theta + \Delta\theta + \mathcal{N}(\mu_{\text{motion}}, \Sigma_{\text{motion}})_2 \tag{3.5}$$

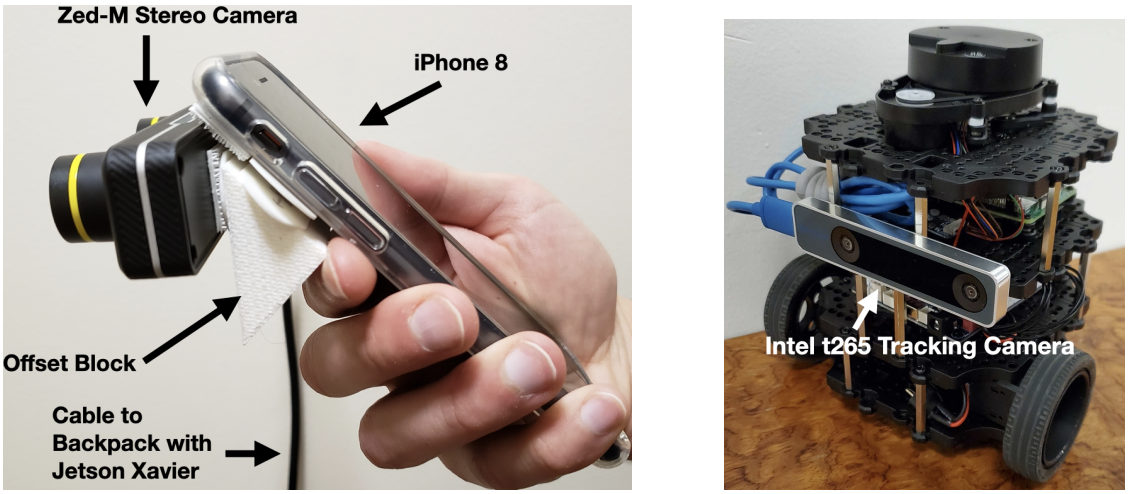
The noise distribution is now 3 dimensional to encode noise in the additional state element θ. Gaussian subscripts are components of the sampled vector and Δ terms are odometry measurements. At test time, encoder odometry is rotated to match the estimated heading before being passed to the map prior network to correct heading drift.

Particle weights are then computed as the value of the Location Likelihood Heatmap at the particle's location using the last N seconds of odometry. Low variance resampling of the particle cloud is used due to its stability and consistent particle space coverage [24]. Resampling copies particles from the original cloud with

probability proportional to their weight. The particle closest to the median of the new particle cloud is used as the estimated location of the user. This guarantees the estimated location is feasible and not inside an obstacle even with a multi-modal particle distribution.

We also implement a re-initialization method for situations where the particle filter has diverged from a reasonable estimate. We reinitialize by sampling particles randomly within a radius r_{reinit} of the last estimated location when more than s_{reinit} of particles have passed through an obstacle. We experimentally set $r_{reinit} = 5$ meters and $s_{reinit} = 75\%$.

3.2 Datasets



(a) BLE+IMU dataset (b) TurtleBot dataset

Figure 3.3: Data collection setups used in this study.

A key feature of our system is its invariance to different sources of odometry data. We test on varied datasets to show the consistent performance of our method.

3.2.1 IDOL Dataset

The IDOL Dataset [22] is an inertial odometry dataset for indoor pedestrian localization. It consists of 20 hours of 10 minute pedestrian trajectories collected by users carrying a smartphone in 3 buildings. Smartphone IMU data and ground truth device pose is available at 100 Hz. Odometry measurements are generated from IMU data by the IDOL method proposed in the same work. 2D occupancy maps of the buildings are extracted from the architectural plans available for the 3 buildings.

3.2.2 BLE+IMU Dataset

In order to compare our method against an exteroceptive measurement for human user tracking, we collect a dataset of bluetooth low energy (BLE) beacon and IMU measurements on 2 different floors in the IDOL dataset building 2. Similar data to the IDOL dataset is also available in this dataset, with the addition of the BLE beacon measurements. Collection areas are instrumented with BLE beacons emitting a unique identifier broadcasts at 1 Hz, which are recorded along with IMU data by a smartphone at 60 Hz. Ground truth device pose and orientation are generated with a Stereolabs Zed Mini stereo camera configured as in Figure 3.3a.

3.2.3 TurtleBot Dataset

A third dataset was collected to test method generalization to new sources of odometry. A ROBOTIS TurtleBot 3 (Figure 3.3b) was driven to collect 8 trajectories of 10 minutes in buildings 1 and 2 from the IDOL dataset. Wheel odometry was recorded from the 2 driven wheels, and ground truth pose was recorded from an attached Intel t265 tracking camera. Odometry data was generated from the wheel encoders and

IMU via a standard differential drive model:

$$\Delta s = r(n_{turnsL} + n_{turnsR})\pi \quad (3.6)$$

$$[x_{t+1}, y_{t+1}, \theta_{t+1}] = [x_t + \Delta s * cos(\Delta\theta + \theta_t), y_t + \Delta s * sin(\Delta\theta + \theta_t), \theta_t + \Delta\theta] \quad (3.7)$$

where n_{turnsL} and n_{turnsR} are the number of revolutions recorded by the wheel encoders since the last time step, and $\Delta\theta$ is calculated via a standard Madgwick filter [13] from onboard IMU data. r is the wheel radius and (x, y, θ) is the robot state.

Chapter 4

Experiments

To test our learned map prior, we directly analyze the prior and utilize the prior as a sensor model for particle filter tracking tasks with various sources of odometry. We evaluate localization performance using two main metrics:

- **Absolute Trajectory Error (ATE)** is the root mean square error of points in the predicted ($\hat{\boldsymbol{x}}$) and ground truth ($\boldsymbol{x}$) trajectories. The error at timestamp i is $E_i = ||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||_2$. ATE measures global consistency and usually increases with time if only odometry is used due to drift.
- **End error (EE)** is the distance between the final estimated and final ground truth positions in a given trajectory, i.e. $E = ||\boldsymbol{x}_T - \hat{\boldsymbol{x}}_T||_2$. EE indicates whether the localization method has been able to correct for drift and localization errors in the long term.

4.1 Training/Testing

We implemented the learnable map prior model in Pytorch Lightning [5] and train with Adam optimization [7] on an Nvidia RTX 2080 Ti using a learning rate of 0.01. A batch size of 32 is used and the network takes between 50 and 100 full passes through the data (epochs) to converge, depending on the map.

At test time, the particle filter initializes with 1000 normally distributed particles ($\sigma = 0.01m$) around the true starting location. The filter runs at 1 Hz. For inertial odometry, the motion model adds Gaussian noise with $\Sigma = 0.1\mathcal{I}_2 m$ and for wheel encoder odometry with $\Sigma = 0.01\mathcal{I}_3 m$. A benefit of our formulation is the decomposition of trajectory and map processing. Once the network is trained, the map only needs to be passed through the network once to obtain the Deep Map Tensor. This can be reused for all trajectories in the same map. At test time, only the trajectory LSTM needs to run. An AMD Threadripper 1920x CPU takes 6.7 ms on average to compute the Location Likelihood Heatmap using a stored Deep Map Tensor. Worst case computation time for our prior is independent of how long the user has been walking, unlike graph-based approaches using the Viterbi algorithm whose computation time grows linearly with the length of the trajectory. Our unoptimized particle filter runs at 4x realtime speed, suggesting that our method would be feasible for smartphone or low-compute platforms.

4.2 Baselines

We compare against several other methods to show the utility of our learned prior. Pedestrian Dead Reckoning (PDR) is implemented similarly to the baseline used in

Yan et al. [28]. A distance traveled and heading are regressed every user step. Step size is assumed to be 0.67 m, and heading is determined using the iOS internal smartphone estimator. IDOL [22] is selected to represent the performance of deep inertial odometry methods. We also compare against our particle filter using a heuristic map prior based on Rechy Rormero et al. [18] representing the best performing of priors used in other works. The heuristic prior is generated by map cross correlation with a kernel based on the noisy odometry trajectory, not the ground truth as for the network training (Section 3.1.2).

For the BLE+IMU dataset, we also compare against absolute localization via BLE beacons. BLE localization is implemented in two ways: (i) pure BLE localization as in Agarwal et al. [1] using ten minutes of fingerprinting data, and (ii) using the BLE estimates as a particle filter sensor model similarly to Ahmetovic et al. [2]. The probability of a particle's state in a $\Sigma = 3$ meters Gaussian with mean at the predicted BLE location is used as its weight in the second case. By comparing against these baselines, we can relate our method's performance using only noisy relative position measurements to the performance of a system which uses significant environmental instrumentation and does not suffer from odometry drift.

4.3 Learned Map Prior Analysis

Figure 4.1 shows several examples of the learnable map prior network output. Spaces where the trajectory window used to generate the prior is feasible are generally given higher weight than extremely unlikely areas. At $T = 300$, the network is also able to handle curving trajectories, placing weight in the bottom right corners of rooms and not nearly as much weight at the top. This matches with the feasible locations for

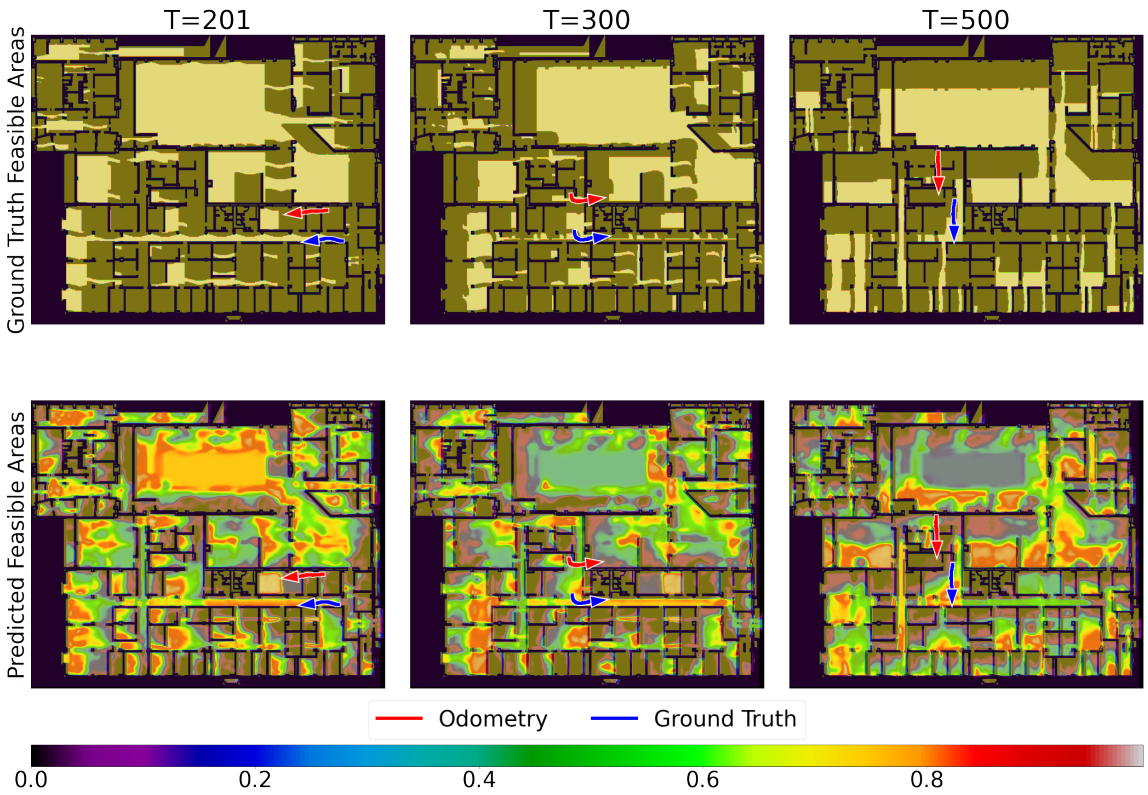


Figure 4.1: Example map prior predictions (generated using red odometry history) below the ground truth feasible locations (generated from the ground truth trajectory in blue).

the trajectory used to generate this prior.

We also compare the KL divergence between a "ground truth" location distribution G and our predicted distribution, and between G and the heuristic prior distribution, shown in Figure 4.2. G is defined as a Gaussian ($\sigma = 1m$) around the ground truth agent location. Our learned prior shows consistently lower divergence from the ground truth than the heuristic prior by a factor of approximately 3. This is because our prior places weight more specifically than the heuristic map prior. The learned prior consistently puts weight in the ground truth location, and it also does not put weight near obstacles as much as the heuristic prior. The heuristic prior simply sees the areas

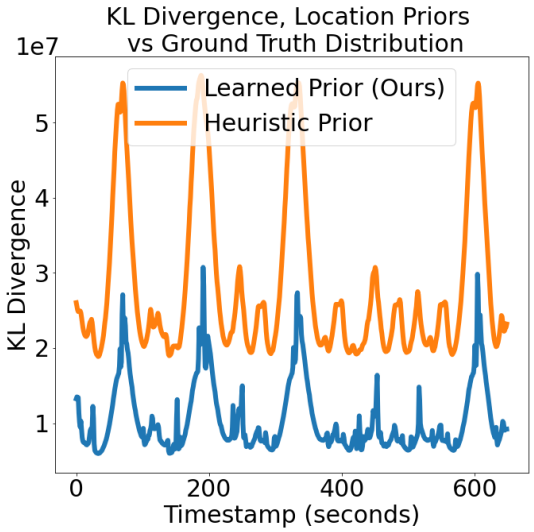
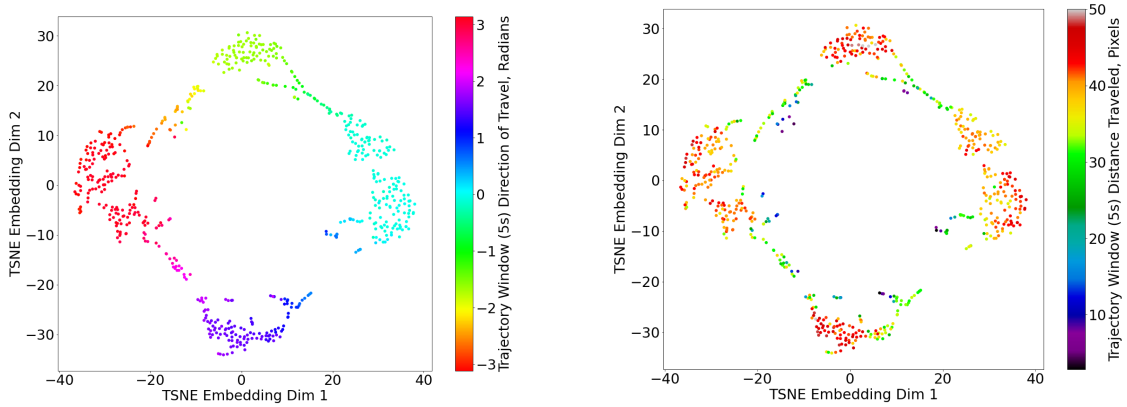


Figure 4.2: KL divergence between ground truth user location distribution and map priors.

very close to obstacles as feasible, and so weight is put there without consideration to how commonly such areas are feasible. The learned prior, on the other hand, is able to draw on its training data to determine whether such areas are normally included in feasible space and correctly discounts them as extremely unlikely.

To check that the odometry branch is learning reasonable embeddings for odometry windows, we perform two comparisons by compressing some example learned embeddings down to two dimensions via t-SNE [11]. t-SNE takes N dimensional vectors and determines an embedding in an $M < N$ dimensional space which aims to preserve relative distances between the original vectors in the embedding space. We select $M = 2$ for visualization.

We first plot the two t-SNE embedding dimensions against each other in Figure 4.3a and color each point by the direction of travel of the input trajectory. The embeddings form a ring, which smoothly changes color as the travel direction of the



(a) Color is the heading direction of the input trajectory.

(b) Color is the distance traveled by the input trajectory

Figure 4.3: 2D t-SNE embedding of the vectors output by the odometry branch of the map prior network.

input odometry changes. Heading direction is key to determining which areas of a map are feasible so, as expected, the network has learned to encode this feature.

Figure 4.3b again plots the 2 t-SNE embedding dimensions, but instead colors the points by the distance traveled by the input trajectory. We can see that longer input odometry trajectories appear to be embedded differently from shorter trajectories. Travel distance is also key to determining whether a given odometry-predicted trajectory will fit into a given map space so, as expected, the network has learned to encode this feature as well.

Our map prior network also learns a deep embedding of the map branch input. This embedding is the same spatial dimension as the input map but with more channels. We expect these channels to encode information about feasible trajectories in that region of the map. Figure 4.4 shows the various channels of the learned embedding for building 2 of the IDOL dataset. Some channels appear to encode areas where a trajectory travelling in a specific direction will be feasible, like channel

22 encoding trajectories moving from right to left. Other channels appear to learn features related to the size of feasible trajectories in that area, like channel 17 which seems to activate only in small rooms. Both direction and size of room are useful to determining feasibility of a given trajectory, so these encodings are as expected.

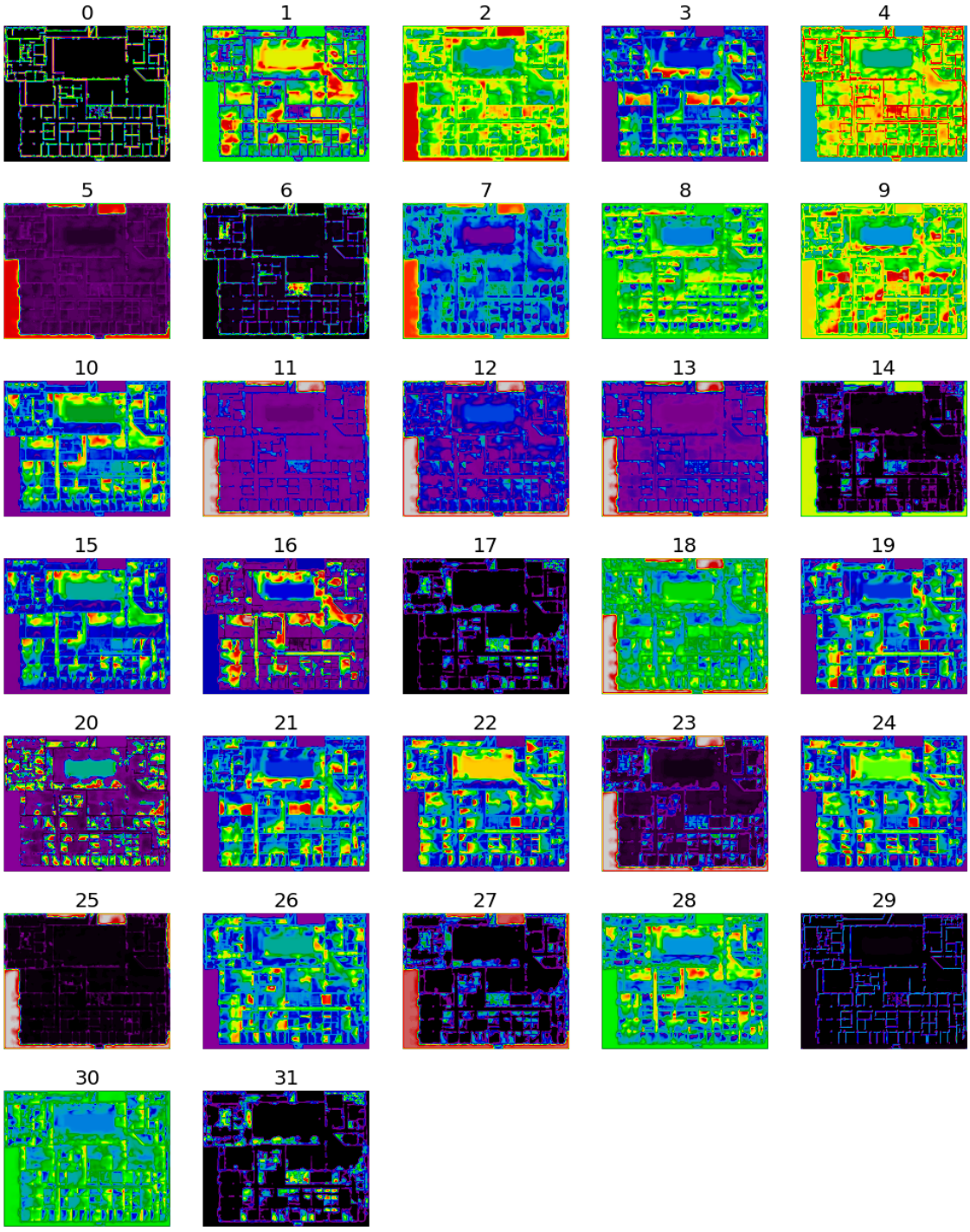


Figure 4.4: Channels of the learned map embedding for building 2.

4.4 IDOL Dataset Experiments

Model	Bldg 1		Bldg 2, Floor 1		Bldg 3	
	ATE	**End Error**	**ATE**	**End Error**	**ATE**	**End Error**
Pedestrian Dead Reckoning (PDR)	24.28	24.96	12.66	13.80	21.86	22.18
Deep Inertial Odometry (IDOL [22])	5.65	8.51	6.62	10.61	8.33	9.71
PF + Heuristic Map Prior	3.11	3.42	11.37	15.36	6.71	**9.67**
PF + Learned Map Prior (OURS)	**2.87**	**1.60**	**2.51**	**6.08**	**5.66**	9.99

Table 4.1: Mean localization errors in different buildings for inertial localization methods, IDOL dataset.

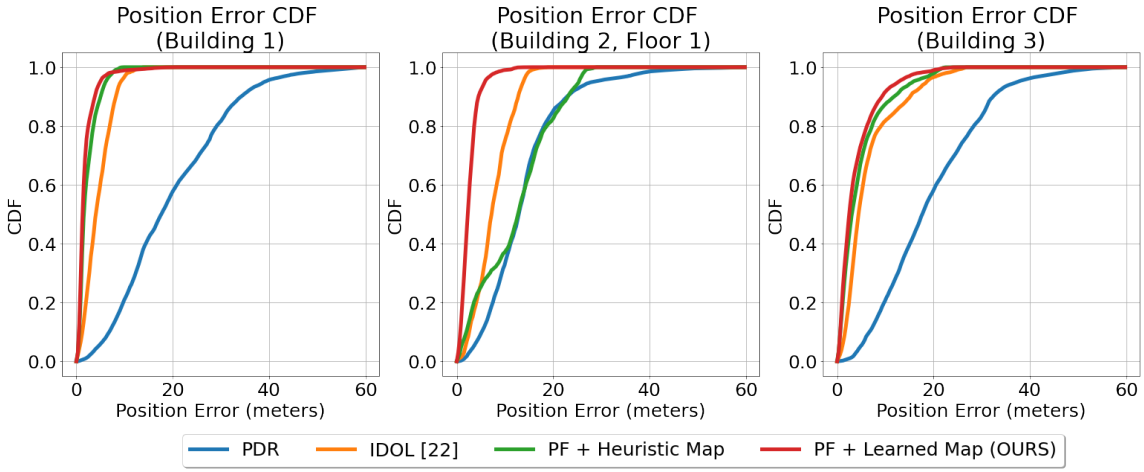


Figure 4.5: CDF of position error on IDOL dataset in 3 different buildings

Table 4.1 presents the results of our localization comparison on the IDOL dataset. Our method generally outperforms others across buildings in both ATE and end error. Buildings 2 and 3 are substantially larger than building 1 leading to significant odometry drift, while our method is able to achieve better results with only the addition of map information. Figure 4.5 shows the cumulative distribution functions (CDFs) of the various methods' errors. Our method estimates consistently more accurate locations and does not exhibit the larger errors skewing the error distribution

of the particle filter with heuristic prior in building 2. Building 2 has a long hallway which pure odometry often drifts out of as users walk down it. Since the rooms by the hallway are large and divided by thin walls, the heuristic prior weights the rooms similarly to the hallway due to only tiny occasional intersections between the walls and the trajectory. Our learned prior learns to be more specific. The hallway is given high likelihood, while the rooms are given low likelihood because there are obstacles (walls) blocking that much travel in the relevant direction. On average, we are able to achieve an error of less than 4 meters 80% of the time using only high drift inertial data and an occupancy map. Figures 4.6, 4.7, and 4.8 show the qualitative improvement in localization accuracy using our method.

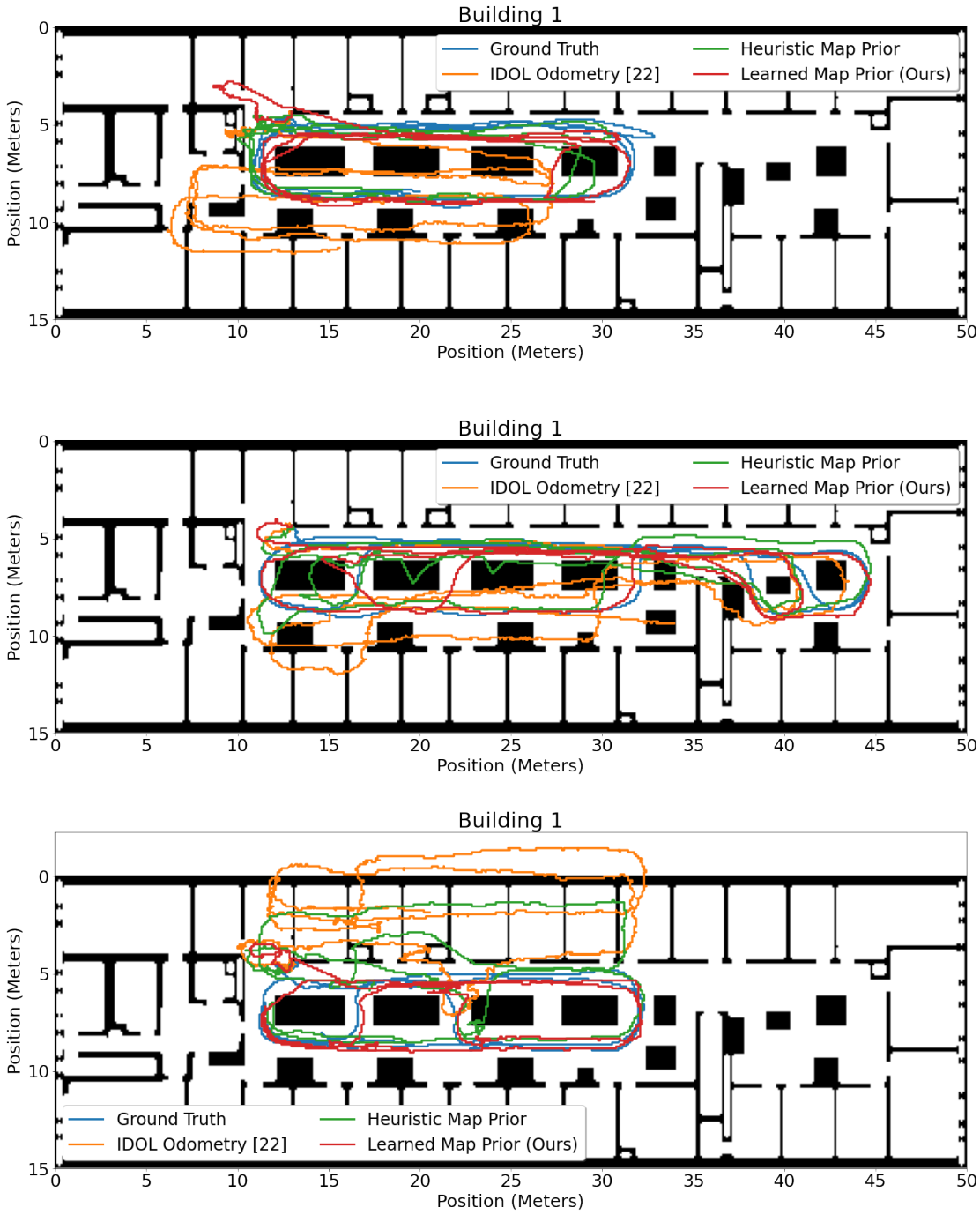


Figure 4.6: Example trajectories in Building 1 comparing IDOL odometry to the particle filter using the heuristic and learned map priors. Truncated slightly for clarity, initial pose is the same across all methods.

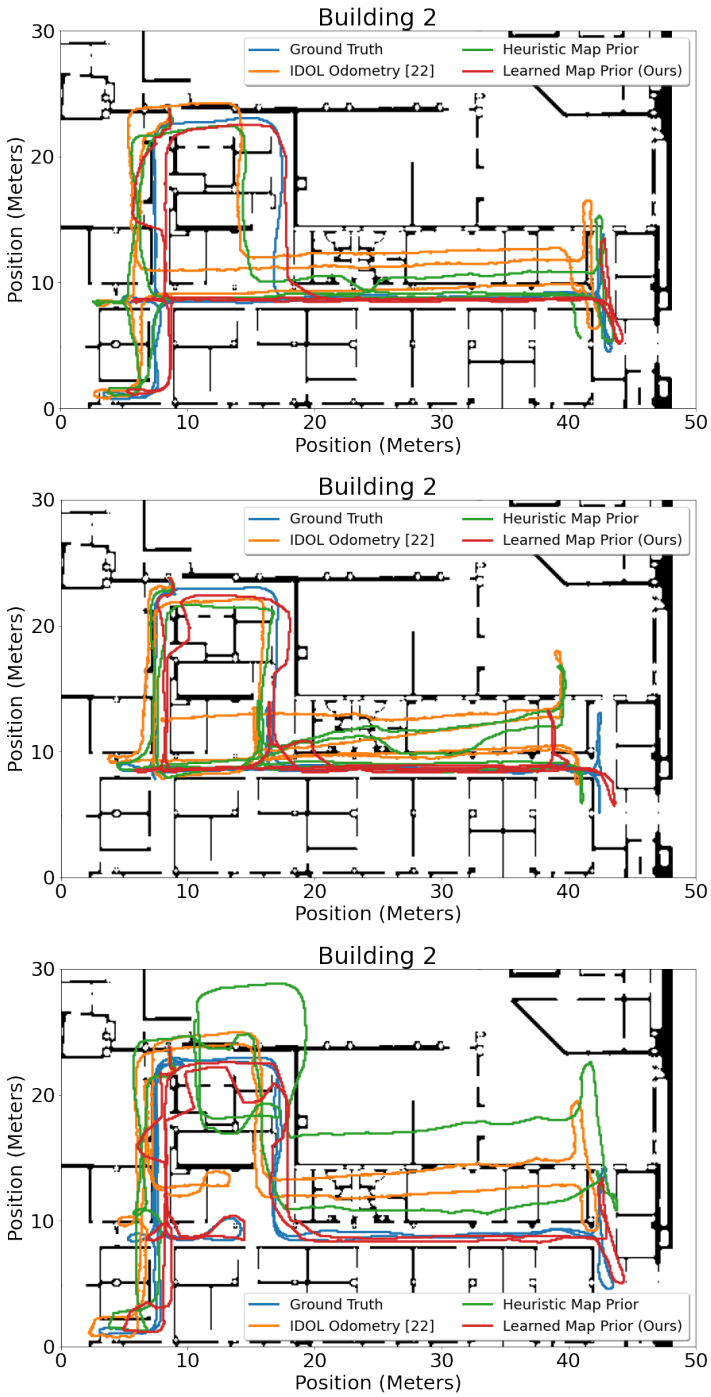


Figure 4.7: Example trajectories in Building 2 comparing IDOL odometry to the particle filter using the heuristic and learned map priors. Truncated slightly for clarity, initial pose is the same across all methods.

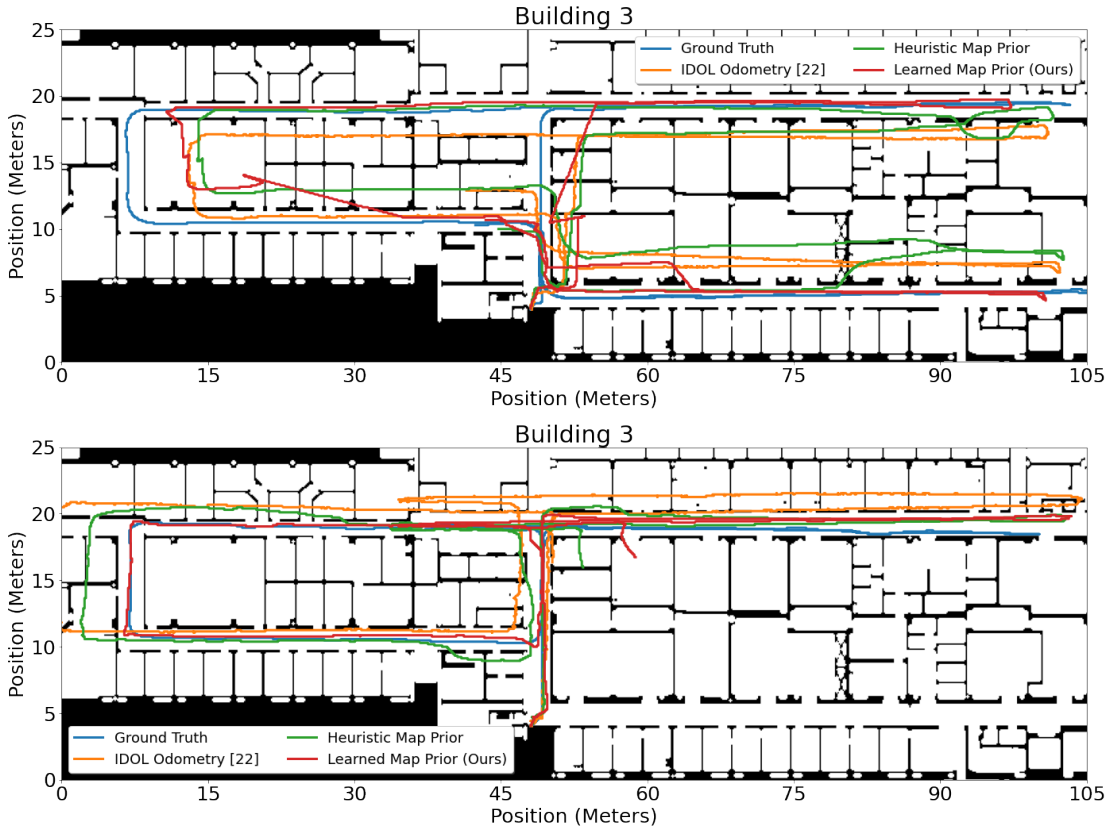


Figure 4.8: Example trajectories in Building 3 comparing IDOL odometry to the particle filter using the heuristic and learned map priors. Truncated slightly for clarity, initial pose is the same across all methods.

4.5 IMU + BLE Experiments

We now aim to compare our method using relative position measurements and a map to a method which directly estimates user pose in a global frame of reference, *i.e.* BLE beacon localization. Figure 4.9 shows the results of using various numbers of beacons in a beacon localization system compared to IDOL and our method. BLE Only uses the raw, 1 Hz BLE estimates of location. The particle filter using the BLE estimates as a prior is able to achieve better performance by smoothing the

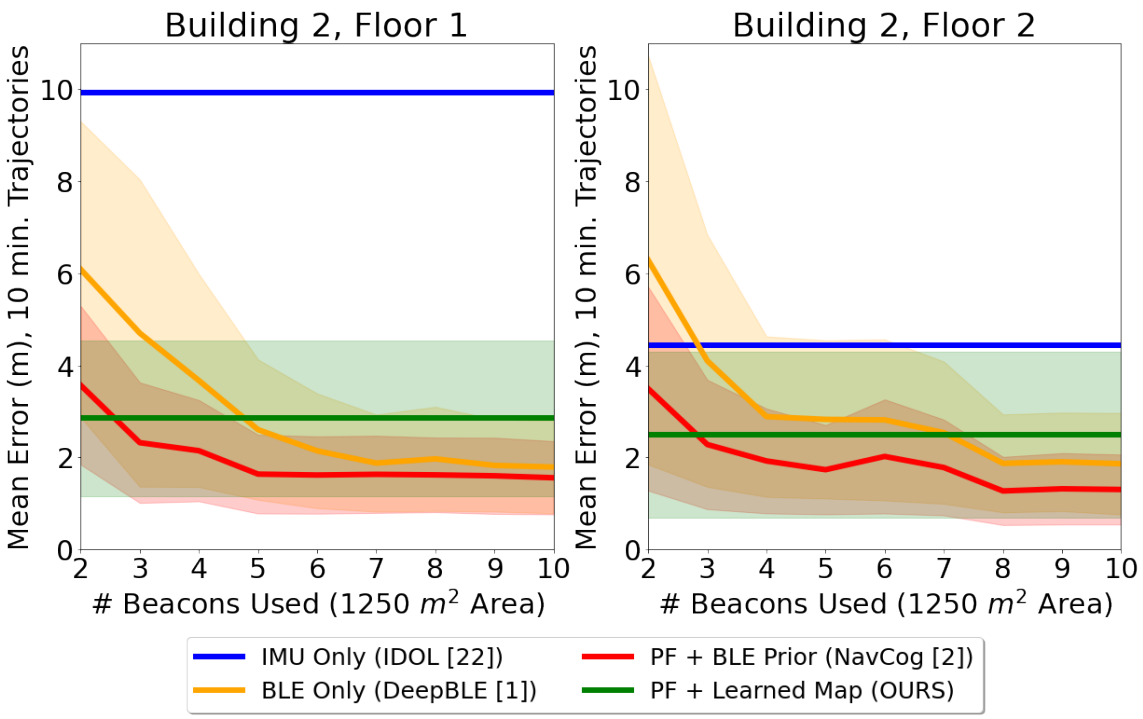


Figure 4.9: Comparison of odometry-based vs. bluetooth beacon-based indoor localization performance for different numbers of bluetooth beacons in the environment. Standard deviation of each method is shown by the shaded region of the relevant color. IDOL standard deviation is not shown due to size. IDOL & our method do not rely on beacons, so have consistent performance as the number of beacons varies.

noisy BLE estimates using inertial odometry. Inertial odometry alone generally does not outperform BLE localization, but with the addition of map information through our learned prior we outperform BLE-only localization with fewer than 5-7 beacons. Overall, we achieve ATE within about 1 meter of both BLE methods using 8-10 beacons. Higher beacon densities, like at 8-10 beacons, make BLE localization very accurate because additional beacons can be used to differentiate between areas which have similar BLE signal characteristics at lower densities. These experiments are performed in a 1250 m^2 area, which is fairly small relative to 10 beacons. Similar performance in large areas could require 100s of beacons which can become infeasible

to install and maintain. Our method is only slightly less accurate and does not require installing anything in the environment.

4.6 Wheeled Robot Odometry Experiments

Model	Bldg 1		Bldg 2, Floor 1	
	ATE	**End Error**	**ATE**	**End Error**
Wheel Odometry Only	4.15	7.84	3.93	3.99
PF + Heuristic Map Prior	3.69	6.25	3.2	2.89
PF + Learned Map Prior (OURS)	**1.88**	**1.69**	**1.48**	**1.48**

Table 4.2: Results of wheeled robot localization using various odometry-only methods across two buildings.

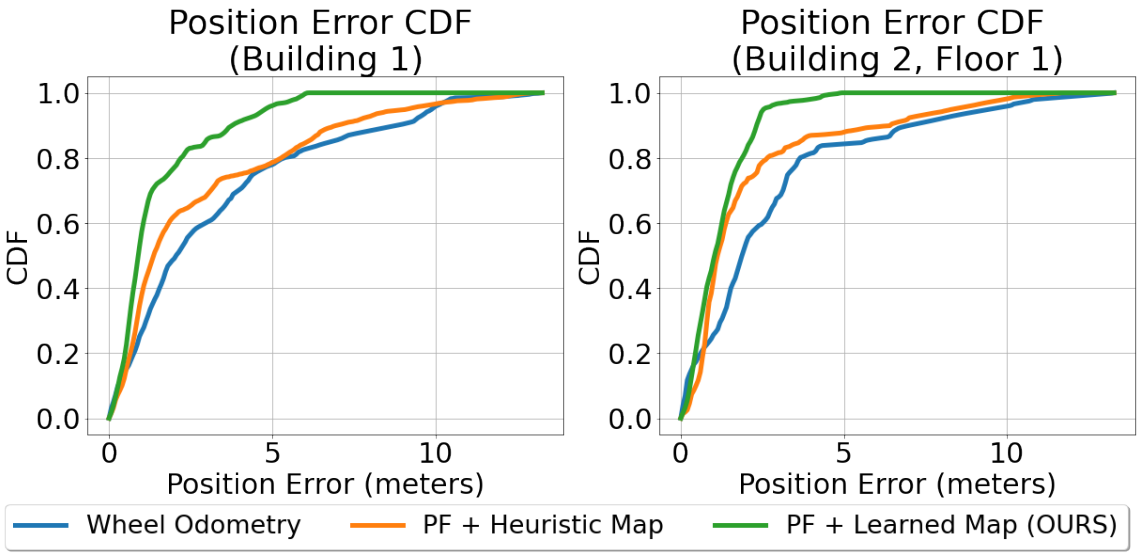


Figure 4.10: Cumulative Distribution Function plots of errors caused by wheel odometry localization methods.

We now show experiments to determine whether our method is able to generalize to a different source of odometry: wheel encoders. Table 4.2 shows mean results for tracking a small wheeled robot in two different buildings using wheel encoder odometry. Our learned prior combined with a particle filter is able to more than halve the error present in the system, which the heuristic prior is unable to handle.

The CDF plots in Figure 4.10 also support the utility of our learned map prior when applied to a particle filter. 90% of errors are less than 2.8 meters using only odometry and map information. These performance levels are achieved without retraining our map prior network. The network is trained on only inertial odometry trajectories and has not seen wheel encoder odometry before.

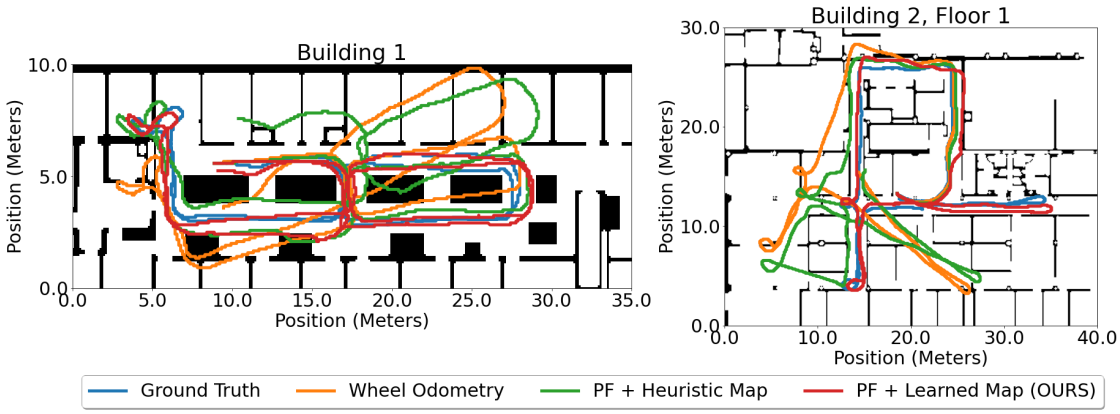


Figure 4.11: Example estimated trajectories of various odometry-only methods on a wheeled robot.

The trajectories in Figure 4.11 show our prior's ability to accurately weight particles. Odometry heading error often accumulates as the robot makes turns, but our learned prior absorbs some of this error and places weight in appropriate locations based on the general curving nature of the trajectory. The heuristic prior is unable to accurately weight regions when heading error occurs and so does not correct the error, leading to significant drift.

Chapter 5

Conclusions

5.1 Limitations

Our learned map prior outperforms and is more versatile than hand-defined methods for fusing inertial odometry and maps but has several limitations. The particle filter in our approach is based on a learned prior, which has an inherent drawback of learning-based methods: performance degradation when generalizing. Within the same map, our method generalized to a variety of trajectories. If some training data is available, this method should generalize to different maps, but specific issues may arise with maps very different to our testing environments. For example, our trials did not explore performance in very large open spaces or trajectories which enter multiple rooms due to access limitations. The network branch which consumes odometry information is another location for possible generalization failure. Our method works well with the noise modalities present in inertial odometry and generalized easily to wheel encoder odometry. However, odometry sources such as those from legged

robots may have very different noise characteristics, and our pre-trained network may be unable to generalize to this new distribution. Specifically for inertial pedestrian localization, our method relies on a learning-based method to generate odometry (IDOL) which has underlying difficulty generalizing to different buildings.

A key requirement of our system is an accurate map of the indoor space with correct transformations between real-world and map coordinates. Architectural plans, like those used in this work, are not necessarily accurate to the built configuration of a space. Indoor mapping systems based on SLAM using cameras or LiDARs have become much easier to use and more accurate, but there is still significant effort required to generate maps if they are not previously available for a given space. Both 2D architectural and SLAM maps do not necessarily capture all obstacles in a map, depending on the height of the sensor used to collect the map. For example, tables are not usually placed in their true locations in architectural plans, and a 2D LiDAR system held by a human operator performing mapping of a space may be held too high or low to accurately capture the tables. Obstacle configurations can also change over time without warning. Since our method relies so heavily on combining obstacle information with user motion to rule out areas where the motion was infeasible, the lack of a full understanding of obstacle locations can limit the effectiveness of our prior.

5.2 Concluding Remarks

In this work, we present a data-driven approach to combining occupancy map information with odometry measurements for indoor localization. Existing work uses hand-defined methods to incorporate map information. These methods make

assumptions which generalize poorly or use noise-sensitive heuristics. Our method provides a more robust prior on indoor user location using a two-branch deep network utilizing separate learned map and odometry embeddings which are combined to determine feasible user locations in the map. Our prior, when used as a sensor model in a particle filter, is able to achieve 49% accuracy improvement over odometry for human inertial localization. Compared to a BLE beacon-based localization system, our method exceeds BLE accuracy in low beacon-density spaces and approaches similar performance to BLE methods in high beacon-density spaces without requiring any devices installed in the environment. Our prior is also versatile, halving error in wheel encoder odometry-based localization for a robot without retraining.

5.3 Future Work

While we are able to achieve significant improvements in localization when only odometry and map data is available, there remain several key avenues for future work. Since our method is theoretically invariant to the underlying source of odometry as long as it is trained with realistic noise, generating synthetic ground truth trajectories for training may reduce data collection burden. Wang et al. [25] proposes a method to generate realistic human motion trajectories given only images ("scenes") of an indoor environment. Utilizing techniques like this could support the generation of long realistic training trajectories in an environment with minimal data collection effort.

An additional benefit of training data generation could be easing the effort required to collect a large training dataset in different buildings. In our experiments, we noted that generalization to buildings with very different configurations did not work

especially well but were unable to resolve this problem due to access to data from only 3 buildings. A larger number of buildings with training data available would allow for training on a variety of buildings and testing map-branch generalization to unseen buildings. If the map branch was able to generalize to new building configurations easily, the amount of training data required for a new building would decrease dramatically. As long as the chosen odometry method generalized to new spaces and a map was available, no new data would need to be collected to generalize to a new space if the map branch could generalize successfully.

Another direction for future work might be utilizing a fully differentiable particle filter [6] in place of our traditional method. Such particle filters are able to learn the various parameters of a particle filter (*e.g.* particle dynamics models, resampling systems, re-initialization heuristics) in a data-driven fashion and have been shown to outperform hand-defined filters for a given task. By making the learning process backpropagate from error between filter-estimated and ground truth positions, our prior may learn more useful information than our hand-defined target is currently able to provide. In addition, our heuristic re-initialization system for resetting the filter when it begins to predict extremely unlikely locations could be learned, reducing the need for tuning to a specific map.

Bibliography

[1] Harsh Agarwal, Navyata Sanghvi, Vivek Roy, and Kris Kitani. DeepBLE: Generalizing RSSI-based Localization Across Different Devices. *arXiv:2103.00252 [cs]*, February 2021. URL http://arxiv.org/abs/2103.00252. arXiv: 2103.00252. 2.3, 4.2

[2] Dragan Ahmetovic, Cole Gleason, Chengxiong Ruan, Kris Kitani, Hironobu Takagi, and Chieko Asakawa. NavCog: a navigational cognitive assistant for the blind. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 90–99, Florence Italy, September 2016. ACM. ISBN 978-1-4503-4408-1. doi: 10.1145/2935334.2935361. URL https://dl.acm.org/doi/10.1145/2935334.2935361. 1, 2.3, 4.2

[3] Stephane Beauregard, Widyawan, and Martin Klepal. Indoor PDR performance enhancement using minimal map information and particle filters. In *2008 IEEE/ION Position, Location and Navigation Symposium*, pages 141–147, Monterey, CA, USA, 2008. IEEE. ISBN 978-1-4244-1536-6. doi: 10.1109/PLANS.2008.4570050. URL http://ieeexplore.ieee.org/document/4570050/. 2.1, 2.2

[4] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. IONet: Learning to Cure the Curse of Drift in Inertial Odometry. *arXiv:1802.02209 [cs]*, January 2018. URL http://arxiv.org/abs/1802.02209. arXiv: 1802.02209. 2.1

[5] William Falcon, Jirka Borovec, Adrian Wälchli, Nic Eggert, Justus Schock, Jeremy Jordan, Nicki Skafte, Ir1dXD, Vadim Bereznyuk, Ethan Harris, Tullie Murrell, Peter Yu, Sebastian Præsius, Travis Addair, Jacob Zhong, Dmitry Lipin, So Uchida, Shreyas Bapat, Hendrik Schröter, Boris Dayma, Alexey Karnachev, Akshay Kulkarni, Shunta Komatsu, Martin.B, Jean-Baptiste SCHIRATTI, Hadrien Mary, Donal Byrne, Cristobal Eyzaguirre, Cinjon, and Anton Bakhtin. PyTorchLightning/pytorch-lightning: 0.7.6 release, May 2020. URL https://zenodo.org/record/3828935. 4.1

[6] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018. ISBN 978-0-9923747-4-7. doi: 10.15607/RSS.2018.XIV.001. URL http://www.roboticsproceedings.org/rss14/p01.pdf. 5.3

[7] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980. 4.1

[8] Jun S. Liu and Rong Chen. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association*, 93(443):1032–1044, September 1998. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.1998.10473765. URL http://www.tandfonline.com/doi/full/10.1080/01621459.

1998.10473765. 2.2

[9] Wenxin Liu, David Caruso, Eddy Ilg, Jing Dong, Anastasios I. Mourikis, Kostas Daniilidis, Vijay Kumar, and Jakob Engel. TLIO: Tight Learned Inertial Odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660, October 2020. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2020.3007421. URL http://arxiv.org/abs/2007.01867. arXiv: 2007.01867. 1, 2.1

[10] An Luo, Shenghua Chen, and Bin Xv. Enhanced Map-Matching Algorithm with a Hidden Markov Model for Mobile Phone Positioning. *ISPRS International Journal of Geo-Information*, 6(11):327, October 2017. ISSN 2220-9964. doi: 10.3390/ijgi6110327. URL http://www.mdpi.com/2220-9964/6/11/327. 2.2

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. ISSN 1533-7928. URL http://jmlr.org/papers/v9/vandermaaten08a.html. 4.3

[12] Robert A MacLachlan and Artur Dubrawski. Applied Indoor Localization: Map-based, Infrastructure- free, with Divergence Mitigation and Smoothing. page 2. 1, 2.2

[13] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, Zurich, June 2011. IEEE. ISBN 978-1-4244-9862-8 978-1-4244-9863-5 978-1-4244-9861-1. doi: 10.1109/ICORR.2011.5975346. URL http://ieeexplore.ieee.org/document/5975346/. 2.1, 3.2.3

[14] Uche Onyekpe, Vasile Palade, Anuradha Herath, Stratis Kanarachos, and Michael E Fitzpatrick. WhONet: Wheel Odometry Neural Network for Ve-

hicular Localisation in GNSS-Deprived Environments. page 26. 2.1

[15] Shreyas Padhy. shreyaspadhy/UNet-Zoo, May 2021. URL https://github.com/shreyaspadhy/UNet-Zoo. original-date: 2017-12-09T20:48:15Z. 3.1.1

[16] Pratik Palaskar, Rajesh Palkar, and Mayur Tawari. Wi-Fi Indoor Positioning System Based on RSSI Measurements from Wi-Fi Access Points –A Tri-lateration Approach. 5(4):5, 2014. 1, 2.3

[17] Cheng Peng and David Weikersdorfer. Map as The Hidden Sensor: Fast Odometry-Based Global Localization. *arXiv:1910.00572 [cs]*, September 2019. URL http://arxiv.org/abs/1910.00572. arXiv: 1910.00572. 2.2

[18] Adrian Rechy Rormero, Paulo V K. Borges, Andreas Pfrunder, and Alberto Elfes. Map-Aware Particle Filter for Localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2940–2947, Brisbane, QLD, May 2018. IEEE. ISBN 978-1-5386-3081-5. doi: 10.1109/ICRA.2018.8460707. URL https://ieeexplore.ieee.org/document/8460707/. 1, 2.2, 4.2

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*, May 2015. URL http://arxiv.org/abs/1505.04597. arXiv: 1505.04597. 3.1.1

[20] Dan Simon. *Optimal state estimation: Kalman, H [infinity] and nonlinear approaches*. Wiley-Interscience, Hoboken, N.J, 2006. ISBN 978-0-471-70858-2. OCLC: ocm64084871. 2.2

[21] Suherman, Fahmi, Waleed Al-Azzawi, Marwan Al-Akaidi, Emerson P. Sinulingga, and Naemah Mubarakah. Radio-Friendly Building for Efficient Signal Distribution. In *2018 IEEE International Conference on Communica-*

tion, Networks and Satellite (Comnetsat), pages 60–63, November 2018. doi: 10.1109/COMNETSAT.2018.8684092. 2.3

[22] Scott Sun, Dennis Melamed, and Kris Kitani. IDOL: Inertial Deep Orientation-Estimation and Localization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6128–6137, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16763. Section: AAAI Technical Track on Intelligent Robots. 1, 2.1, 3.2.1, 4.2, ??

[23] Adel Thaljaoui, Thierry Val, Nejah Nasri, and Damien Brulin. BLE localization using RSSI measurements and iRingLA. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 2178–2183, March 2015. doi: 10.1109/ICIT.2015.7125418. 2.3

[24] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, Cambridge, Mass, 2005. ISBN 978-0-262-20162-9. OCLC: ocm58451645. 3.1.3

[25] Jingbo Wang, Sijie Yan, Bo Dai, and Dahua LIn. Scene-aware Generative Network for Human Motion Synthesis. *arXiv:2105.14804 [cs]*, May 2021. URL http://arxiv.org/abs/2105.14804. arXiv: 2105.14804. 5.3

[26] Hao Xia, Jinbo Zuo, Shuo Liu, and Yanyou Qiao. Indoor Localization on Smartphones Using Built-In Sensors and Map Constraints. *IEEE Transactions on Instrumentation and Measurement*, 68(4):1189–1198, April 2019. ISSN 0018-9456, 1557-9662. doi: 10.1109/TIM.2018.2863478. URL https://ieeexplore.ieee.org/document/8444074/. 1, 2.2

[27] Zhuoling Xiao, Hongkai Wen, Andrew Markham, and Niki Trigoni. Lightweight map matching for indoor localisation using conditional random fields. In *IPSN-14*

Proceedings of the 13th International Symposium on Information Processing in Sensor Networks, pages 131–142, April 2014. doi: 10.1109/IPSN.2014.6846747. 1, 2.2

[28] Hang Yan, Sachini Herath, and Yasutaka Furukawa. RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, and New Methods. *arXiv:1905.12853 [cs]*, May 2019. URL http://arxiv.org/abs/1905.12853. arXiv: 1905.12853. 1, 2.1, 4.2

[29] Ji Zhang and Sanjiv Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation, July 2014. ISBN 978-0-9923747-0-9. doi: 10.15607/RSS.2014.X.007. URL http://www.roboticsproceedings.org/rss10/p07.pdf. 1

[30] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181, Seattle, WA, USA, May 2015. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139486. URL http://ieeexplore.ieee.org/document/7139486/. 1, 2.3

[31] Kai Zhao, Jie Feng, Zhao Xu, Tong Xia, Lin Chen, Funing Sun, Diansheng Guo, Depeng Jin, and Yong Li. DeepMM: Deep Learning Based Map Matching with Data Augmentation. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 452–455, Chicago IL USA, November 2019. ACM. ISBN 978-1-4503-6909-1. doi: 10.1145/3347146.3359090. URL https://dl.acm.org/doi/10.1145/3347146.3359090. 2.2