LOCALIZATION USING A PARTICLE FILTER LSTM NETWORK IN HABITAT AI

RASHMI PHADNIS [RPHADNIS@SEAS.UPENN.EDU],
WILLIAM C FRANCIS [WILLCF@SEAS.UPENN.EDU],
HARSHINI VIKRAM [VIKRAMHA@SEAS.UPENN.EDU],

ABSTRACT. This project explores the use of a new family of Long Short Term Memory networks applied to the problem of robot localization in the AI Habitat simulator. The LSTM used is a Particle Filter-LSTM which employs particle filtering as the embedded algorithmic prior. Several experiments are conducted to test the PF-LSTM by generating trajectories and collecting observations in Habitat simulator. The 3D-World in Habitat AI is converted to a 2D map to use as an input. Landmarks were introduced to provide 2D observations instead of 3D sensors. The results are evaluated by comparing the deviation of the resulting trajectories from PF-LSTM and the ground truth trajectories from the simulation. The estimated trajectories averaged an MSE of 0.4 from the ground truth, for a justifiable trade-off of reduction in training data by almost 460 times.

1. INTRODUCTION

LSTMs have been extraordinarily successful for prediction with sequential data. To tackle highly variable and multi-modal real-world data, we introduce Particle Filter LSTM (PF-LSTM) networks, a new LSTM family that explicitly models uncertainty in its internal structure. The main difference is while an LSTM relies on a long, deterministic latent state vector, a PF-LSTM maintains a latent state distribution, approximated as a set of particles.

We use the PF-LSTM network in the problem of robot localization. The model gets distances from certain landmark points distributed over the map and the environment map as inputs over a series of trajectories. The maps and trajectories are simulated in the Habitat AI environment. From these, the PF-LSTM gives us particles and the output. The particles are estimated poses and the output is the weighted mean of these estimated poses.

1.1. **Contributions.** In this project, we present the following contributions:

(1) An interface between a localization system which uses PF-LSTM and the Habitat-Sim simulator that can effectively evaluate the performance of PF-LSTM in different environments.
(2) Realized the localization problem in a top-down map by translating the 3D Habitat environment to a 2D map environment and introduced landmarks that provide simulated observations.

2. BACKGROUND

2.1. **PF-LSTM.** Using sequential data for prediction in machine learning has always been a challenge. For effective prediction, predictors require “memory”, which summarizes and tracks information in the input sequence. We need a belief since this memory state is not observable,i.e., a posterior state distribution that captures the sufficient statistic of the input for making predictions. Sequence predictors often use LSTMa, which learn a vector h of deterministic time-dependent latent variables as an approximation to the belief. The challenge is real-world data are highly variable and multi-modal.To cope with the complexity of uncertain real-world data and achieve better belief approximation, one could increase the length of latent vector h, thus increasing the number of network parameters and the amount of data required for training.
We use PF-LSTMs, a new family of LSTMs that seeks to improve belief approximation without lengthening the latent vector h, thus reducing the data required for learning. [1]

2.2. **Particle Filtering.** Particle Filters are a great way to approximate a belief over states given an observation. The approximated belief is given by a set of particles which are sampled from the probability distribution. For a given set of particles and their weights $< s_t^k, w_t^k >_{k=1:K}$ where k is the number of particles and t is the timestep, we can get the estimated output by using the weighted mean: $\sum_{k=1}^{K} s_k w_k$. The particle states and weights are periodically updated. To deal with degeneracy when the weights become too small, we use resampling.

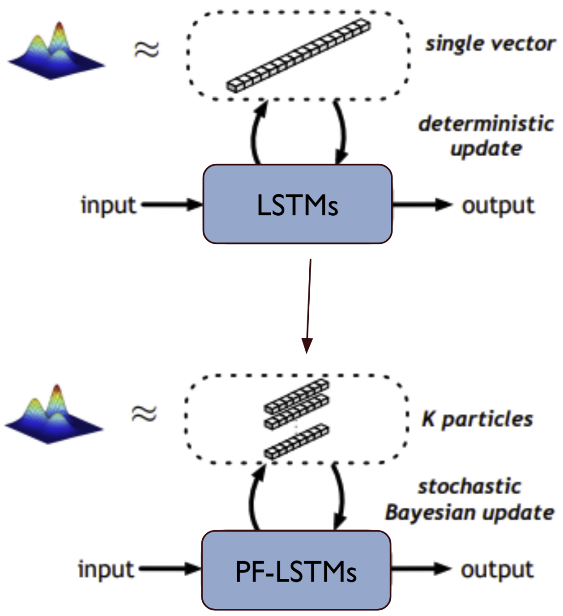


FIGURE 1. LSTMs & PF-LSTMs

2.3. **Environment: Habitat AI.** Habitat is a simulation platform for research in Embodied AI. It enables training of such embodied AI agents (virtual robots and egocentric assistants) in a highly photorealistic & efficient 3D simulator, before transferring the learned skills to reality. This empowers a paradigm shift from 'internet AI' based on static datasets (e.g. ImageNet, COCO, VQA) to embodied AI where agents act within realistic environments, bringing to the fore active perception, long-term planning, learning from interaction, and holding a dialog grounded in an environment. Habitat-sim is a highly efficient, photorealistic 3D simulator for embodied agents operating in the virtual environments. We have features like loading a scene, setting and configuring an agent, as well as its sensors (e.g., color, semantic, and depth sensors), instruct the agent to navigate and obtain the observations.
Navmesh: An efficient mechanism to handle the collision constraints during the navigation. We used Navmesh in our project to ensure our trajectories stay within the boundaries.

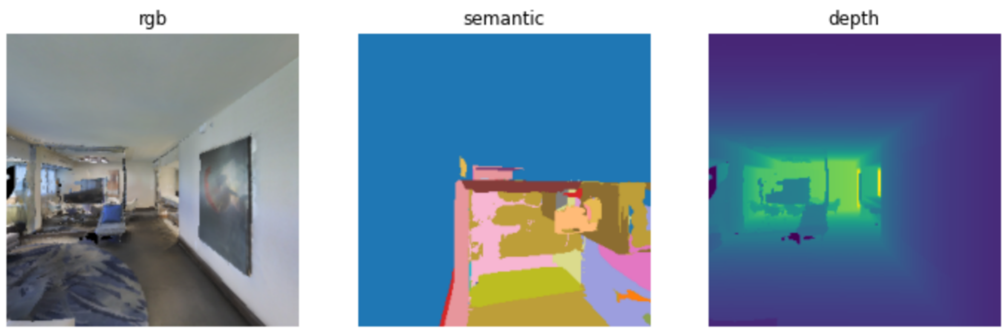


FIGURE 2. Habitat AI 3D sensor data visualization

3. RELATED WORK

Sequential data prediction is often tackled with either of the two techniques: model based and model free. The model based approaches include Hidden Markov models (HMMs), and Bayesian Networks. Particle filters could be used to represent the belief as a set of sampled states. Model based approaches, however, take up high computational complexity because the state space of a model based network grows exponentially with the number of state dimensions.

To tackle this difficulty, a model free approach can be employed. An LSTM can be used to approximate the belief as a latent state vector and update it through a deterministic non linear function learned from the data during training.

PF-LSTMs builds upon this idea of a model free method and combines its powerful data-based approximation capabilities with the sample-based belief representation and approximate Bayesian inference used in particle filters. Some generative models have already employed this model such as the variational auto-encoder described in the 2015 paper [2]. Further exploration of this idea was witnessed in sequence generation [3] and reinforcement learning [4]. In this work, we combine particle filters and LSTMs and train them discriminatively, instead of generatively, for sequence prediction. In such a discriminative learning environment, the target loss function becomes the model output, enabling target prediction instead of generation.

Earlier work embeds a particle filter in an RNN for learning belief tracking, but follows a model-based approach and relies on handcrafted belief representations [5]. In PF-LSTMs, we exploit the idea of using algorithmic priors [6] and training filtering algorithms in neural networks discriminatively [5]. Jonschkowski et al., in their paper, follows a model based approach which relies on handcrafted belief representations as opposed to a model free approach in PF-LSTM, utilizing LSTMs' powerful approximation capabilities to learn belief representation directly from data. Belief representation learning has been previously explored on RNNs [7], but without leveraging particle filtering or Bayesian belief updates.

4. Approach

Our simulation environment uses a 3D world. The traditional approach to solving localization problems in Habitat AI involve taking information from the 3D environment as observations. For example, the image data from the camera sensor and depth data from the depth sensor could be used for the update step during agent localization. This, however, is not feasible for the PF-LSTMs since the LSTM would require extremely large computing capabilities to train on image data for thousands of trajectories. For the purpose of our project, we created a 2D top-down map of the 3D Habitat AI environment. The localization pipeline constitutes of the following three blocks:

(1) Creating the 2D top-down map from an environment in Habitat AI and defining landmarks
(2) Initializing an agent and creating trajectories for training and evaluation
(3) PF-LSTM

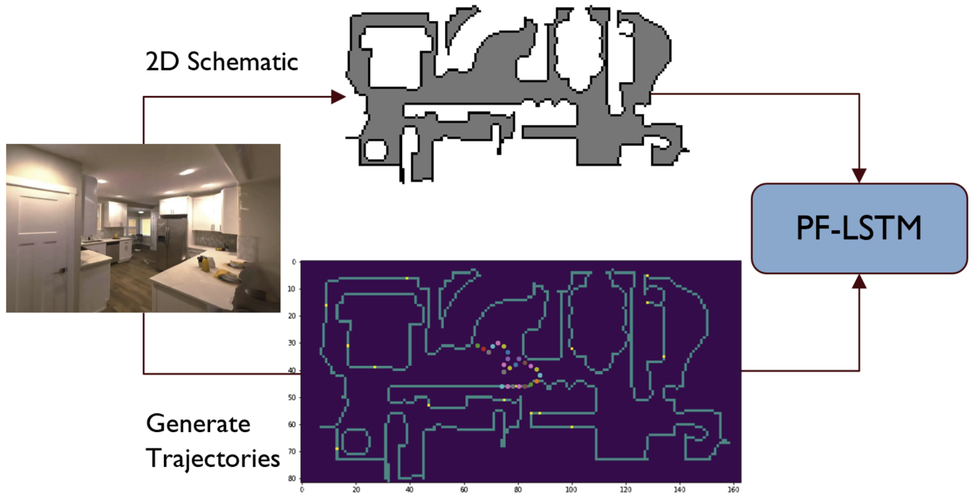


FIGURE 3. High-Level Overview

4.1. Map creation.

Using a 2D map instead of a 3D environment for the task not only helps simplify the localization task but also improves the efficiency of the PF-LSTM as it can process more number of trajectories without a substantial increase in training data. We created a topdown view of a scene, which is just a two-dimensional birds-eye representation of the scene. The topdown view is a two-dimensional array of 1s and 0s where 1 means that pixel is "navigable" in the scene (i.e. an agent can walk on top of that point) and 0 means that pixel is "unnavigable".

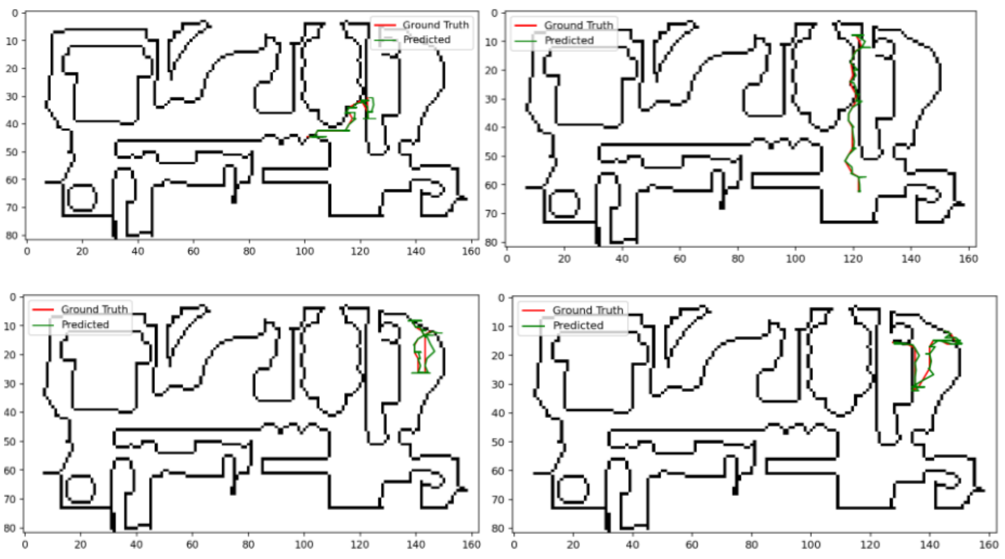
By reducing the dimensions of the environment, we are essentially sacrificing on data which could be otherwise used for more accurate state estimation. We also lose the ability to use image or depth data as the observations for our PF-LSTM. While this is advantageous in substantially reducing the compute, time and data required, it now poses the question of ”what constitutes the observations in the 2D map?”. We introduce the concept of using landmarks to collect observations. Sixteen landmarks are randomly created on the map with four in each quadrant, visualized in Figure 3 as yellow points. These landmarks are placed as a value of ’2’ along the walls of the environment. Now the map constitutes of three values: ’0’ - empty, ’1’ - walls, and ’2’ - landmarks. At each time step, the observations received by the agent are the distances of the k closest landmarks from the agent. We used k=5 for training and validation. The landmarks were evenly distributed across the four quadrants of the map so that the agent receives observations from different k landmarks at each time step.

4.2. **Trajectory Generation.** The trajectory generation was implemented in an environment in Habitat AI. The environment loaded was an apartment room. The agent that was used could perform 3 actions - turn left, turn right, move forward. We had the agent navigate in the loaded environment for 80 timesteps by randomly choosing one of the three actions. After each action was performed, the X-coordinate, Y-coordinate and the rotation in quaternion was collected. The X and Y co-ordinates were transformed to be in the scale of the 2D map. The quaternion was converted into Euler angles and then the yaw was stored. In this way, we generated 10,000 trajectories with 80 timesteps for training and 500 trajectories for testing with each trajectory timestep represented by X, Y, θ pose of the agent. The action taken by the agent is encoded as the difference between the current pose and the old pose.

4.3. **Network Architecture.** The network receives the 2D schematic map with the landmarks as an input along with the observations which are the (X, Y, θ) pose, the actions and the distances from the 5 nearest landmarks. The map is encoded into a one-dimensional vector by using a 2-layer CNN network. The observations are encoded in a similar manner. These 2 vectors are combined together by a simple element-wise multiplication. The combined vector is passed to our PF-LSTM as the input. The generated particles are the estimated poses. We chose 30 as our number of particles. The output is the weighted mean of these particles that is the weighted mean of the estimated poses.

5. Experimental Results

We trained and tested the PF-LSTM in different simulation scenes from Habitat AI test scenes. Figure 2 shows an example image from an apartment scene. The scenes were converted into a 2D map of the environment along with landmarks added as shown in Fig. 3. The test observations were obtained from the Habitat AI environment by randomly generating 500 trajectories and collecting the pose, action and distance from landmarks. To evaluate the generated trajectories, we performed qualitative analysis by plotting the estimated and ground truth along with a quantitative analysis which was the Mean Squared Error between the estimated and ground truth.
Qualitative Results:



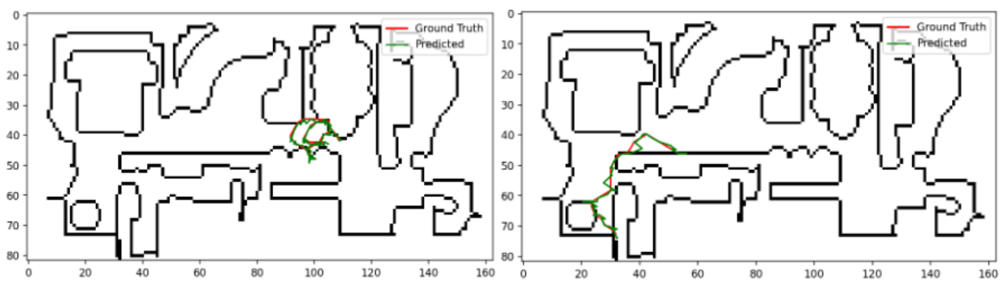


FIGURE 4. Estimated and Ground Truth Trajectories

Quantitative Results: For the shown trajectories, the MSE loss is as follows:

Trajectories	1	2	3	4	5	6
MSE (X-coordinate)	0.5	0.3635	0.5517	0.3683	0.3662	0.3970
MSE (Y-coordinate)	0.439	0.6885	0.2056	0.3060	0.5440	0.4318

6. DISCUSSION

The PF-LSTM model, despite a massive reduction in training data, performed fairly well on the localization task. Let us examine the extent of data lost and the extent of estimation accuracy lost, and determine if the trade off is justifiable. A camera observation would require an image of 48x48 pixels as observations at each time step for 10,000 trajectories for training. That is $2.3 * 10^7$ data points for training. In contrast, the landmark-based observation model introduced in this project uses only observations for 10,000 trajectories which is 50,000 data points. Hence, we are reducing the data by approximately 460 times. On an average, the estimation on the other hand deviates from the ground truth by an MSE of 0.5. The estimated trajectories seem to follow the ground truth quite effectively and this trade-off between the decrease in data and the increase in error may be justified. The reduced data model still takes around 30 hours to train on an AWS EC2 instance with NVIDIA T4 GPUs.

Due to limited compute and time, this project only utilized a top-down map and 2D observations. Given ample time and computing power, this solution can be combined with the traditional Habitat AI sensory inputs such as the RGB camera and depth camera to produce more accurate state estimations. A large amount of data in terms of the surrounding objects is lost while converting the data from 3D to 2D such as the placement of furniture in a room, the color of the walls, etc,. The architecture could be further optimized by embedding the images or depth information through a CNN before feeding into the LSTM. The LSTM could also be replaced with a transformer to form a PF-Transformer. State-of-the-art transformer models have recently proved to outperform LSTMs in many sequential data models and would likely translate well into the localization problem at hand.

References

[1] ***Particle Filter Recurrent Neural Networks***, Xiao Ma, Peter Karkus, David Hsu, Wee Sun Lee

[2] ***Importance weighted autoencoders***, Burda Y., Grosse R., Salakhutdinov

[3] ***Auto-encoding sequential monte carlo***, Le T. A., Igl M., Rainforth T., Jin T., and Wood F

[4] ***Deep variational reinforcement learning for POMDPs***, Igl M., Zintgraf L., Le T. A., Wood F., and Whiteson S.

[5] ***Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors***, Jonschkowski R., Rastogi D., and Brock O.

[6] ***Differentiable ´ algorithm networks for composable robot learning***, Karkus P., Ma X., Hsu D., Kaelbling L. P., Lee W. S., and Lozano-Perez T

[7] ***Temporal difference variational auto-encoder***, Gregor K., and Besse F

[8] ***Habitat: A Platform for Embodied AI Research***,Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, Dhruv Batra

[9] ***Habitat 2.0: Training Home Assistants to Rearrange their Habitat***,Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, Dhruv Batra

[10] ***Discriminative Particle Filter Reinforcement Learning for complex partial observations***, Xiao Ma, Peter Karkus, David Hsu and Wee Sun Lee, Nan Ye