Belief-Based Localization using Deep Neural Networks

Timmy Hussain and Alexandros Tzikas
Stanford University

BIOGRAPHY

Timmy Hussain is an M.S Candidate in the Department of Aeronautics and Astronautics at Stanford University. He received his B.S in Aerospace Engineering from Massachusetts Institute of Technology. His research interests include the control and navigation of autonomous systems.

Alexandros Tzikas is a Ph.D. Candidate in the Department of Aeronautics and Astronautics at Stanford University. He received his B.S. in Electrical and Computer Engineering from the Aristotle University of Thessaloniki. His research interests are in the field of navigation of autonomous systems.

I. INTRODUCTION

Autonomous vehicles are an exciting technological area that has spawned several research directions in the past few decades. It is an intersection of numerous disciplines ranging from aerospace to electrical and mechanical engineering among others. Different parts of the autonomy stack pose different challenges. One of these challenges is the problem of robot localization: the ability of a robot system to infer its position in an environment. Localization is pivotal in the robot's ability to perform a task.

The problem of robot localization is affected by several different factors. One of these is noise and uncertainty in the robot's sensing ability and motion. Left unchecked, this can easily result in a degradation of the robot's localization ability over time as errors in its belief will accumulate. Another factor to take into consideration is the robot's ability to sense and gather full or partial information about its environment. The sensing modality contributes strongly to the robot's ability to localize within its environment, but introduces further considerations, trade-offs and noise.

It is typical to consider localization works along the axis of global and local localization. Global localization refers to a robot's ability to infer its position from a global frame of reference in contrast to local/relative localization approaches, which aim to compute a position estimate relative to some given initial or prior belief of the robot's position. Sensor modalities such as GNSS receivers are able to provide global localization but are strongly dependent on the environment (line of sight is required between receiver and GNSS satellites), while environment-agnostic sensors such as IMU/odometry encoders work well for local localization, but can easily fall victim to error accumulation over time and could lead to large deviations between the robot's belief of its position and its actual position. In addition, local localization algorithms often make restrictive assumptions about the agent's location and might not be able to recover after a localization error, as described in [1]. Thus, global localization techniques are often preferable.

The consequences of inaccurate localization also lend gravity to the localization problem. Robot systems do not often work in isolated environments. Typically some task is intended to be completed, which will involve interaction with other objects in the environment or other known or unknown agents, such as humans or cars. An inability to accurately localize can be detrimental to the safety not only of the robot system but of other agents in the environment[2].

Several approaches have emerged in the past few decades which approach the localization problem from different perspectives. One prominent approach is sensor fusion which essentially bridges the gap between different sensor modalities. These incur challenges in data fusion and require principled frameworks to be successful. They often also require precise calibration and knowledge of the sensor systems.

In recent years, there has been a shift towards more end-to-end and learning-based approaches[3]. End-to-end approaches have two benefits, They preclude the need for explicit sensor modelling or principled fusion techniques and they allow the localization problem to be solved alongside a secondary objective, such as navigation. While this is the case, and in contrast to the traditional approach which isolates the localization problem, it is not uncommon to have learning-based localization solutions embedded within an end-to-end network.

Learning-based architectures also open up the possibility of combining multiple information sources without needing to explicitly

model the relationship between the sources. However, while these approaches do not require principled models of the underlying information sources, the use of neural networks still elicits the need for a principled model architecture, training and evaluation process. This work aims to develop, train and apply such a model architecture towards the problem of global localization within a known environment through the use of 2D LiDAR point cloud data.

II. RELATED WORK

There exists a significant amount of work attempting to solve the problem of global localization given a known map. Geometric approaches to map matching like the Iterative Closest Point algorithm[4] are often successful but can easily fall into pitfalls when the environment has repetitive features or geometries. Image-based approaches can do well in terms of feature detection[5] which provide additional localization information but are not entirely agnostic to environmental factors such as lighting and occlusions.

Certain approaches combine solving the problems of both Simultaneous Localization and Mapping (SLAM) and, again, these can vary depending on the sensors of choice. However, the heart of a number of SLAM approaches is a probabilistic filtering using Bayesian statistics in a similar fashion to [6].

Use of LiDAR within learning has also been explored although the formulations are often quite different. In [7], the LiDAR measurements are passed through a network to produce an intensity map which is compared to a similar embedding obtained from the ground truth map and the comparison is done via convolutional matching. While this approach extends well to any LiDAR model, this requires training a model to to generate this embedding given a measurement.

Other approaches avoid the problem of learning structure from data and instead pass as inputs into networks unordered sets of point cloud data. An influential work in this domain in PointNet[8] network which is able to perform 3D classification and segmentation given only point cloud data. While this helps solve the localization problem in a similar way to visual feature recognition and semantic object detection, it is an indirect approach.

Finally, other approaches exist which leverage more directly images and advances in computer vision by transforming point cloud data to visual representations using 3D structures such as voxels[9]. However, this does not directly result in a belief of a robot's position and results in needlessly complex representations of LiDAR point cloud data.

In this work we avoid learning unnecessary representations of point cloud data and instead focus on inferring a belief of a robot's position given a known map.

The remainder of the paper is organized as follows.In Section III we describe the problem formulation. Section IV provides details of our proposed architecture. Finally, in Section V we present simulation results .

III. PROBLEM STATEMENT

We assume a known map of the environment in which the agents of the problem are located. The map consists of an occupancy grid M of dimensions $d_x \times d_y$, where each cell in the grid is either empty, occupied by an obstacle or occupied by an agent. An occupied cell is represented by a 1 in the occupancy map, while an empty cell is represented as a 0 in the occupancy map. The dimensions of the cells dx, dy are assumed to be known in advance.

The problem we aim to solve is that of state estimation of an agent from various incoming types of data. We adopt a probabilistic approach and account for the uncertainty in the agent's state via a probabilistic distribution, which is called a belief. Accounting for uncertainty is necessary because both the sensing and motion of the robot are corrupted by noise. The belief at a specific state is the probability of the agent being in that state. Given the input data that can include range measurements, odometry, previous agents' beliefs and actions, we aim to update the belief distribution in order for it to converge to a distribution concentrated around the agent's true state.

The state of the agent consists of three elements: x-coordinate, y-coordinate and orientation θ. The x-coodrinate can take values $\{1, ..., d_x\}$. The y-coordinate can take values $\{1, ..., d_y\}$ and there is also a discrete number of possible orientations d_θ, where each index corresponds to a different angle. Therefore the belief is actually a discrete probability distribution over $d_x \times d_y \times \theta$ possible outputs-state tuples $l = (x, y, \theta)$. We represent the belief that the robot is in state l at time t as:

$$Bel(L_t = l). \tag{1}$$

IV. PROPOSED APPROACH

The architecture for the proposed system is shown in Figure 1. The proposed approach is a Convolutional Neural Network which takes in as inputs the LiDAR measurements and an occupancy grid representation of the map M and outputs a belief over the

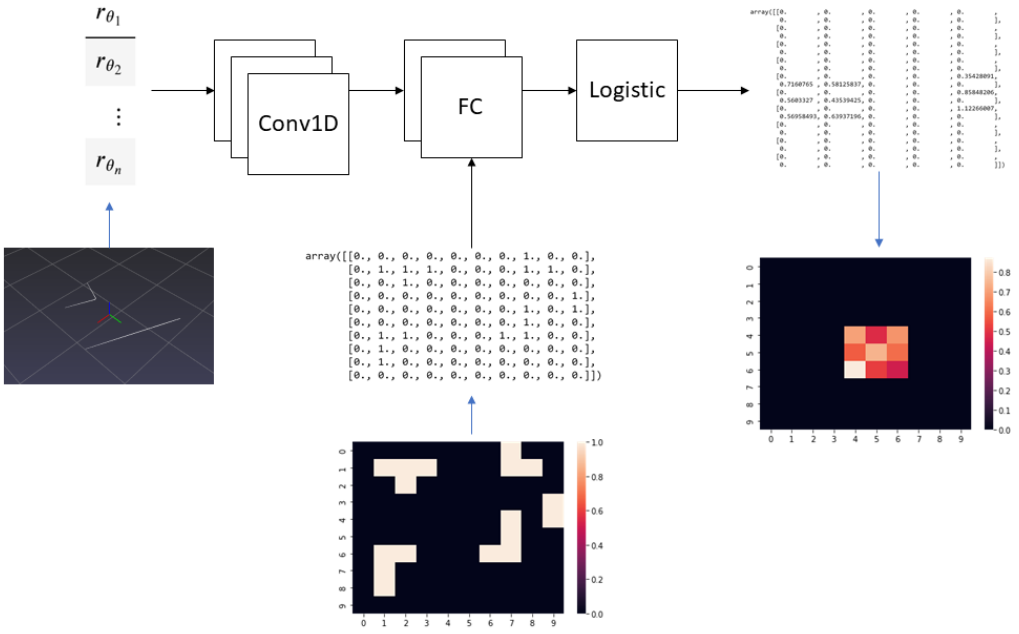


Figure 1: Proposed Neural Network architecture

space of the robot's position. The architecture consists of a series of Convolutional (and max pool) layers followed by a series of dense fully connected (FC) layers which will take in the outputs of the convolution layers and a vector representation of the map. The output of the FC layers is passed into a logistic activation function to constrain the magnitude of the output between 0 and 1. Subsequently, this can be normalized to produce a probability distribution over the space.

One of the challenges of working with LiDAR data specifically is that it is unstructured. In contrast with image data which is of fixed dimensions determined by the resolution of the camera, LiDAR measurements can vary in the number of points depending on the environment. The raw LiDAR measurements are of the following form:

$$\mathbf{O_1} = \begin{bmatrix} x_1, y_1 \\ \vdots \\ x_n, y_n \end{bmatrix}$$

where n represents the number of points received in the scan. However, because n varies between measurements, we develop the following transformation:

$$\mathbf{O_1'} = \begin{bmatrix} r_{\theta_1} \\ \vdots \\ r_{\theta_n} \end{bmatrix}$$

where $\theta_i = 0 + 0.5i$ for all integers i between 0 and 719 and $r_i = \frac{\sum_j^{n'} \sqrt{x_j^2 + y_j^2}}{n'}$ for all measurements, such that $\theta_i < \tan^{-1}(x_i/y_i) \leq \theta_{i+1}$. This results in a fixed size vector of length 720 with values corresponding to the average range measurement of the data points at angle separations of 0.5^o.

V. PRELIMINARY RESULTS

1. Baseline Algorithm

The baseline approach is based on [10] and [6] and is termed Markov localization. Markov localization is a method that globally estimates the state of a robot in the environment. It is based on a probabilistic framework, since it uses a belief to represent the confidence about the state of the agent. Markov localization is able to re-localize the robot in the case of localization failures, while also providing accurate position estimates [10]. The method discussed in [10] and [6] uses a fine-grained discretization of the state-space. This type of representation has several advantages over previous ones, such as using Gaussian or coarse-grained topological representations, because it provides more accurate estimates and can incorporate raw sensor inputs.

In our simulations, we have also adopted the assumption from [10] that the environment is static (Markov assumption), meaning that the robot's location is the only factor affecting sensor inputs.

We also assume we start with an initial belief $Bel(L_0)$ that is uniform over all possible states. The belief is updated as follows, assuming that we have the probability distribution $Bel(L_{t-1} = l)$ at time $t-1$:

- If at time t a sensor measurement s_t appears, then the belief at every state $l = (x, y, \theta)$ is updated as follows:

$$Bel(L_t = l) = a_t p(s_t|l) Bel(L_{t-1} = l), \tag{2}$$

  where a_t is just a normalization constant and $p(s_t|l)$ is the environment perception model. The likelihood that we observe sensor measurement s given that we are in state l is dependent only on the expected measurement from state l, o_l, as in [6]. Therefore:

$$p(s_t|l) = p(s_t|o_l). \tag{3}$$

  This probability distribution is assumed to be Gaussian in our case, centered around the expected measurement, as in [10]. Binning is used to group together continuous sets of measurements and assign them a single probability. This is intuitively necessary, given that our environment is assumed discrete. The standard deviation is left as a hyper-parameter and depends on the accuracy of the sensor model and the environment model. We choose a specific value in the simulations that will follow.

- If at time t an odometry measurement s_t arrives, then the belief at every state l is updated as follows:

$$Bel(L_t = l) = \int p(l|s_t, l') Bel(L_{t-1} = l') dl', \tag{4}$$

  where $p(l|s_t, l')$ is the motion model. This can be assumed to be a deterministic probability distribution, since given an odometry measurement in the gird world, we can assume that the agent arrives at the intended cell and orientation. The inherent noise would only alter the continuous position and orientation in the intended cell.

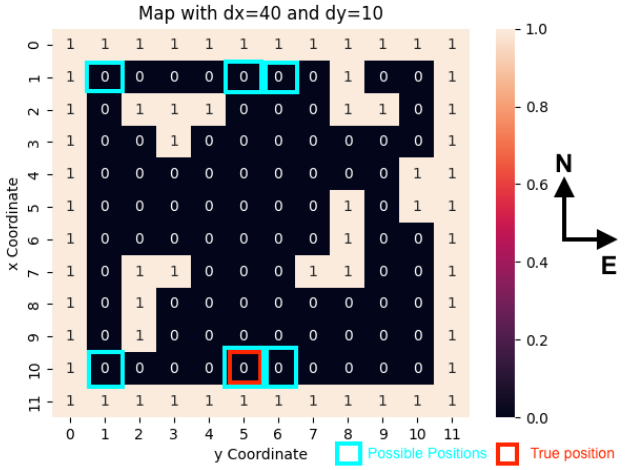
The Markov localization approach presented in [10] and [6] is however limited by two conditions:

- According to the described formulation only one range measurement can be integrated into the belief at each update. This is limiting, since the available LiDAR data usually consist of multiple measurements at different directions. To integrate these data points, it would be necessary to include an odometry update between LiDAR measurements with different directions. This occurs because the previous belief at the previous orientation, must now become the belief at the current orientation, from which we receive measurements. We can also observe that the motion model results in a change of the state (including orientation), which allows us to make the aforementioned assumption.
- The presented formulation is only applicable to static environments, as previously mentioned. However, an extension exists in [10] for dynamic environments.

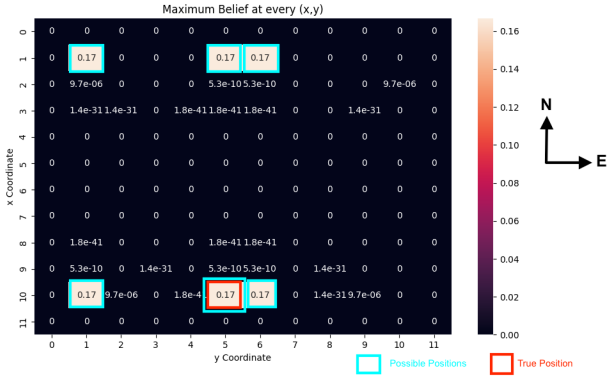
2. Simulations of the Baseline algorithm

In order to demonstrate the correctness of the baseline implementation, we performed some simple initial simulations. We first extracted the AirSim grid map and imported it into our Python code. The map is shown in Figure (2a), where the cells with 1 indicate the presence of obstacles. We assume orientations as shown, where North (N) is upwards and East (E) is towards the right. Also, we assume $dx = 40$, $dy = 10$ and a low standard deviation for the perception model, since we are interested in testing the case where the actual measurement is always really close to the expected measurement (high fidelity sensor model). This helps in validating the correctness of the implementation.

An agent is placed in cell $(10, 5)$ with orientation (N), as indicated by the red box. This means that the expected measurement for that agent is $9.5dx$. Therefore, in order to test our implementation, we feed the baseline algorithm with perturbations



(a) Grid Map from AirSim.



(b) Updated Belief after Range Measurements.

Figure 2: Baseline Simulations.

of this distance measurement to indicate the corruption by sensing noise. It is easily noticeable that there are a few more position-orientation tuples that have the same expected measurement. These are shown in Figure (2a). One of these is, for example the tuple $l = (1, 1, S)$. We expect that the belief table will have greater probabilities for these position-orientation tuples.

By applying the baseline algorithm and updating the initial uniform belief, we get the belief shown in Figure (2b). In this figure only the maximum belief over the orientations is shown for each cell-position. [10] follows the same approach in representing the belief table. We observe 6 cells with a far greater probability than the rest. These correspond to the state tuples with the same expected measurement as the true state, as described previously. In this sense, the baseline algorithm is shown to perform as expected. In addition, we notice a 0 belief for the cells with obstacles, and low values for the other cells. Due to the asymmetry in the dx and dy values, cells not included in the possible positions in the figure have quite different expected measurements and thus have a very low belief.

3. Performance Metrics

The final goal of the project is to compare the performance of the proposed deep-learning based belief model with the approach discussed in [10] and [6]. The comparison will be done as follows. True states for the agent will be sampled from the available (non-obstacle) states in the map. Range measurements will then be performed from each of these states and two belief tables will be produced for each sample: one by passing the range measurements from the sample state to the baseline approach and one by passing them to the proposed deep-learning model. The state with the maximum belief will be considered the state estimate in each of two methodologies. Finally, the MSE error of the estimate with respect to the true position in each sample will be determined for both the baseline and the newly proposed approach.

VI. PLAN FOR PROJECT COMPLETION

Up to this point, the AirSim simulation environment has been set up and is ready to be used as the simulation platform for our experiments. Range measurements have also been collected through AirSim and have been pre-processed in order to be used for the training of the deep-learning belief model.

In addition, the baseline method that will serve as the direct comparison for our method is functional. The plan for the upcoming weeks is therefore the following:

- **Week 9**: Make final decision about the perception model architecture and start training.
- **Week 10**: Test the baseline approach and the newly proposed deep-learning approach with the AirSim collected data and compare the performance of the two schemes.
- **Week 10**: Project Write-Up and presentation

VII. CONTRIBUTIONS

Timmy set up the AirSim simulation environment from which range measurements are taken. He also collected the data and pre-processed them. Alex implemented the baseline approach, which will serve as the main comparison for the proposed method. He also tested the baseline algorithm in simple cases using the AirSim map to establish that it works properly. Both Timmy and Alex reviewed the literature and found prior related work.

VIII. CONCLUSION

The main goal of this project is to provide a superior alternative to classic Markov localization using beliefs, by introducing a deep-learning model that transforms input data to an updated belief about the state of the robot. The deep-learning approaches are known to generalize well to unseen environments and can leverage multiple input sources easily. Therefore, showing that a deep-learning approach performs well in the global localization task is of value to the field of localization and might lead to more efficient localization schemes.

IX. FUTURE DIRECTIONS

After implementing the deep-learning model that determines the localization belief of a single agent, we intend to extend it to a multi-agent system. When determining the belief at each agent we would have to take into account information shared by other agents, such as other agents' beliefs and sensor measurements, in addition to the own agent's LiDAR measurements. We also plan to test the model in more complex maps and possibly extend it to take as inputs the previous belief of the agent as well.

APPENDIX

The code for the project can be found at `https://github.com/alextzik/multi-robot_active_localization`.

REFERENCES

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

[2] Everett, M., Chen, Y. F., and How, J. P., "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.

[3] Lin, J., Yang, X., Zheng, P., and Cheng, H., "End-to-end Decentralized Multi-robot Navigation in Unknown Complex Environments via Deep Reinforcement Learning," *Proceedings of 2019 IEEE International Conference on Mechatronics and Automation, ICMA 2019*, pp. 2493–2500, 2019.

[4] Besl, P. J. and McKay, N. D., "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[5] Ziegler, J., Lategahn, H., Schreiber, M., Keller, C. G., Knöppel, C., Hipp, J., Haueis, M., and Stiller, C., "Video based localization for bertha," in *2014 IEEE intelligent vehicles symposium proceedings*. IEEE, 2014, pp. 1231–1238.

[6] Fox, D., Burgard, W., Kruppa, H., and Thrun, S., "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[7] Barsan, I. A., Wang, S., Pokrovsky, A., and Urtasun, R., "Learning to localize using a lidar intensity map," *arXiv preprint arXiv:2012.10902*, 2020.

[8] Qi, C. R., Su, H., Mo, K., and Guibas, L. J., "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[9] Zhou, Y. and Tuzel, O., "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[10] Fox, D., Burgard, W., and Thrun, S., "Markov Localization for Mobile Robots in Dynamic Environments," *Journal of Artificial Intelligence Research*, vol. 11, no. 1, pp. 391–427, 1999.