Deep Robot Localization

Nathan Hatch Gavin Parpart Daniel Starikov
nhatch2|gparpart|starikov@cs.washington.edu

ABSTRACT

Given an occupancy grid and a one-dimensional depth scan centered around a robot, we wish to localize the robot within the grid using deep learning. Several network architectures are proposed, including a regression model and a classification model. Qualitative results show that none of these approaches performs very well.

1 INTRODUCTION

The typical setup for a robot localization problem is shown in Fig. 1. The robot is in an environment represented as a binary occupancy grid. It has sensor information in the form of an egocentric laser depth scan with limited angular resolution, distance resolution, and field of view. The task is to determine the robot's pose (x, y, θ) given only the occupancy grid and the depth scan.

One approach, described in detail in Section 3, is to discretize the configuration space, simulate a laser scan from each pose, and choose the pose that best matches the given scan. However, this brute-force approach is expensive—it may take a second or two to estimate the pose for a single frame of data. Hence, we are interested in faster methods for robot localization.

Given the recent popularity of deep learning in the robotics community, we decided to try to use neural networks to solve this problem. We experimented with a number of network architectures, described in detail in Section 4. For evaluations, we compared the qualitative performance of these networks against the brute-force force approach of Section 3. We find that while the network predictions are sometimes reasonable, they are not very robust. We discuss potential reasons for this and suggest future research directions in Section 6.

2 DATASET GENERATION

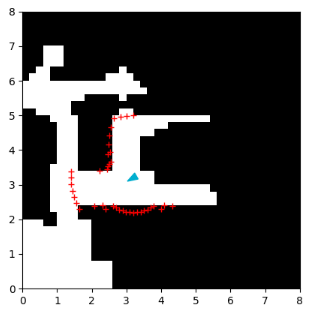
The set of 96 occupancy grids used for training and testing were provided by the instructors of the University of Washington Robotics (CSE 571) Spring 2020 course [website].

We augment this dataset by generating randomly cropped maps around the robot and laser scan data. First, a robot pose and associated laser scan is sampled from the unoccupied space of a randomly chosen map. Our laser scanner uses 40 laser rays with a range of $M = 2$ meters and an FOV of 240°. We then crop the map to a square of side length $4M$ randomly such that the location of the robot is uniform within the central square of side length $2M$. This ensures that our models are not just learning to predict the center of the map. The resolution is chosen so that the cropped occupancy grid is 40 by 40 pixels.

The dataset can be stored on disk or run in online mode where datapoints are generated on the fly as needed. This avoids consuming disk space with training data and enables the use of arbitrarily large datasets.

We partition the set of maps into 61 train, 15 validation, and 20 testing maps. By splitting the source maps used for dataset

Figure 1: Example localization problem.



generation, we are able to test that our model generalizes to new maps.

For more details, please refer to our code, available [here]. For the purposes of input to our neural network architectures, the laser scan is represented as a one-dimensional vector of depth values.

3 BRUTE-FORCE BAYESIAN LOCALIZATION

The Bayesian approach to localization proceeds as follows. Suppose we can compute a probability density function (PDF) for the laser scan s, given a robot pose (x, y, θ) and an occupancy grid m:

$$p(s|x, y, \theta, m)$$

Suppose we also have a prior $p(x, y, \theta|m)$ over robot poses. Then we can compute the posterior

$$p(x, y, \theta|s, m) \propto p(s|x, y, \theta, m)p(x, y, \theta|m) \tag{1}$$

which gives a distribution over robot poses according to how well they match the laser scan with the occupancy grid. To calculate the proportionality constant, we can discretize the configuration space, calculate (1) for each pose, and then normalize.

For the experiments described below, we used the sensor model

$$p(s|x, y, \theta, m) = \prod_{i=1}^{B} \left(\varepsilon N(s_i|s'_i, \sigma^2) + (1-\varepsilon)\frac{1}{M} \right)$$

where s' is the simulated laser scan for the given (x, y, θ, m) pose and map, B is the number of beams per laser scan, M is the maximum range of a laser scan, $\varepsilon \in [0, 1]$ is a mixture weight, and $N(\cdot|\mu, \sigma^2)$ is a one-dimensional normal PDF with mean μ and variance σ^2. Our experiments used $\varepsilon = 0.2$ and $\sigma = 3$ pixels. Some sample posterior distributions are visualized in the top half of Figure 3.

4 NEURAL NETWORK ARCHITECTURES

We implemented two network architectures: a regression model and a classification model. For the classification model, we experimented with two training objectives: cross-entropy loss and KL-divergence.

Both models share a "backbone" consisting of six 2D convolutions on the map and four 1D convolutions on the laser scan, with

ReLU activations, batch normalization, and max pools every 2 convolutions. The laser scan features are flattened and periodically concatenated with the map features at each 2D position.

After the backbone, our regression architecture has two fully connected layers with final output dimension 4. Our classification architecture instead applies two transpose convolutions to increase the resolution, with skip connections back to the earlier high-resolution layers of the backbone, followed by two 1x1 convolutions.

The training objective for the regression model is mean square error (MSE) with respect to the pose (x, y, θ) originally used to generate the laser scan. To address wrap-around for θ and balance the units of θ against those of (x, y), we represent pose targets as $(x, y, 5\sin(\theta), 5\cos(\theta))$, where x, y are measured in pixels.

The KL-divergence objective (KL) for the classification model is $\sum_{x\in X} P(x)\log\left(\frac{P(x)}{Q(x)}\right)$ where $Q(x)$ is the classifier prediction and $P(x)$ is the Bayes prediction (see Section 3). The cross-entropy objective (CE) is the same, except that $P(x)$ is a one-hot distribution with all weight on the pose originally used to generate the scan. The KL objective provides more training signal but is much more expensive to calculate.

We optimize the networks with Adam [1] on the training maps. We use batch size 32, initial learning rate 0.0001, and gradient clipping with norm 100. Since we generate our data online, we do not have training epochs in the normal sense, but we run the optimization for 1000 "epochs" of 2048 examples each.[1] This takes about six hours on an Nvidia GPU.

5 RESULTS

For qualitative evaluations, we compare the predictions of our networks against the brute-force Bayesian approach described in Section 3. See Figures 2 and 3. More qualitative results may be seen in the accompanying video presentation [video] [slides].

The regression model sometimes predicts poses inside of obstacles. Although the classification model avoids this, it still does not understand the geometry of the problem, often predicting poses for which the given laser scan does not make sense. To address this, we simplified the classification model to predict only the robot's (x, y) location, ignoring the heading θ (right column of Figure 3). This seems to perform best out of all of our models, but still it often makes nonsensical predictions. Additional results from this model may be seen in Appendix A.

Given our poor qualitative results, we did not spend much time on quantitative evaluation. We trained the best-performing of each kind of architecture once to completion. The results on validation data are shown in Table 1.

6 DISCUSSION

This project has shown that it is difficult to train neural networks to solve problems that involve geometry. One potential direction for future work could be to use 3D transpose convolutions. Using 2D convolutions is helpful to preserve x, y locality, but this problem has θ locality as well.

[1]For the KL objective, given the computational cost, we precalculate an offline dataset of only 610 examples.

Figure 2: Qualitative regression results. Scan centered around ground truth pose.

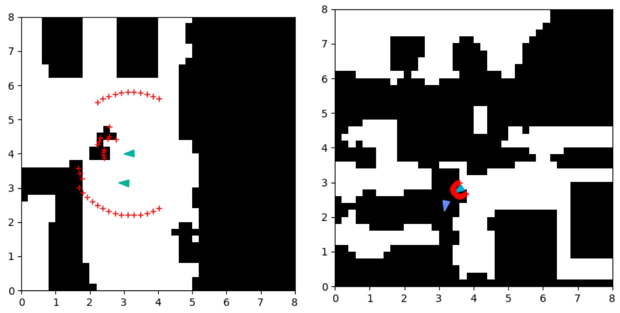


Figure 3: Qualitative classification results. Bayes prediction on top, neural net bottom. Poses are represented as triangles. Hue encodes θ, and alpha encodes the predicted weight of the pose. The scan is represented by red "+" marks centered around the most likely pose. **Left:** CE objective. **Center:** KL objective. **Right:** CE objective, x, y only.

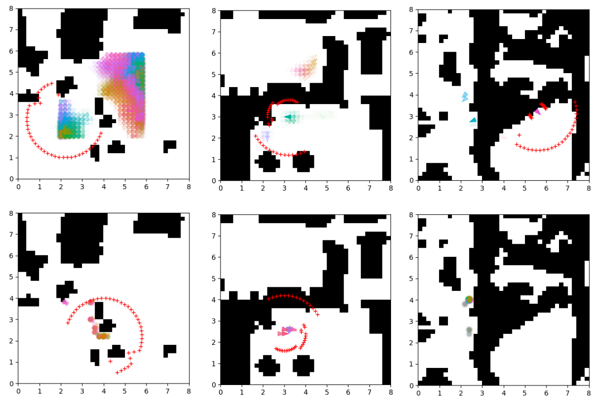


Table 1: Quantitative Results

	Initial loss	Final loss (1000 epochs)
Regressor (MSE)	45.04	19.73
Classifier (CE)	8.60	5.94
Classifier, x, y only (CE)	5.43	1.86
Classifier (KL)	4.02	2.10

Another strategy to improve performance could be to change the representation of the laser scan. Rather than using a 1D vector of depths, we could render the laser beams into a 2D occupancy grid. This would have the advantage that the model might generalize to other laser scanners with differing angular resolutions.

In general, we found that mixing the laser and occupancy data earlier in the network improved performance, as did using deeper networks with more skip connections. Continuing along these lines, we might discover an architecture that solves localization more quickly than, and as accurately as, the Bayesian approach.

REFERENCES

[1] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. (2014). arXiv:cs.LG/1412.6980

A MORE QUALITATIVE RESULTS

Here are some more results from our best-performing model, the classifier which predicts only (x, y) location. As in Figure 3, the top row shows the Bayesian prediction, and the bottom row shows the neural network prediction. The bottom-right shows an extreme failure case.