Neural Networks for Obstacle Avoidance

Joseph Djugash
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
josephad@andrew.cmu.edu

Bradley Hamner
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
bhamner@andrew.cmu.edu

Abstract

Learning a set of rules to perform reliable obstacle avoidance has been proven to be a difficult task. In this paper, a neural network learned from human driving data is introduced to model obstacle avoidance through dense areas of obstacles. The learned neural network is then tested on different scenarios and compared using cross-validation to determine the optimal network structure (number of nodes and layers). The results are also compared with another obstacle avoidance approach, which acts as a basis for comparison.

1 Introduction

Collision avoidance is a common problem in robotics, and many methods have been employed to solve it. Most common approaches derive their models from intuition and ”magic” parameters. Neural networks present a solution, different from classical models, that is able to learn complex solutions to obstacle avoidance. Most existing approaches that make use of neural networks tend to either only deal with scenarios with very few obstacles or with path following implementations where a direct path to the goal is always available. In this paper, we propose a neural network implementation that performs reliable obstacle avoidance in dense obstacle environments.

The proposed approach takes as inputs the data from a human driving in a dense obstacle environment and the goal points that were given to the human driver and produces a angular velocity used to drive the vehicle. In order to fully utilize and neural network’s capability we investigate changing the number of layers and nodes in the network. Cross validation is used to systematically evaluate the results produced by these different networks and to find the best model that produces accurate and computationally efficient results.

2 Related Work

Traditional obstacle avoidance methods use smooth models which are easily understandable by scientists. The lane-curvature method [1] and dynamic window [2] are reactive methods which search the space of admissable vehicle translational and rotational velocities to maximize a function based on goal following and obstacle clearance. VFH chooses the direction for the robot which maximizes obstacle clearance [3]. In [4] and [**?**], Huang, Fajen, and Warren present a high-dimensional model which controls the robot using the

gradient of a potential field in steering angle space.

All of the above methods rely on well-formed functions to properly control the vehicle around obstacles. Typically these functions are tuned by the researchers until the robot's behavior is considered sufficient. In recent work, Hamner, Scherer, and Singh present a method to learn the parameters of Fajen and Warren's control model by matching the robot's behavior to the human-driven paths [**?**]. We will use the same data to learn our neural network and consider their results as a benchmark to compare our results.

Conversely, neural networks have been applied in various ways to the task of collision avoidance. Hiraga, et al., use neural networks to avoid collisions between two ships passing each other [**?**]. Similarly, Durand, et al., learn a neural network for air traffic control [10]. In [9], Graf and Lalonde use a neural network to represent valid configurations of a robot manipulator.

3 Method

We now present our method: the data collection process, the design of the neural network, and the training method to produce a robot controller.

3.1 Human Data

During the data collection process, the human driver was presented with a display of the vehicle's status. This included the goal point and obstacles relative to the vehicle's location. The driver was instructed not to look up at the road ahead, but to look only at the on-board monitor. In this way we could ensure the human knew only what the system (and our neural network) would know.

While the human was driving, we recorded the vehicle's position and heading, its translation velocity, and its rotational velocity. We also recorded the goal point he was given and the detected obstacles.

3.2 Network Design

We chose a simple feed-forward model for the neural network. The goal and obstacle data are the inputs, and the output is an angular velocity command.

The goal is represented as two inputs, its distance and angle to the robot. To represent the obstacles, we discretize the area in front of the robot into wedges, as shown in Figure 1. Each wedge is five degrees. For each wedge, we find the nearest obstacle to the vehicle. Then the input for that wedge is $10 - d$, where d is the distance to the nearest obstacle in meters. Obstacles are not detected further than 10 meters away because of laser scanner limitations, so that is a safe cutoff. If there is no obstacle in the wedge, its input is 0.

We had considered a more intuitive grid notation for the obstacles, where the area in front of the robot is represented by a discretized map. Each input is then a 0 or 1 indicating the presence of an obstacle in the corresponding map cell. This representation is more powerful, but we thought this might provide too much information to the network and lead to overfitting. To reduce the dimensionality of the input space, we use the wedge representation, though we may revisit the grid notation if we have time.

For the hidden layers we use sigmoid units. The output is an angular velocity command, so we must allow negative values. Also, we consider the shape of the sigmoid and $tansig$ to be undesirable for this purpose, since small changes in the input to the node can produce large variance in the node's value. Therefore, we consider a linear function to be more suited to the purpose of the output node.

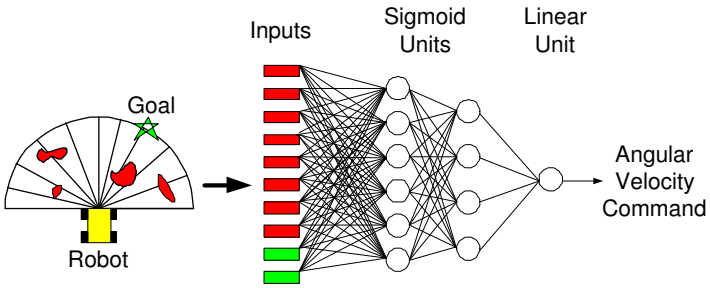

Figure 1: Illustration of the inputs and output of a neural network in our form with two hidden layers. The obstacles in front of the robot are input as their distance in their corresponding wedge. The goal star is input as its distance and angle to the robot. The sigmoid units are combined into a linear unit which represents the steering angle command.

(a) (b) (c)

Figure 2: (a) The time sequence of the angular velocity training. After minimization, the network (green) matches the training sequence (blue). Magenta is the angular velocity sequence of the initial network. (b) After 100 iterations, the neural network (black) simulates a path similar to the human's (blue) (c) After 200 iterations, the network is overfitted to the angular velocity sequence and takes a bad route.

3.3 Training

We cut the human driving data into short segments on which to train the network. For every data point, we apply the network given the goal and obstacle information the driver received at that time. The network produces an angular velocity command. The target is the recorded angular velocity of the vehicle for that instant.

We use Matlab's neural network toolbox to learn the weights. It minimizes the error between the target angular velocities and the produced angular velocities using gradient descent backpropagation with momentum.

4 Results

Figure 2 shows the training error of a network with one hidden layer trained on a small data segment. The network has properly learned how to produce steering angles that match the targets (Figure 2-(a)).

However, notice the paths in Figure 2-(b) and -(c). The blue paths show the simulated result of a vehicle driving in that region using our neural network to produce angular velocity commands. After 100 iterations, the network has learned a reasonable concept of obstacle avoidance for this situation. After 200 iterations, though, the network is overfitted to the

target steering angles. It can reconstruct the steering angles quite well, but does not steer the vehicle in reasonable directions.

The problem is that the network has no real model of the situation presented to it. It does not know it is supposed to find a way to avoid obstacles, only reproduce steering angles. To reduce the tendency to overfit, we must train the network with many more examples in future experiments.

5 Conclusions

Under construction. Forgive our mess.

6 Plan of Activities

6.1 The Plan

November 8

- Decide on a form of network (nodes, inputs, etc.)
- Have a method to produce the neural network for a given size
- Have a network learning how to drive given human data

Research done jointly. Joe does network building. Brad formulates the error metric. Joint work on training.

November 17

- Cross validation to decide which size network is best
- Analyze computation speed for the different sizes

Cross validation to be done jointly.

6.2 Status

We have designed and built the network and trained it on a few sample cases. It now produces reasonable paths for obstacle avoidance. Therefore, we consider our November 8 goal complete.

Joe worked on the network construction in Matlab. Brad worked on the parsing of human driving data into training segments and the error calculation. We worked together on the rest.

The goal for the next week is refinement of our procedure and cross validation experiments.

References

[1] R. Simmons, "The curvature-velocity method for local obstacle avoidance," In IEEE International Conference on Robotics and Automation, volume 4, pages 2275-2282, 1996.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," IEEE Robotics and Automation Magazine, 4(1):23 V33, March, 1997.

[3] I. Ulrich and J. Borenstein, ”VFH*: Local obstacle avoidance with look-ahead verification,” IEEE International Conference on Robotics and Automation, pages 2505 V2511, 2000.

[4] B. Fajen and W. Warren, ”Behavioral Dynamics of Steering, Obstacle Avoidance, and Route Selection,” Journal of Experimental Psychology: Human Perception and Performance, Vol. 29, No. 2, 2003.

[5] I. Hiraga, T. Furuhasi, Y. Uchikawa, ans S. Nakayama, ”An Acquisition of Operator’s Rules for Collision Avoidance Using Fuzzy Neural Networks”, IEEE Transactions on Fuzzy Systems, Vol. 3, No. 3, August 1995.

[6] M. Meng and A.C. Kak, ”Mobile Robot Navigation Using Neural Networks and Nonmetrical Environmental Models,” IEEE Control Systems, 1993.

[7] D. Pomerleau, ”Progress in Neural Network-based Vision for Autonomous Robot Driving,” Proceedings of the 1992 Intelligent Vehicles Conference, June, 1992, pp. 391-396.

[8] Z. Mao and T.C. Hsia, ”Obstacle Avoidance Inverse Kinematics Solution of Redundant Robots by Neural Networks,” Robotica, 1997.

[9] D. Graf and W. LaLonde, ”A Neural Controller for Collision-Free Movement of General Robot Manipulators,” IEEE International Conference on Neural Networks, 1988.

[10] N. Durand, J. Alliot, and J. Noailles, ”Collision Avoidance using Neural Networks Learned by Genetic Algorithms,” IEA-AEI, 1996.

[11] B. Kröse and J. van Dam, ”Adaptive State Space Quantisation for Reinforcement Learning of Collision-Free Navigation,” IEEE International Conference on Intelligent Robots and Systems, 1992.

[12] R. Glasius, A. Komoda, and S. Gielen, ”Neural Network Dynamics for Path Planning and Obstacle Avoidance,” Neural Networks, March 1994.