Deep Turtle Control

Michael Farrell, Skyler Tolman

***Abstract*— abstract here**

I. Introduction

II. Classical Approach

III. Network Approach

The end goal of our project was to get the turtlebot to follow a track purely using an image input to a neural network. This section describes the architecture and training methods used to train a control network for the turtlebot.

A. Network architecture

We experimented with multiple architectures, including a simple convolutional neural network (CNN), and a ResNet [**?**], but we eventually decided to use the DenseNet as our base architecture [**?**].

The Densenet is an network architecture with densely connect convolutional blocks, as seen in Figure **??**. The DenseNet architecture is unique because of the densely connected skip connections across multiple convolutional blocks, which allows the architecture to re-use features across mutliple blocks. Accordingly, it has achieved better results on benchmarks such as ImageNet with a smaller memory footprint.

We used a pre-trained model for the DenseNet convolutional blocks, and added three linear layers with dropout to train the turtlebot control using the DenseNet features.

B. Network Training

We used supervised "imitation" learning to train the control network. We collected data by driving the turtlebot around multiple tracks on different ground surfaces and matched RGB camera images to angular velocity command outputs. We initially used the classical vision control method to gather data on a single track, then diversivied the data by manually driving the turtlebot around other varied tracks.

Using the image-command pairs, we trained the neural network to predict the appropriate command for each input image. We trained the network on about 7 epochs of 15,000 images.

C. Continous Control

Ideally, we would like the network to be able to predict a continous control output given an input image. We allowed the network output to range from -1 to 1 radians per second. A simple way to limit the control effort was to use a hyperbolic tangent function as the final activation on our linear layer backend.

After multiple attempts at training the network to predict a continouous output, we were able to get some success, but the turtlebot had a hard time making sharper turns and would often drive straight off the course instead of making the turns.

IV. Results

V. Conclusion