Modeling a Transformable Wheel Mobile Robot With a Simulator Neural Network

Anthony J. Clark

Missouri State University, Springfield, MO 65804
anthonyclark@missouristate.edu

Abstract

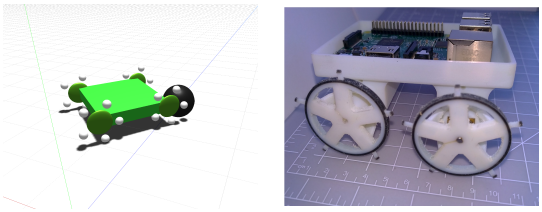
It is difficult to model kinematics of a legged-wheel robot; primarily due to the complex interactions between their irregularly shaped wheels and the ground. Yet, for many applications such models are vital. For example, to predict the current velocity of the robot. We propose using a simulator neural network (SNN) to model the kinematics of a transformable wheel mobile robot. We use a physical simulation of our robot to generate training data, which we in turn use to optimize an SNN that can predict changes to the robot's pose based on its current control commands. The SNN is better able to predict the location of the physically simulated mobile robot when compared to a differential drive model. Using the trained network, we next evolve a simple controller to navigate a series of way-points. Finally, we transferred the evolved control parameters to the physically simulated robot where we see nearly identical behaviors to those of the SNN. Our results show that an SNN can be effective for predicting the movement of a transformable wheel mobile robot.

Introduction

Robots are frequently used in remote and unpredictable environments. For example, in *search and rescue* robots are designed to aid first responders in searching for victims in hazardous and varied terrain (Graf et al., 2017). One solution to this problem is to use an unmanned aerial vehicle (UAV). However, UAVs only operate for short periods of time (typically less than one hour).

More recently, lightweight mobile robots with transformable wheels have been developed for search and rescue. As shown in Figure 1, a transformable wheel robot is capable of extending struts radially outward from the center of each wheel. These struts help the robot climb over obstacles that are roughly the same size as the robot (Clark et al., 2018). These devices have the benefits of normal wheels (e.g., increased stability and less vibration) and legged-wheels (e.g., increased mobility), and they have a longer operating duration compared to UAVs.

A drawback of using a transformable wheel mobile robot (or a legged-wheel robot) is that they are difficult to model. Without an accurate model of the robot, it is difficult to determine when wheels should transform from normal wheels into legged-wheels. Specifically, a model can calculate the *expected* velocity of the robot, and we can compare this quantity to the *real* velocity as measured by sensors. If these two quantities differ by a specified threshold, then the robot should infer that it is stuck or slipping (i.e., it is exhibiting poor mobility) and extend its wheel struts. Without an accurate model of the robot kinematics, this process cannot work.



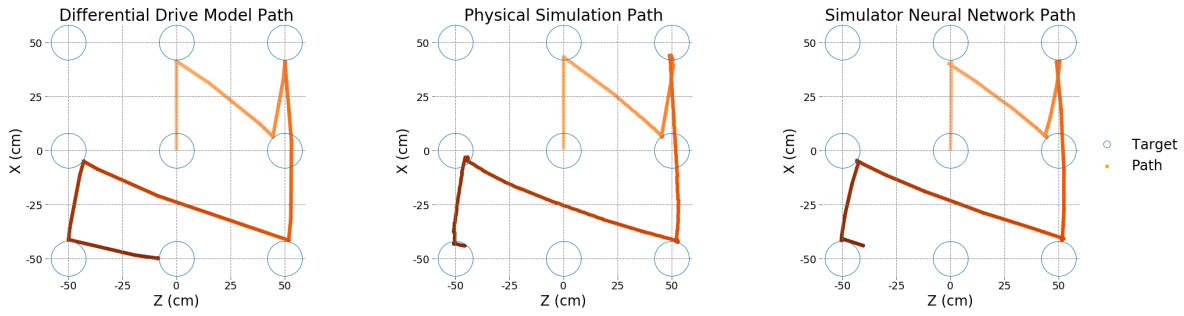
(a) Physical Simulation (b) Prototype

Figure 1: A mobile robot with transformable wheels.

Related work. In prior work, we use a differential drive model to calculate *expected* velocity (Clark et al., 2018). This process was error prone. The model did not account for strut extension, wheel slippage (i.e., differing friction properties of different terrains), uneven ground, or that the wheels might rotate at a different rate than commanded. In this study, we develop a simulator neural network (SNN) similar to that described by Pretorius et al. (2014), where a neural network is trained to map control signals to the pose of a robot. The goal of this study is to produce an accurate model of our mobile robot. The model will have two uses: (1) to act in place of a physical simulation for optimizing control parameters and (2) to determine when the robot exhibits poor mobility and should extend struts.

Training and Testing an SNN

The goal of our study is to develop an SNN that can determine a new pose for our mobile robot based on the input control signals. To train the neural network, we first needed to collect training data.



(a) (b) (c)

Figure 2: Example paths taken by the robot (a) as predicted by the differential drive model, (b) during a physical simulation, and (c) as predicted by a trained SNN. Blue circles represent way-points that the robot is trying to reach in a pre-defined sequence starting at the origin. The robot moves on to its next way-point once it is within 10cm of its current. Robot paths are shown in orange, and lighter shades denote an earlier time in simulation.

Collecting training data. We built a physical simulation of our robot using DART (Lee et al., 2018). We can run the simulation with different values for the robot's input control signals: the speeds of the left and right wheels and the strut extension amount. The simulation did not include any obstacles or uneven terrain. We ran the simulation with 9012 different combinations of these input signals and collected the resulting change in position and heading.

Training the SNN. The SNN comprises three neural networks, one each for predicting changes to the robots longitudinal position, lateral position, and heading. We evaluated several neural network hyper-parameter combinations (i.e., architectures, optimization algorithms, learn rates, etc.). We achieved the highest accuracy on our validation data when using a network with two hidden layers, each with 20 nodes, and the L-BFGS optimizer with an adaptive learning rate.

Comparing the SNN model. Figure 2 shows paths of a robot as dictated by the differential drive model, the physical simulation, and our trained neural networks. As shown in the figure, compared to the kinematics model the SNN more closely resembles the actual path of the simulated robot. Other example paths (using different control parameters) showed similar relative performances between the SNN and the differential drive model. The SNN outperforms the differential drive model because it accounts for the wheel extensions and slippage. Since the SNN takes into account these physical properties and limitations, and the real world has even more sources of noise and external influence, we expect the relative accuracy of the SNN to be even higher when these experiments are repeated with our physical device.

Evolving a simple controller. As the SNN is an accurate model of the physical simulation, we next attempted to use the SNN to optimize a simple controller for performing the way-point following task (as shown in Figure 2). An advantage of using the SNN is that it is approximately 30 times faster than the physical simulation. Thus on a standard laptop, optimizing a simple finite state machine with differential evolution took only two minutes, whereas it would have taken one hour using our physical simulation. An interactive visualization of the way-point following robot can be found at this address: `http://bit.ly/2ChVuqr`.

Conclusions. We have two goals for the SNN: (1) to act in place of a physical simulation and (2) to detect poor mobility and dictate when wheel struts should extend. In this study, we demonstrated goal (1) by evolving the parameters of a simple controller for way-point following. As part of our ongoing studies, we will use the trained SNN to detect poor mobility when the robot is operating in an environment with obstacles and uneven terrain, and we will perform similar experiments on our real device. All code for this study can be found in this repository: `https://github.com/anthonyjclark/adabot-snn/`.

References

Clark, A. J., Cissell, K. A., and Moore, J. M. (2018). Evolving Controllers for a Transformable Wheel Mobile Robot. *Complexity*, 2018:1–12.

Graf, M., Poy, M., Bischof, S., Dornberger, R., and Hanne, T. (2017). Rescue path optimization using ant colony systems. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7.

Lee, J., X. Grey, M., Ha, S., Kunz, T., Jain, S., Ye, Y., S. Srinivasa, S., Stilman, M., and Karen Liu, C. (2018). DART: Dynamic Animation and Robotics Toolkit. *The Journal of Open Source Software*, 3(22):500.

Pretorius, C. J., du Plessis, M. C., and Gonsalves, J. W. (2014). A comparison of neural networks and physics models as motion simulators for simple robotic evolution. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2793–2800, Beijing, China. IEEE.