Using Neural Networks to Learn Quadruped Leg Models

Tarun Punnoose

CS 229 Final Project

1 Introduction

The Stanford Robotics Club has designed and built three highly capable quadruped robots: Doggo, Woofer, and Pupper. These robots robots currently operate in an open loop manner with regard to control inputs. In order to close the loop around the system, an accurate estimation of system state must be given to the robot. An important part of knowing the state of the robot is knowing which feet are in contact at a given time and the approximate contact force.

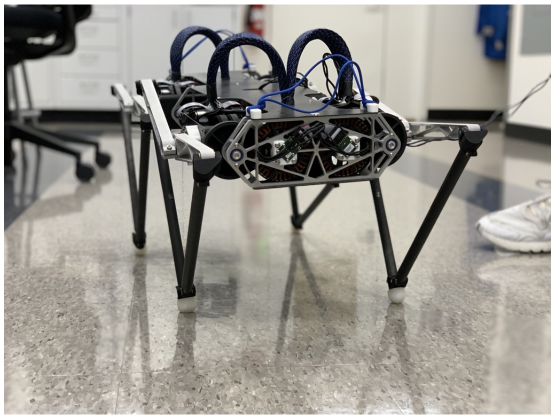

Figure 1: Woofer Quadruped Robot

Force sensors are expensive, unreliable, and subject to difficult space constraints at the end of the legs. Because Woofer and Doggo have Quasi Direct Drive (QDD) transmissions, this makes getting proprioceptive contact force measurements possible. In order to do this, the traditional approach involves using physics based models for the legs. The most common approach for this is the generalized momentum observer that uses residual dynamics and a control law to drive the error in the estimated external torque on the system to zero [2]. The MIT Cheetah robot employs an extended version of this generalized momentum observer on their robot to identify contact modes of the robot [1]. However, these approaches require having accurate models of the the system and often don't characterize complexities like friction very well. Although some researchers employ Least Squares approaches to find system parameters, a machine learning algorithm could be more expressive and accurate [3].

Training a machine learning algorithm to predict contact force could take into account friction in the system and provide an estimate with higher accuracy. This would allow the general state estimator to have a better estimate of the overall state of the quadruped. With an accurate contact force, the contact state for each of the legs and the force could be fed to a traditional state estimator like a Kalman filter to update the state.

2 Problem Formulation

The dynamics of the legs can be written with the general manipulator equation:

$$\begin{aligned} \tau + J^T F_{ext} &= D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) \\ &= \tau_{free} \end{aligned} \tag{1}$$

This means that the external force on the system can be written as:

$$F_{ext} = J^{-T}\left(\tau_{free} - \tau\right) \tag{2}$$

$$\tau_{free} = f\left(q, \dot{q}, \ddot{q}\right) \tag{3}$$

where f is a non-linear mapping between the joint positions, velocities, and accelerations and the inertia of the system. This free torque represents the torque necessary to have the leg moving at q, $\dot{q}$, and $\ddot{q}$. Because the Jacobian of the leg can be found with relatively simple kinematics, with the free torque and the torque from the actuators the estimated contact force can easily be found. Based off of this, the machine learning algorithm needs to approximate the mapping from joint positions, velocities, and accelerations to the free torque. This is the same approach that Smith et al. took to estimate contact forces with neural networks. [4]

3 Data Collection

Although the ultimate goal is to train on real data from the physical robot, getting accurate force sensor data for model validation is difficult and expensive. For this reason, all the data for this project has been collected inside of the MuJoCo physics simulator. MuJoCo is fast and able to model contact relatively accurately which make it the natural choice for modeling robotic systems like quadrupeds. MuJoCo also provides force sensors, torques sensors, and joint position/velocity sensors which are the inputs to the neural network model. By

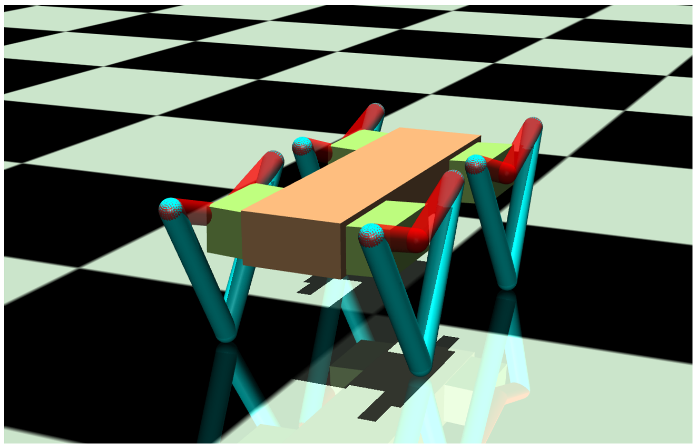

Figure 2: MuJoCo Model of Woofer

using MuJoCo, it is possible to collect training data easily in the simulator and also test in the simulator with relative ease.

In order to learn the dynamics of the leg, the leg was controlled with a simple PD loop with two different random inputs: a random walk trajectory inside an envelope of the leg and a random position for the leg. These two types of random trajectories provide a rich enough dataset to describe the inertial properties of the leg.

4 Models

4.1 Feedforward Neural Network

Because the physical robot only joint position and velocity information, the first network that was tried has only these as the input. This neural network structure is shown in Figure 3. However fundamentally, q and $\dot{q}$ don't provide

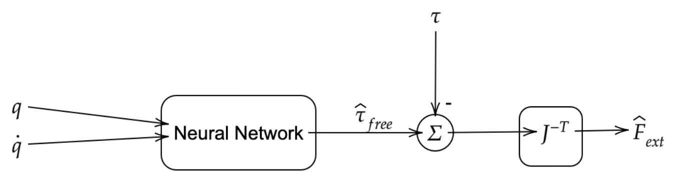

Figure 3: Feedforward Neural Network

necessary information about $\ddot{q}$. To resolve this a history of joint positions and velocities are given to the neural network in order to give some sort of information about $\ddot{q}$.

4.2 Recurrent Neural Network

Instead of just collecting joint information histories and passing that to the neural network, applying a recurrent neural network (RNN) would allow the model to keep track of a hidden state between time steps. This hidden state would allow the model to have information about the previous joint positions and velocities. Similar to using a traditional recursive state estimator like a Kalman filter, using an RNN would provide convergence properties and additional stability to the model.

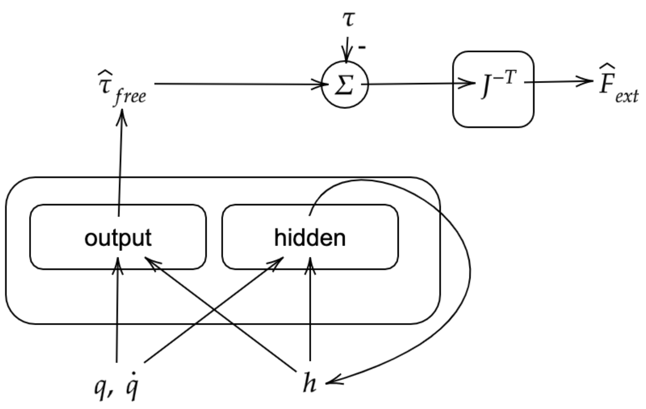

Figure 4: Recurrent Neural Network

Figure 4 shows a diagram of the proposed RNN model layout. Both the output and hidden blocks in the diagram represent fully connected networks that use the softsign activation function.

5 Results

5.1 Feedforward Neural Network

model activation	input	MSE
linear	$q, \dot{q}$	5.481
softsign	$q, \dot{q}$	4.004
softsign	$q, \dot{q}$ history	0.317
sigmoid	$q, \dot{q}$ history	0.447

Table 1: Feedforward Network Comparison

In Table 1, a plot of all the different mean squared errors (MSE) for the test set is shown. As explained in the previous section, the model needs to have information about $\ddot{q}$ in order to be able to predict the torques. Although

Smith et al. include a network without $\ddot{q}$ that has a low MSE, fundamentally information about the joint acceleration is necessary to predict joint torques.

As shown in the results, the addition of joint position and velocity history was necessary in order to be able to predict torque from joint position and velocity. The networks used had 2 hidden layers with 100 and 50 nodes. The softsign activation function seemed to have the best predictive ability.

5.2 Recurrent Neural Network

Training the RNN proved to be a significant challenge with to accomplish accurate predictions of $\hat{\tau}_{free}$. Choosing when to truncate the forward propogation of the hidden states and the backward propogation adds to the number of possible parameters in the network. More work is necessary in order to demonstrate the predictive ability of this network for this problem.

6 Conclusion

A forward neural network that used joint position and velocity histories was able to predict torques to a reasonable accuracy. Although the predictive ability of an RNN has not been demonstrated yet on this problem, the structure of an RNN seems promising and will be worked on in the future. Future implementations will also have all the training data must be taken from real measurements in order to translate to hardware. All code for the project can be found on Github: https://github.com/tpunnoose/WooferML.

References

[1] Gerardo Bledt, Patrick M Wensing, Sam Ingersoll, and Sangbae Kim. Contact model fusion for event-based locomotion in unstructured terrains. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[2] Alessandro De Luca and Raffaella Mattone. Sensorless robot collision detection and hybrid force/motion control. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 999–1004. IEEE, 2005.

[3] Enrique del Sol, Ryan King, Robin Scott, and Manuel Ferre. External force estimation for teleoperation based on proprioceptive sensors. *International Journal of Advanced Robotic Systems*, 11(3):52, 2014.

[4] Andrew C Smith, Farid Mobasser, and Keyvan Hashtrudi-Zaad. Neural-network-based contact force observers for haptic applications. *IEEE Transactions on Robotics*, 22(6):1163–1175, 2006.