Fido: A Universal Robot Control System Using Reinforcement Learning with Limited Feedback

Joshua Gruenstein   Michael Truell

Control System Objectives

Fido was created to fulfill the following goals:

- **Trainability**: Allow both human and autonomous training rather than reprogramming
- **Universality**: Run on any robot, even without prior knowledge of the host

These goals were achieved through the training of artificial neural networks with a wire-fitted moving least squares interpolator following the Q-learning reinforcement algorithm and an action selection policy that utilizes a Boltzmann distribution of probability.

Implementation

Fido was programmed in C++, with no external dependencies. However, the simulator does use the SFML graphics library. The hardware implementation uses the Intel Edison embedded platform, a 3D printed chassis and a differential drive system.

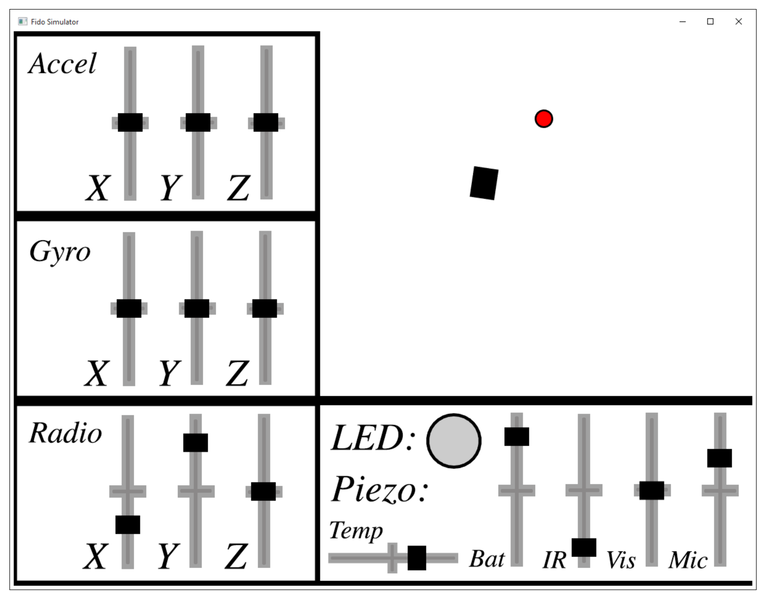

Figure 1: Fido Simulator Graphical User Interface

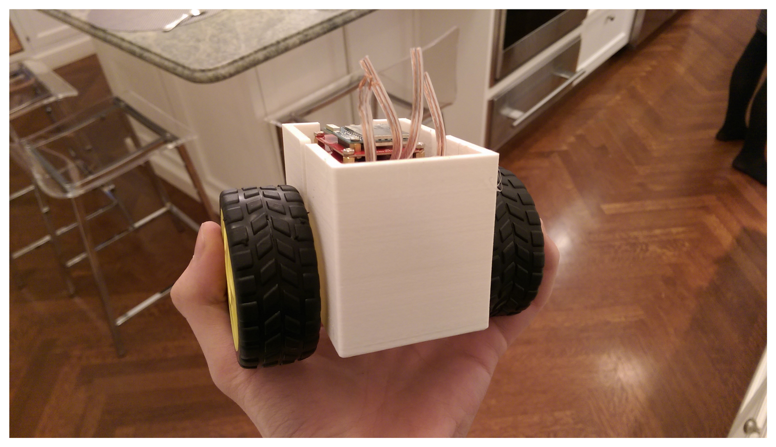

Figure 2: Fido Hardware Implementation

Learning Algorithm

- From a macro perspective, Fido can be viewed as a "black box:" inputs go in and outputs go out, Fido must optimize the relationship of inputs to outputs to maximize reward
- **Reward system:** Trying to determine the expected reward for an action in a given state based on past reward received
  - Must have a scalable, performance-optimized way of storing past state-reward sets and detecting patterns

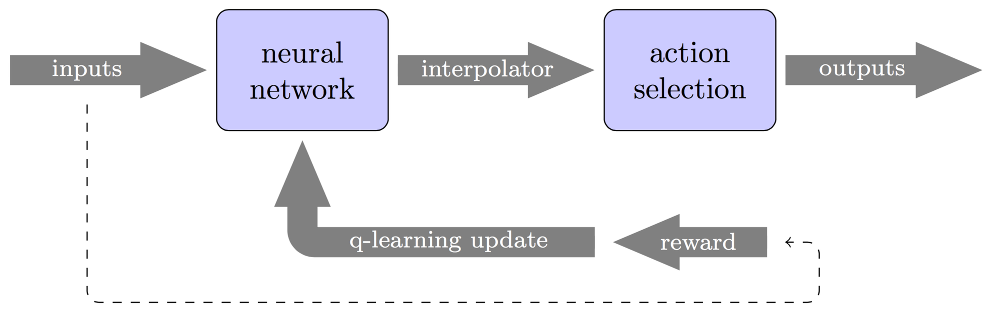

Figure 3: Control System Diagram

Artificial Neural Networks

- Function approximators modeled after nature with the capability to take in a large number of inputs, parallelly process them, and produce a set of outputs

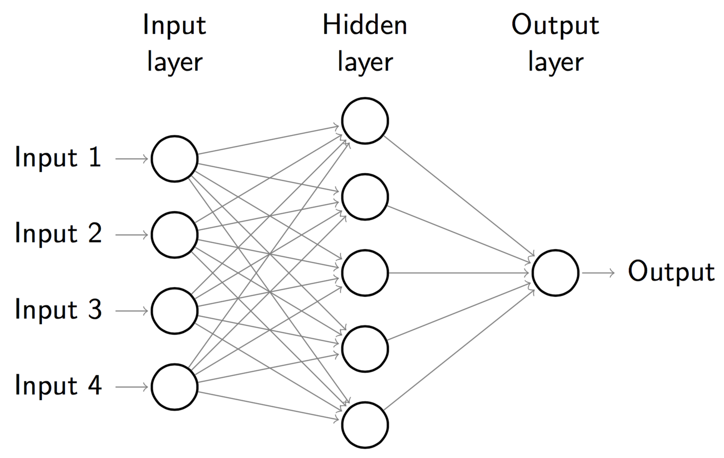

Figure 4: Single Output Feed-forward Neural Network

Reinforcement Learning

- **Q-Learning:** Develops a function that intakes a state-action pair and outputs expected utility
- Ordinarily, the Q-function is modeled by storing state-action pairs in a table
  - Is impractical for large state spaces, use a function approximator instead: **Artificial Neural Networks**
- Usually discrete: no relation made between states or actions
  - Can be optimized by coupling a wire-fitted interpolator with our neural network (**Wire-Fitted Q-Learning**)
- Cannot just pick the action with the greatest expected utility: must "explore" to be trainable and re-trainable
  - Use a **Boltzmann Probability Distribution** selection policy

Results

Results were gathered both from simulation and hardware for a variety of tasks. Gathered data for each task included the number of learning iterations, or how many pieces of reward it took for Fido to master the task, action selection time, or the latency in probabalistic action selection, and training time, or the latency in updating Fido's model.

Table 1: Fido Results in Simulation

Task	Learning Iterations	Action Selection (ms)	Training Time (ms)
Flash	6	0.	6
Float to Point	14	1	6
Drive to Point	17	1	11
Line Following	18	0.	2
Noisy Line Following	21	0.	105

Table 2: Fido Results on Thing One

Task	Learning Iterations	Action Selection (ms)	Training Time (ms)
Stay Still	3	1	43.5
Drive to Point	18	4	65

Table 3: Fido Results on Thing Two

Task	Learning Iterations	Action Selection (ms)	Training Time (ms)
Drive Straight	13	2	30
Line Following	15	21	95
Fetch	8	1	70
Limping Line Following	6	20	37

Future Development

We would like to experiment with dynamic optimization of hyperparameters, changing factors such as neural network architecture and Boltzman temperature constant to best fit the task at hand. We also plan to package Fido as a machine learning library for embedded electronics and robotics, and build a microcontroller-based hardware implementation to further explore for resource-limited environments.

References

[1] C. Gaskett, D. Wettergreen, and A. Zelinsky, "Q-learning in continuous state and action spaces," pp. 417–428, 1999.

[2] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," 1974.

[3] S. Dini and M. Serrano, "Combining q-learning with artificial neural networks in an adaptive light seeking robot," 2012.

[4] C. MacLeod, "An introduction to practical neural networks and genetic algorithms for engineers and scientists," Robert Gordon University, Tech. Rep., 2010.

[5] D. S. Kim and A. J. Papagelis, "Multi-layer perceptron: Artificial neural networks," http://www.cse.unsw.edu.au/~cs9417ml/MLP2/.

[6] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. New York, NY, USA: Cambridge University Press, 2000.

[7] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge England, 1989.

[8] G. A. Rummery, "Problem solving with reinforcement learning," Ph.D. dissertation, University of Cambridge Ph. D. dissertation, 1995.

[9] L. C. Baird and A. H. Klopf, "Reinforcement learning with high-dimensional, continuous actions," *Wright Laboratory, Wright-Patterson Air Force Base, Tech. Rep. WL-TR-93-1147*, 1993.

[10] G. Biggs and B. MacDonald, "A survey of robot programming systems," pp. 1–3, 2003.