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

Joshua Gruenstein Michael Truell
Horace Mann School

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.

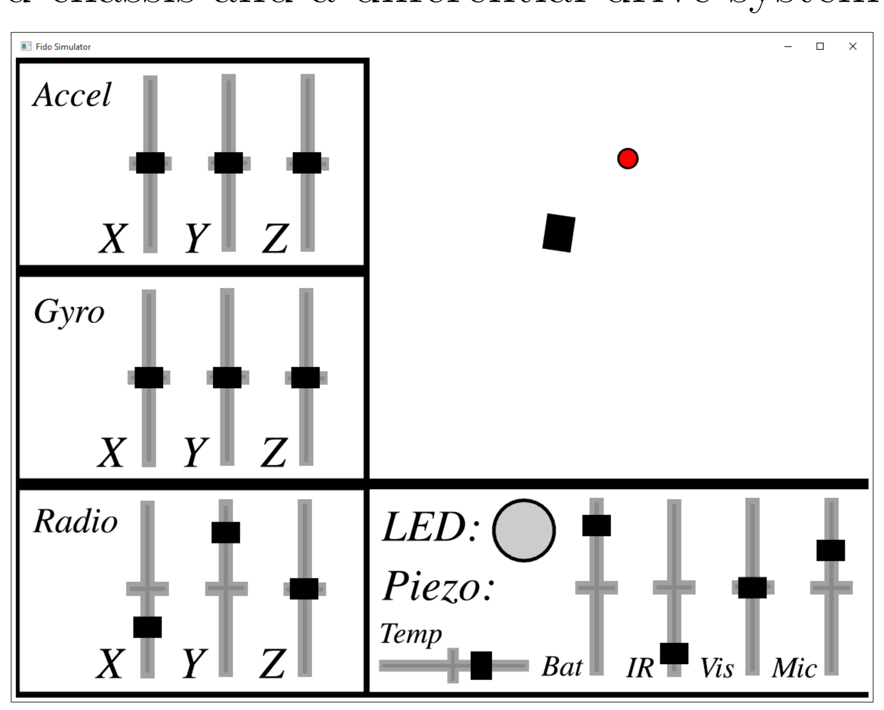

Figure 1: Fido Simulator Graphical User Interface

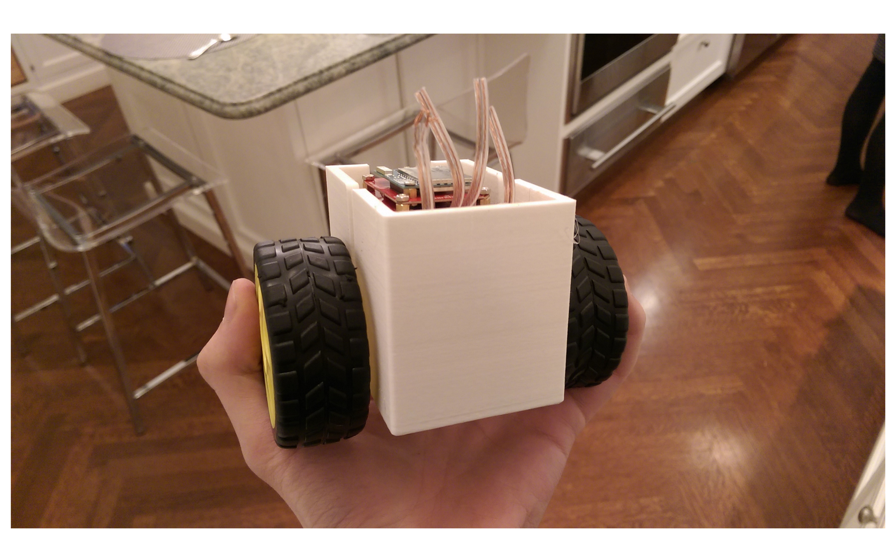

Figure 2: Fido Hardware Implementation

Learning Algorithm

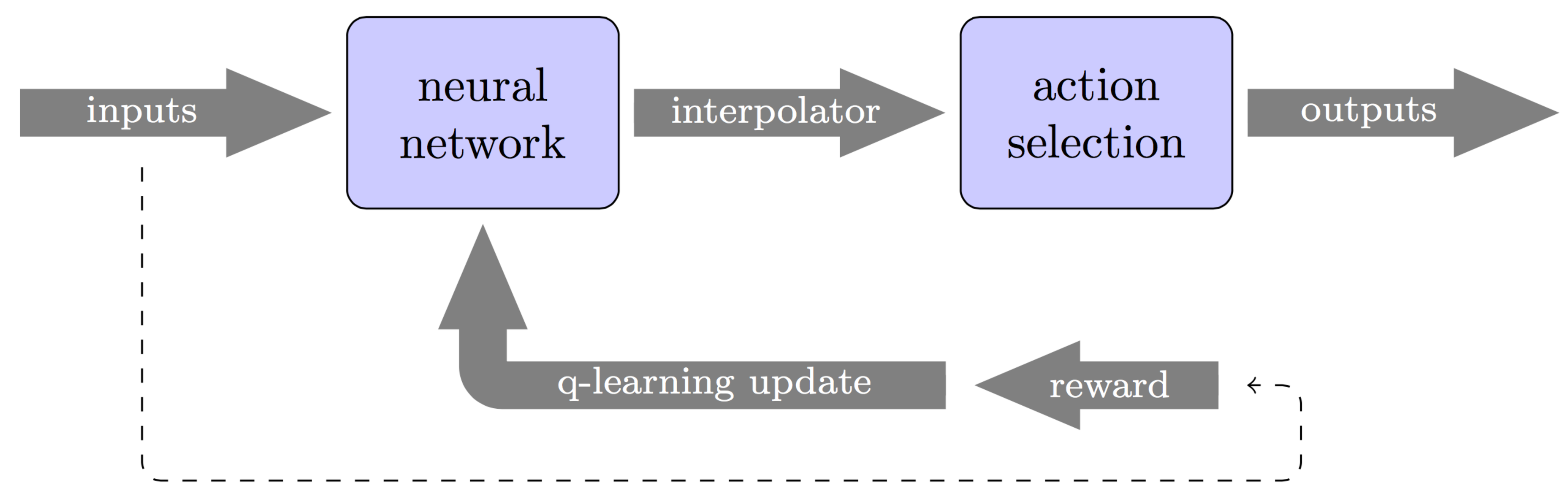

Figure 3: Control System Diagram

The following materials were required to complete the research:

- Curabitur pellentesque dignissim
- Eu facilisis est tempus quis
- Duis porta consequat lorem
- Eu facilisis est tempus quis

The materials were prepared according to the steps outlined below:

1. Curabitur pellentesque dignissim
2. Eu facilisis est tempus quis
3. Duis porta consequat lorem
4. Curabitur pellentesque dignissim

Reinforcement Learning

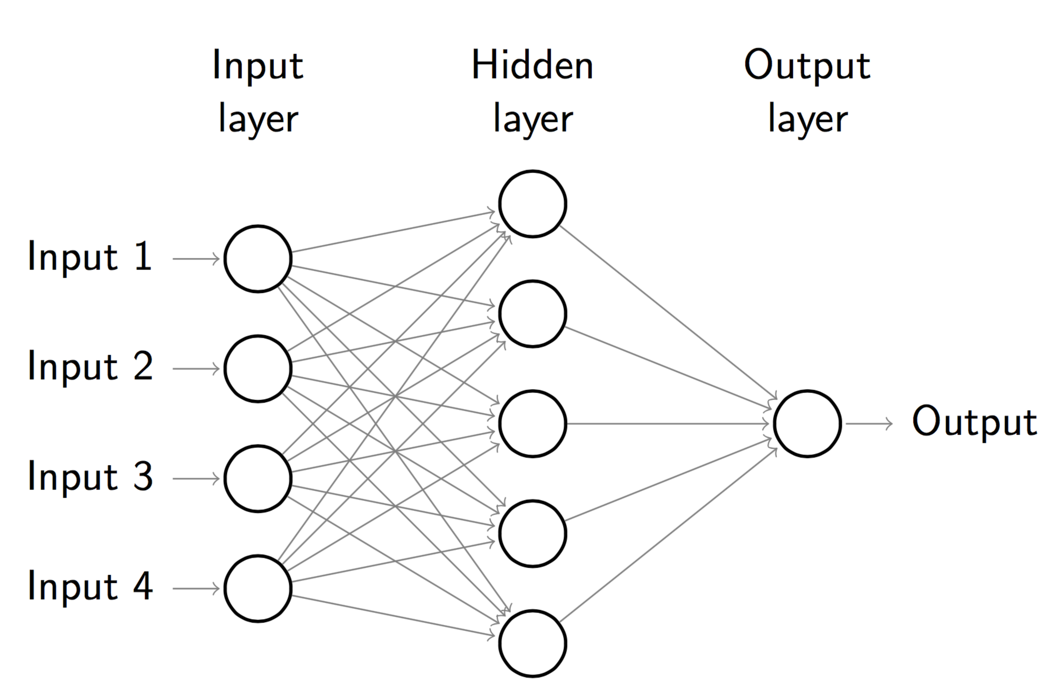

Figure 4: Single Output Feed-forward Neural Network

Action Selection Policy

Boltzmann stuff

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 optimize 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.

Acknowledgements

Thank you to Dr. Jeff Weitz of Horace Mann School, who gave us guidance in the art of scientific research.

Results