Fast Connectionist Learning for Trailer Backing using a Real Robot *

Dean F. Hougen, John Fischer, Maria Gini, and James Slagle
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

Abstract

This paper presents the application of a connectionist control-learning system to an autonomous mini-robot. The system's design is severely constrained by the computing power and memory available on board the mini-robot and the on-board training time is greatly limited by the short life of the battery. The system is capable of rapid unsupervised learning of output responses in temporal domains through the use of eligibility traces and data sharing within topologically defined neighborhoods.

1 Introduction

Control-learning in autonomous robotic systems provides many challenges. The learning system must be robust enough to overcome the problems of noisy input data and uncertain interactions between motor commands and effects in the world, be compact enough to fit into available on-board memory, and be able to give responses in real time. The challenges are even greater if the learning system must acquire its proficiency during a short demonstration period consisting of a small number of trials. The system we present here meets these specifications, yet is capable of unsupervised learning of a difficult credit-assignment problem.

Connectionist control-learning systems have recently received much attention, and numerous papers and several books have been published on this topic in the last five years (e.g. [10], [7]). Most of these works, however, have concentrated on simulated systems and therefore have not had to deal with the ambiguities of the real world. We present a new connectionist learning system designed for use on a real robot with limited computational power and that has to acquire its proficiency in a minimum number of learning trials.

Learning responses has generally been classified into supervised and unsupervised learning. In supervised learning an agent or function, often called the teacher, provides the desired output response for each input vector. Systems that do not make use of a teacher, then, are known as unsupervised learning systems.

Within unsupervised learning different levels of feedback may be available. Often an evaluation of system output is immediately available. This allows learning to occur for each input vector and output response pair. We are concerned with learning in situations in which a less immediate response is available.

In this paper we examine the problem of backing a car and trailer rig to a target location by steering the front wheels of the car. No feedback is available until the task is completed. We refer to these problems as *terminal feedback problems.* Further, we are interested in problems for which the terminal feedback is simply a boolean value (a success or failure signal).

2 ROLNNET

For learning in terminal feedback problems with boolean evaluation functions, Hougen [1] proposed the Self-Organizing Neural Network with Eligibility Traces (SONNET). The use of SONNET networks on the "pole-balancing" problem in simulation [1] and for a real-world system [2] have been presented. Prominent features of SONNET systems are self-organization for input-space partitioning, the use of eligibility traces to provide temporal sensitivity, and response learning through inter-neural cooperation.

The system presented here, Rapid Output Learning Neural Network with Eligibility Traces (ROLNNET), is a simplification of the SONNET paradigm. ROLNNET uses eligibility traces and output space inter-neural cooperation, as do SONNET systems, but does not use self-organization for input-space partitioning. Instead, the input space is partitioned by problem specification evaluation, prior to the application of the learning system. This simplifies the problem and allows the network to learn more rapidly.

*This work was funded in part by the NSF under grant NSF/DUE-9351513 and by the AT&T Foundation.

2.1 Neighborhood function

The internal structure of a ROLNNET map is defined by a topological ordering of the neurons that remains unaltered as the network learns. We use a planar topology. Each neuron is uniquely numbered with a pair of integers that can be thought of as its coordinates in topology space. The existence of a network topology allows for the definition of a *distance* function for the neurons. Typically, this is defined as the Cartesian distance between coordinates in topology space.

The distance function often is used indirectly through the definition of *neighborhoods*. A neighborhood may have any width from zero (the neighborhood is restricted to the unit itself) to the maximum separation between units in topology space (the entire network is in the neighborhood) and may vary with time, typically starting large and subsequently decreasing.

More formally, if U is the set of units u in the network, d the distance function defined on topology space, and W a time dependent function specifying the width of the neighborhood, then the neighborhood N of neuron n at time t may be defined as

$$N_n(t) = \{u \in U \mid d(n,u) < W(t)\} \qquad (1)$$

All units within a neighborhood may be treated as belonging to a single class for a particular computation, and those outside as belonging to a separate class, giving a discretization which improves the computational efficiency of the method.

The concept of the neighborhood relationship is borrowed from Kohonen's Self-Organizing Topological Feature Maps [4] where the neighborhoods are used for self-organization of the maps.

2.2 Competition

Each neural element in the network is sensitive to a particular region in the input space of the problem. For the present application, the input dimensions are evenly partitioned into eight regions along each axis. The sensitivity regions of the neurons are set to be the 64 resulting divisions of the input space such that nearest neighbors (according to the neighborhood relation described above) are assigned adjacent regions of the input space.

Each time a new input vector is given to the network the neuron sensitive to the input region into which the vector falls is declared the "winner". The winning neuron gives an output response based on its response weight value (see 2.4) and has its eligibility for adaptation increased (see 2.3).

2.3 The eligibility trace

One function of biological neurons which has not been approximated in the more standard connectionist systems is what we refer to as the eligibility trace. It is known that many neurons become more amenable to change when they fire (see, e.g. [3]). This plasticity reduces with time, but provides an opportunity for learning based on feedback received by the neuron after its activity.

All neural elements in a ROLNNET system have an eligibility value associated with them. Initially, all neurons are given an eligibility value of zero. Each time a neural element fires (gives an output response), its eligibility is increased by a preset amount which is uniform for all neurons in the network. The eligibility value for each neuron decays exponentially regardless of whether or not it fires on any given time step.

2.4 Output weights

Each neural element has a single output weight which is initially given a random value. Together with the input region sensitivities described above (2.2), the weights can be understood as describing a mapping from car-trailer states to output responses.

The output weights are used to determine the system's response to an input vector. For the present application, the weight value of the winning neuron (see 2.2) is examined for its sign alone. If the weight is positive, the wheels of the car are turned to the right. Otherwise, the wheels are turned to the left.

When success or failure is signaled, the weights are updated according to the following equation:

$$w^{new} = \mathrm{sign}(w^{old})(|w^{old}| + e\, s(T)\, f) \qquad (2)$$

where w is the weight, e is the eligibility for adaptation, s is a scaling function that changes with the trial number T, and f is the feedback signal (+1 for success, -1 for failure). The scaling function s is used to allow for large changes to the weights in early training trials and smaller changes in subsequent trials. In this application, s is defined to be

$$s(T) = \frac{1}{1 + (T-1) \bmod 10} \qquad (3)$$

2.5 Inter-neural cooperation

After the completion of a trial and the subsequent updating of each neuron's weight according to its eligibility, inter-neural cooperation takes place. This consists of neurons updating their weights a second

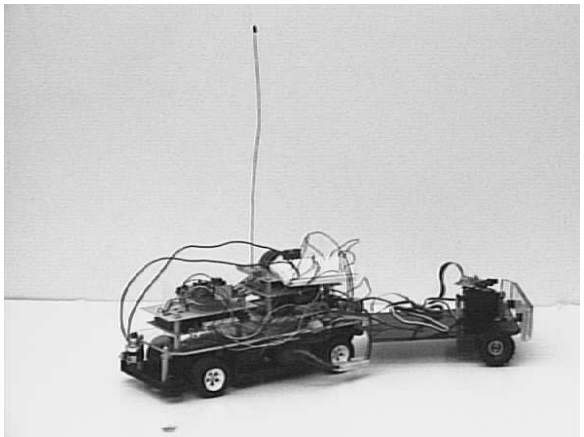


Figure 1: TBMin, the trailer backing mini-robot

time, this time based on the weight values of the other neurons in their neighborhood. For the present application, the neighborhood has a constant size of 1.

Each individual neuron i is influenced by its neighbors according to the following equation:

$$w_i = (1 - \alpha(T))w_i + \alpha(T) \sum_{n \in N} \frac{w_n}{m} \qquad (4)$$

where each w is a weight, N is the neighborhood of neuron i, m is the number of neurons in that neighborhood, and α determines the degree to which a neuron's value is "smoothed" with that of its neighbors. In general, α decreases with time. This means that each neuron's value becomes more independent from those of its neighbors as time passes. In this particular application, α is defined to be

$$\alpha(T) = \frac{1}{2 + (T-1)\, mod\, 10} \qquad (5)$$

where T is the trial number. (I.e. For the first ten trials, α is 1/2, for the second 10 it is 1/3, and so on.)

3 Autonomous mini-robots

Figure 1 shows the autonomous mini-robot *TBMin* that we have designed and built for this problem. The body of TBMin is made up of a small radio controlled car and a trailer. The original electronics have been replaced by specially designed boards. We use (1) a micro-computer board built around a 68hc11 microcontroller, with 16k of ROM and 32k of RAM. (2) a motor board also built around a 68hc11 microcontroller, with 16k of EPROM and a L293E dual-motor-driver chip. This board, that operates as a slave, keeps the car at a constant velocity, controls the steering, and controls a light-tracking head; (3) an interface board that synchronizes the communication between the micro-computer and the motor board. A 7.2 volt rechargeable battery is used to drive the car and the boards. The battery lasts approximately 15 minutes.

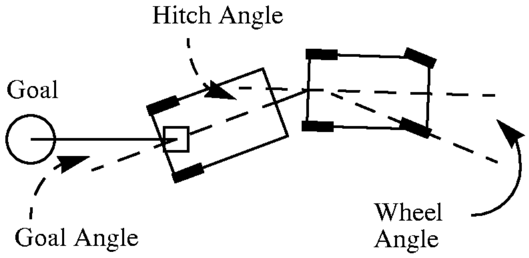


Figure 2: The trailer-backing problem

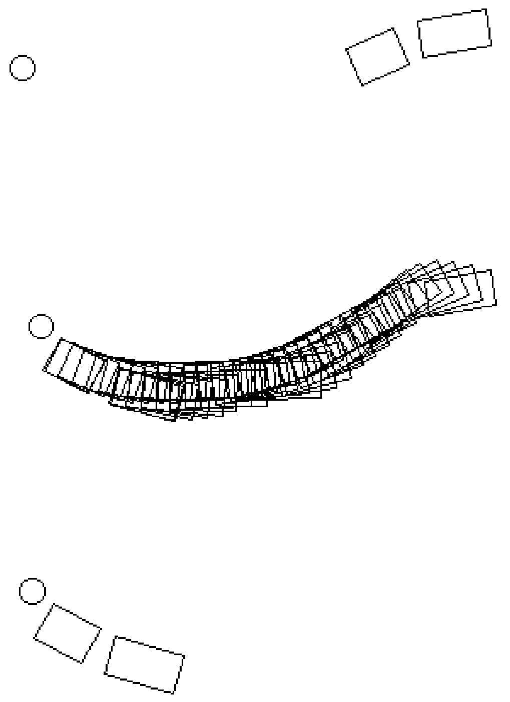
TBMin uses two inputs: (1) the angle from the spine of the trailer to the goal and (2) the angle of the hitch, as shown in Figure 2. The angle to the goal is computed using a light-tracking head (mounted on a servo) that tracks a 100W light bulb. The angle of the hitch is sensed using a variable resistor. If the rear of the trailer reaches the goal, success is signaled. If the angle to the target exceeds 90° or the angle of the hitch exceeds 45°, failure is signaled. The system is clocked to operate in discrete time units. No other sensory data or feedback is available.

Other researchers have defined the problem differently. Whereas most trailer-backing systems include such variables as the x and y position of the rear of the trailer in Cartesian space, we do not. This is primarily because our system is designed to operate in the real world where it is difficult for the robot to acquire its own x and y coordinates.

4 Experimental results

The system described in this paper must meet many varied and sometimes conflicting challenges. Some of these are inherent in the task to be learned, some are imposed by the use of real robotic systems, some come from our use of small yet autonomous mini-robots, and some we have imposed on ourselves to study learning in completely unsupervised domains. The results in this section show that the system is able to function in the demanding environment in which we have placed it and produce good results as quickly as we need them.

The system was tested in simulation and on TBMin. For both the simulation and the real robot, the learning system was initialized to having random values for output weights. At the start of each trial, the eligibility for adaptation for each neuron was set to zero. On each time step the learning system was



(a)

(b)

(c)

Figure 3: Example simulation trial. (a) start, (b) progress (shown every tenth time step), (c) end.

given the current values of the trailer and hitch angles and, if applicable, a failure or success signal. The network response was thresholded and values less than zero were used as *hard left* control signals, while values equal to or greater than zero are used as *hard rights* by TBMin. In this way, the system was given *bang-bang* control. The output values would not have to be thresholded and the system could learn smoother control, if more sophisticated hardware were utilized.

4.1 TBMin results

For testing the real robot, the car-trailer rig was positioned to match as closely as possible the initial positions selected as in case 1 (see 4.2). The system was trained in a series of runs of 50 trials each. (The maximum period for which a single battery can be expected to last is approximately 100 trials. To insure that battery decline does not cause failure of the onboard computer, runs were limited to half this extent.) Five runs of the learning system were made on the actual robot, each starting with the output weights initialized to random values. These runs are plotted below for comparison with the runs done in simulation. A comparison of the simulation plots with the runs using TBMin shows that the learning system is not significantly affected by the inaccuracy inherent in the real-world system.

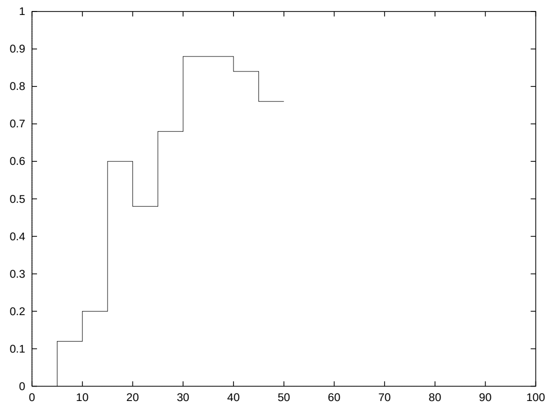


Figure 4: Graph of runs on TBMin. Average performance for 5 runs of 50 trials each. Trials graphed together in groups of five. Initial position: Rear of trailer 4 to 7 feet from target, angle to target $+/-60°$, angle of hitch $+/-15°$. New random position for each trial.

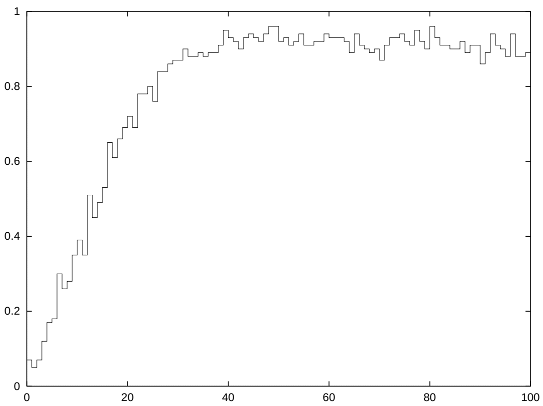
4.2 Simulation results

For the simulation testing, we studied four cases that varied in the range of possible initial car-trailer positions and in the presentation of new positions to the learning system. For each case the system was trained in a series of runs of 100 trials each.

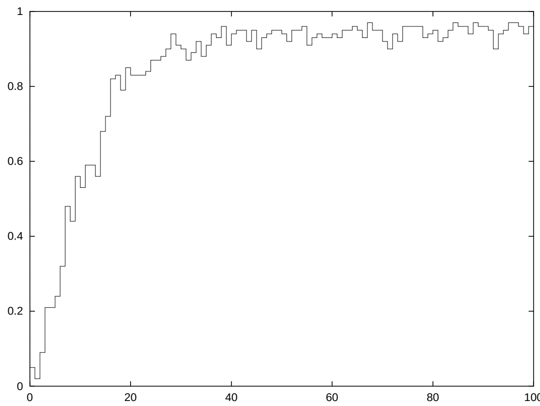
Case 1: the car-trailer rig was started with the back of the trailer from four to seven feet from the target, with an angle to the target between $-60°$ and $+60°$, and with a hitch angle between $-15°$ and $+15°$. New initial conditions were chosen randomly at the start of each trial (see below) with a uniform distribution over the entire range.

Case 2: this is identical to case 1, except that the rig was started with the back of the trailer from three to six feet from the target and with an angle to the target between $-45°$ and $+45°$.

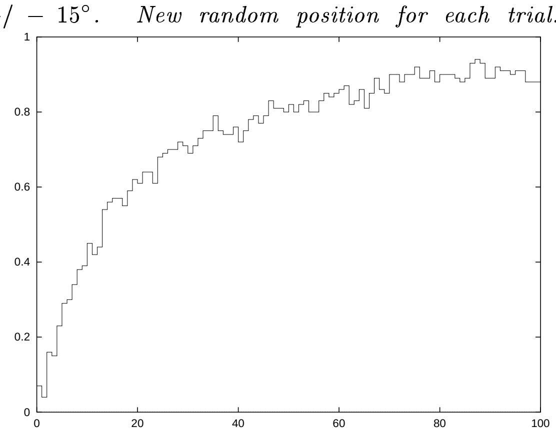
Case 3: resembled case 1, differing only in when new initial positions were presented. In case 3, new initial positions were given only when success had been achieved with the previous position; when the system failed to reach the target, the same initial position would be repeated.



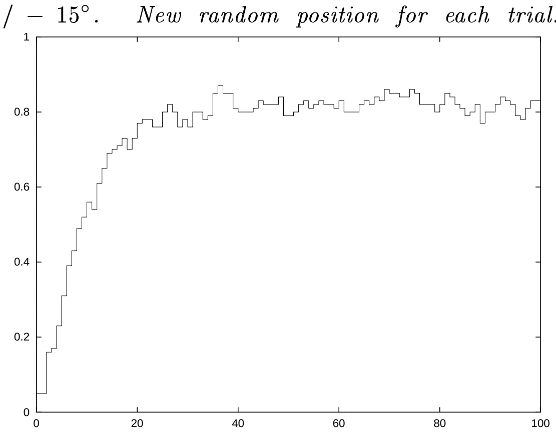
Graph of simulation case 1. Average performance for 100 runs of 100 trials each. Initial position: Rear of trailer 4 to 7 feet from target, angle to target $+/-60°$, angle of hitch $+/-15°$. New random position for each trial.



Graph of simulation case 2. Average performance for 100 runs of 100 trials each. Initial position: Rear of trailer 3 to 6 feet from target, angle to target $+/-45°$, angle of hitch $+/-15°$. New random position for each trial.



Graph of simulation case 3. Average performance for 100 runs of 100 trials each. Initial position: Rear of trailer 4 to 7 feet from target, angle to target $+/-60°$, angle of hitch $+/-15°$. New random position only following success; previous position repeated after failure.



Graph of simulation case 4. Average performance for 100 runs of 100 trials each. Initial position: Rear of trailer 3 to 6 feet from target, angle to target $+/-45°$, angle of hitch $+/-15°$. New random position only following success; previous position repeated after failure.

Figure 5: Simulation data

Case 4: resembled case 2, differing from it in the same way case 3 differed from case 1.

A total of 100 runs were made in simulation for each of four cases, using different random seeds for both the initial starting positions for each trial and for the random initial values of the output weights. The results are plotted in the graphs in Figure 5.

As can be seen from these graphs, the system quickly improved its performance, reaching its maximum performance in three of the four cases within 40 trials. In the fourth case, the system reached its maximum performance in well under 100 trials. This is amazingly fast learning. (While a realistic comparison of systems cannot be made it might be of interest to know that other authors have used as many as 20,000 trials to train their systems to control trailer-backing). In fact, the success rate of completely random control for the given initial conditions was determined experimentally to be zero, and the success rate using control by a random network without learning was determined to be approximately 6%, so it is obvious that significant learning has taken place within the first ten trials.

The results for Cases 1 and 2 were very similar with performance increasing only slightly faster and reaching a slightly higher level in Case 2 than in Case 1. While the greater possible starting angles in Case 1 made the problem more difficult, this was apparently mostly offset by the greater distance from the target, which provided more room for the control system to bring the trailer around. The performance in Cases 3 and 4 respectively increased more slowly and reached a lower maximum level than either Case 1 or 2. This might be explained by the bias towards presentation of hard initial positions in Cases 3 and 4. The steep initial slope of all the graphs, including that for the runs using TBMin, indicates that significant learning occurred on failure as well as success.

5 Related work

The trailer-backing problem is gaining attention as a simple to understand yet difficult to solve learning-control problem. Approaches such as the Cerebellar Model Articulated Controller (CMAC) [8], adaptive fuzzy systems [5], backpropagation through time [6], and "fuzzy BOXES" [11] have all been applied to versions of this problem. It is difficult to directly compare results across many of these systems and with our results.

The most obvious difference between this study and those of most other authors writing on this problem, is that our system was implemented on a real robot, whereas theirs were restricted to simulated systems. Restricting their attention to simulated systems has freed many researchers from having to deal with unpleasantries such as noisy input data and imprecision in the execution of tasks, although some have tried to model these types of effects in simulation (e.g. [9]). The use of simulation has also allowed other researchers to include many more learning trials in their training runs and to use learning systems with much greater memory and computational demands. For these reasons, the results of these other learning systems cannot be directly compared with ours. Unfortunately for the sake of honest comparison, we are aware of no other research involving trailer-backing using real robots. All of the differences taken in combination make it clear that simple comparison of success rates between ROLNNET and other learning systems for trailer-backing are of no value.

6 Conclusions

We have described a paradigm for learning simple tasks for real robots and we have presented experimental evidence to support our proposed approach. We have shown that our system is able to learn extremely quickly, despite noise, and with limited computing power and feedback. The paradigm is rich with possibilities for further study, including novel network architectures and hybridization with other systems (such as self-learning critics).

References

[1] Dean F. Hougen. Use of an eligibility trace to self-organize output. In *Science of Artificial Neural Networks II, Proceedings SPIE*, volume 1966, pages 436–447, 1993.

[2] Dean F. Hougen, John Fischer, and Deva Johnam. A neural network pole balancer that learns and operates on a real robot in real time. In *Proceedings of the MLC-COLT Workshop on Robot Learning*, pages 73–80, 1994.

[3] A. Klopf. Brain function and adaptive systems – a heterostatic theory. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, 1974.

[4] T. K. Kohonen. *Self-organizing and associative memory.* Springer-Verlag, Berlin, 3rd edition, 1989.

[5] Seong-Gon Kong and Bart Kosko. Adaptive fuzzy systems for backing up a truck-and-trailer. *IEEE Trans. on Neural Networks*, 3(2):211–223, 1992.

[6] D. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, 10(3):18–23, 1990.

[7] H. Ritter, T. Martinetz, and K. Schulten. *Neural computation and self-organizing maps: an introduction.* Addison-Wesley, Reading, MA, 1992.

[8] Robert O. Shelton and James K. Peterson. Controlling a truck with an adaptive critic CMAC design. *Simulation*, 58(5):319–326, 1992.

[9] T. Troudet and W. Merrill. Neuromorphic learning of continuous-valued mappings from noise-corrupted data. *IEEE Trans. on Neural Networks*, 2(2):294–301, 1991.

[10] III W. Thomas Miller, Richard S. Sutton, and Paul J. Werbos. *Neural Networks for Control.* MIT Press, Cambridge, MA, 1990.

[11] N. Woodcock, N. J. Hallam, and P. D.Picton. Fuzzy BOXES as an alternative to neural networks for difficult control problems. *Artificial Inteligence in Engineering*, pages 903–919, 1991.