Chapter 6

CONCLUSIONS

In this dissertation we addressed an extremely general robot scenario. A robot is placed in an unfamiliar environment. It explores the environment by performing actions and observing their perceptual consequences. The robot's task is to construct an internal model of the environment that will allow it to predict the outcomes of its actions and to determine what sequences of actions it should take to reach particular goal states.

We presented a technique, called SLUG, for building internal models of finite state environments. SLUG combines techniques from computational learning theory and connectionism. Several benefits of SLUG have been demonstrated. It improves on the Rivest and Schapire's learning alogrithm (hereafter, *the RS algorithm*) in the following ways:

- Connectionist learning algorithms allow for a great deal of parallelism. The network can update nearly all its weights after each action. In contrast, the RS algorithm gains information about at most a few update graph arcs at a time after each action. Furthermore, SLUG performs parameter estimation and state estimation simultaneously, whereas the RS algorithm performs parameter estimation and state estimation as two separate and distinct tasks.

- In contrast to the RS algorithm, SLUG's performance degrades gracefully as the number of units decreases below the required number of units, and its performance does not appear to degrade, and often improves, as the number of units increases above the required number of units [58].

- SLUG permits the continous availability of predictions, which gradually improve with experience. Furthermore, before learning is complete, the predictions can be substantially correct. In contrast, the RS algorithm can only make predictions after the update graph is complete.

- SLUG can be generalized to accomodate stochastic environments, including Markov environments (see Durbin and Bachrach [18]) and environments in which the sensations are somewhat unreliable. The RS algorithm is designed specifically for deterministic environments.

- As pointed out in Mozer and Bachrach [57], treating the update graph as a connectionist network suggests generalizations of the update graph formalism that don't arise from a more traditional analysis. First, there is the direct extension of allowing complementation links. Second, because SLUG can be considered a time-varying linear system where the weight matrix is chosen according to the current action, any rank-preserving linear transform of the weight matrices will produce an equivalent system, but one that does not have the local connectivity of the update graph. The linearity of the network also permits the use of linear algebra tools for analysis of the resulting connectivity matrices. Finally, the update representation can be even further extended (e.g., by adding hidden units) without changing the basic learning algorithm.

Rivest and Schapire's approach offers a few advantages over SLUG. First of all, their learning algorithm is provably polynomial for a restricted class of environments. There are no analogous results for SLUG. Secondly, their algorithm scales well for complex environments—ones in which the number of nodes in the update graph is quite large and the number of distinguishing environmental sensations is relatively small. One example environment is the n-bit shift register. This is probably the greatest weakness of SLUG. In these environments, an intelligent exploration strategy is crucial.

SLUG has a number of beneficial properties that differentiate it from typical connectionist learning methods:

- After training, because the activities and weights are binary, the update graph can perfectly predict the sensations infinitely far into the future; the state units are readily interpretable within the update graph formalism; many formal and empirical results exist that lend intuition and rigor.

- SLUG outperforms other "conventional" recurrent networks. These recurrent networks—Elman network [22] and Jordan network [34]—are spectacularly unsuccessful at learning to model FSA's (see Section 3.6).

- Unlike other networks, SLUG greatly facilitates the coding of a priori knowledge about the environment. Knowledge of relations between actions and the effect of particular actions can be hardwired into the weights of the network. The explicit representation of the effects of actions on the internal state as separate weight matrices eases this process.

- Unlike other connectionist architectures (including Giles et al's second-order network [26]), SLUG can be considered a time-varying linear system. A large number of well-studied techniques from optimal estimation apply to linear time-varying systems including the Kalman filter [25].

We also presented a control acquisition architecture that can use SLUG to determine a minimal sequence of actions to follow to reach a particular goal state. This architecture borrows heavily from other control acquisition techniques. In particular, our control acquisition architecture can be considered a 3-net architecture without the controller. This simplifies the relationship between the critic and the chosen actions—the critic directly defines the actions. Empirically, this was found to improve the learning performance.

We also considered the complexities of more realistic environments, characterized by high-dimensional continuous state-spaces with real-valued actions and sensations. We presented a connectionist control acquisition architecture tailored for simulated short-range homing in the presence of obstacles. Our architecture overcomes previous limitations of the artificial potential function approach (see also Koditschek [43], LaTombe [47], and Connolly, Burns, and Weiss [15] for techniques that also overcome these limitations). In particular, our approach can avoid local minima and produce efficient trajectories.

In solving this control task, we demonstrated several techniques. First, we demonstrated the use of Gaussian units whose initial weights are determined using a soft form of competitive learning, called expectation maximization, that, in our case, create spatially tuned units. Second, we demonstrated the use of a hard-wired range-flow model whose performance is good in a wide range of environments. Finally, we demonstrated the use of a nominal initial controller.

Much work is needed to scale the navigation architecture to real-world navigation tasks involving real robots and large-scale environments. Lessons learned from the simulated small-scale environments will be very beneficial.