Learning Continuous Perception-Action Models Through Experience

Ashwin Ram and Juan Carlos Santamaría
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{ashwin,carlos}@cc.gatech.edu

Autonomous robotic navigation is defined as the task of finding a path along which a robot can move safely from a source point to a destination point in a obstacle-ridden terrain and, often, executing the actions to carry out the movement in a real or simulated world. Several methods have been proposed for this task, ranging from high-level planning methods to reactive methods.

High-level planning methods use extensive world knowledge and inferences about the environment they interact with (see [Fikes *et al.*, 1972; Georgeff, 1987; Maes, 1990; Sacerdoti, 1975]). Knowledge about available actions and their consequences is used to formulate a detailed plan before the actions are actually executed in the world. Such systems can successfully perform the path-finding required by the navigation task, but only if an accurate and complete representation of the world is available to the system. Considerable high-level knowledge is also needed to learn from planning experiences (see, for example, [Hammond, 1989; Minton, 1988; Mostow and Bhatnagar, 1987; Segre, 1988]). Such a representation is usually not available in real-world environments, which are complex and dynamic in nature. To build the necessary representations, a fast and accurate perception process is required to reliably map sensory inputs to high-level representations of the world. A second problem with high-level planning is the large amount of processing time required, resulting in significant slowdown and the inability to respond immediately to unexpected situations.

Situated or reactive control methods have been proposed as an alternative to high-level planning methods (see [Arkin, 1989; Brooks, 1986; Kaelbling, 1986; Payton, 1986]). In these methods, no planning is performed; instead, a simple sensory representation of the environment is used to select the next action that should be performed. Actions are represented as simple behaviors, which can be selected and executed rapidly, often in real-time. These methods can cope with unknown and dynamic environmental configurations, but only those that lie within the scope of predetermined behaviors. It requires a great deal of careful design and tuning on the part of the human designer to develop the control systems that drive such robots, and even then these systems run into serious difficulties when faced with environments which are different from those that the designer anticipated. Furthermore, even if the designer could anticipate and model all the relevant aspects of the operating environment of the robot, the dynamic nature of the real world would render parts of this model obsolete over time.

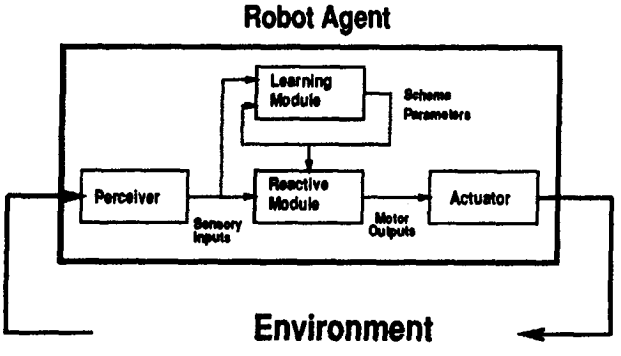

Figure 1: System architecture

The ability to adapt to changes in the environment, and learn from experiences, is crucial to adequate performance and survivability in the real world. We have developed a self-improving navigation system that uses reactive control for fast performance, augmented with multistrategy learning methods that allow the system to adapt to novel environments and to learn from its experiences (see figure 1). The system autonomously and progressively constructs representational structures that encapsulate its experiences into "cases" that are then used to aid the navigation task in two ways: they allow the system to dynamically select the appropriate robotic control behaviors in different situations, and they also allow the system to adapt selected behaviors to the immediate demands of the environment (see [Moorman and Ram, 1992; Ram *et al.*, 1992; Ram and Santamaria, 1993a; Ram and Santamaria, 1993b] for further details).

The system's cases are automatically constructed using a hybrid case-based and reinforcement learning method without extensive high-level reasoning. The learning and reactive modules function in an integrated manner. The learning module is always trying to find a better model of the interaction of the system with its environment so that it can tune the reactive module to perform its function better. The reactive module provides feedback to the learning module so it can build a better model of this interaction. The behavior of the system is the result of an equilibrium point established by the learning module, which is trying to refine the model, and the environment, which is complex and dynamic in nature. This

equilibrium may shift and need to be re-established if the environment changes drastically; however, the model is generic enough at any point to be able to deal with a very wide range of environments.

The learning method is based on a combination of ideas from case-based reasoning and learning, which deals with the issue of using past experiences to deal with and learn from novel situations, and from reinforcement learning, which deals with the issue of updating the content of system's knowledge based on feedback from the environment (e.g., see [Sutton, 1992]). However, in traditional case-based planning systems (e.g., [Hammond, 1989]) learning and adaptation requires a detailed model of the domain. This is exactly what reactive planning systems are trying to avoid. Earlier attempts to combine reactive control with classical planning systems (e.g., [Chien *et al.*, 1991]) or explanation-based learning systems (e.g., [Mitchell, 1990]) also relied on deep reasoning and were typically too slow for the fast, reflexive behavior required in reactive control systems. Unlike these approaches, our method does not fall back on slow non-reactive techniques for improving reactive control.

Each case represents an observed regularity between a particular environmental configuration and the effects of different actions, and prescribes the values of the schema parameters that are most appropriate (as far as the system knows based on its previous experience) for that environment. The learning module performs the following tasks in a cyclic manner: (1) **perceive** and represent the current environment; (2) **retrieve** a case whose input vector represents an environment most similar to the current environment; (3) **adapt** the schema parameter values in use by the reactive control module by installing the values recommended by the output vectors of the case; and (4) **learn** new associations and/or adapt existing associations represented in the case to reflect any new information gained through the use of the case in the new situation to enhance the reliability of their predictions.

The **perceive** step builds a set of four input vectors, one for each sensory input described earlier, which are matched against the corresponding input vectors of the cases in the system's memory in the **retrieve** step. The case similarity metric is based on the mean squared difference between each of the vector values of the case over a trending window, and the vector values of the environment. The best match window is calculated using a reverse sweep over the time axis similar to a convolution process to find the relative position that matches best. The best matching case is handed to the **adapt** step, which selects the schema parameter values from the output vectors of the case and modifies the corresponding values of the reactive behaviors currently in use using a reinforcement formula which uses the case similarity metric as a scalar reward. Thus the actual adaptations performed depend on the goodness of match between the case and the environment.

Finally, the **learn** step uses statistical information about prior applications of the case to determine whether information from the current application of the case should be used to modify this case, or whether a new case should be created. The vectors encoded in the cases are adapted using a reinforcement formula in which a *relative similarity measure* is used as a scalar reward or reinforcement signal. The

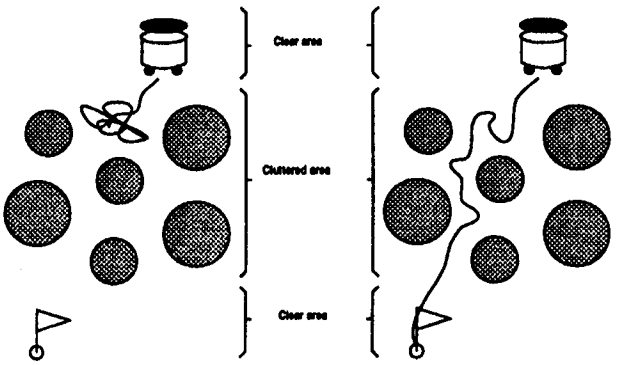

Figure 2: Typical navigational behaviors of different tunings of the reactive control module. The figure on the left shows the non-learning system with high obstacle avoidance and low goal attraction. On the right, the learning system has lowered obstacle avoidance and increased goal attraction, allowing it to "squeeze" through the obstacles and then take a relatively direct path to the goal.

relative similarity measure quantifies how similar the current environment configuration is to the environment configuration encoded by the case relative to how similar the environment has been in previous utilizations of the case. Intuitively, if case matches the current situation better than previous situations it was used in, it is likely that the situation involves the very regularities that the case is beginning to capture; thus, it is worthwhile modifying the case in the direction of the current situation. Alternatively, if the match is not quite as good, the case should not be modified because that will take it away from the regularity it was converging towards. Finally, if the current situation is a very bad fit to the case, it makes more sense to create a new case to represent what is probably a new class of situations.

A detailed description of each step would require more space than is available in this paper (see [Ram and Santamaria, 1993a; Ram and Santamaria, 1993b] for details). Here, we note that since the reinforcement formula is based on a relative similarity measure, the overall effect of the learning process is to cause the cases to converge on stable associations between environment configurations and schema parameters. Stable associations represent regularities in the world that have been identified by the system through its experience, and provide the predictive power necessary to navigate in future situations. The assumption behind this method is that the interaction between the system and the environment can be characterized by a finite set of causal patterns or associations between the sensory inputs and the actions performed by the system. The method allows the system to learn these causal patterns and to use them to modify its actions by updating its schema parameters as appropriate.

We have developed a three-dimensional interactive visualization of a robot navigating through a simulated obstacle-ridden world that allows a user to configure and test the navigational performance of the robot (see figure 2). The user can also test two robots simultaneously, and compare the performance of a robot using traditional, non-learning reactive control with another that uses our method to adapt itself to the

environment to enhance the performance of the navigation.

The system has been tested through extensive empirical simulations on a wide variety of environments using several different performance metrics (see [Moorman and Ram, 1992; Ram *et al.*, 1992; Ram and Santamaria, 1993b] for further details). The system is very robust and can perform successfully in (and learn from) novel environments, yet it compares favorably with traditional reactive methods in terms of speed and performance. A further advantage of the method is that the system designers do not need to foresee and represent all the possibilities that might occur since the system develops its own "understanding" of the world and its actions. Through experience, the system is able to adapt to, and perform well in, a wide range of environments without any user intervention or supervisory input. This is a primary characteristic that autonomous agents must have to interact with real-world environments.

References

[Arkin, 1989] Ronald C. Arkin. Motor schema-based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112, August 1989.

[Brooks, 1986] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, August 1986.

[Chien *et al.*, 1991] S. A. Chien, M. T. Gervasio, and G. F. DeJong. On becoming decreasingly reactive: Learning to deliberate minimally. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 288–292, Chicago, IL, June 1991.

[Fikes *et al.*, 1972] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Georgeff, 1987] M. Georgeff. Planning. *Annual Review of Computer Science*, 2:359–400, 1987.

[Hammond, 1989] Kristian J. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task.* Perspectives in Artificial Intelligence. Academic Press, Boston, MA, 1989.

[Kaelbling, 1986] L. Kaelbling. An architecture for intelligent reactive systems. Technical Note 400, SRI International, October 1986.

[Maes, 1990] Pattie Maes. Situated agents can have goals. *Robotics and Autonomous Systems*, 6:49–70, 1990.

[Minton, 1988] Steven Minton. *Learning effective search control knowledge: An explanation-based approach.* PhD thesis, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA, 1988. Technical Report CMU-CS-88-133.

[Mitchell, 1990] T. M. Mitchell. Becoming increasingly reactive. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1051–1058, Boston, MA, August 1990.

[Moorman and Ram, 1992] Kenneth Moorman and Ashwin Ram. A case-based approach to reactive control for autonomous robots. In *Proceedings of the AAAI Fall Symposium on AI for Real-World Autonomous Mobile Robots*, Cambridge, MA, October 1992.

[Mostow and Bhatnagar, 1987] J. Mostow and N. Bhatnagar. FAILSAFE – A floor planner that uses EBG to learn from its failures. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 249–255, Milan, Italy, August 1987.

[Payton, 1986] D. Payton. An architecture for reflexive autonomous vehicle control. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1838–1845, 1986.

[Ram and Santamaria, 1993a] Ashwin Ram and Juan Carlos Santamaria. Continuous case-based reasoning. In D. B. Leake, editor, *Proceedings of the AAAI Workshop on Case-Based Reasoning*, Washington, DC, July 1993.

[Ram and Santamaria, 1993b] Ashwin Ram and Juan Carlos Santamaria. A multistrategy case-based and reinforcement learning approach to self-improving reactive control systems for autonomous robotic navigation. In R. S. Michalski and G. Tecuci, editors, *Proceedings of the Second International Workshop on Multistrategy Learning*, Harpers Ferry, WV, May 1993. Center for Artificial Intelligence, George Mason University, Fairfax, VA.

[Ram *et al.*, 1992] Ashwin Ram, Ronald C. Arkin, Kenneth Moorman, and Russell J. Clark. Case-based reactive navigation: A case-based method for on-line selection and adaptation of reactive control parameters in autonomous robotic systems. Technical Report GIT-CC-92/57, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1992.

[Sacerdoti, 1975] E. D. Sacerdoti. A structure for plans and behavior. Technical Note 109, Stanford Research Institute, Artificial Intelligence Center, 1975. Summarized in P.R. Cohen and E.A. Feigenbaum's Handbook of AI, Vol. III, pp. 541-550.

[Segre, 1988] Alberto M. Segre. *Machine Learning of Robot Assembly Plans.* Kluwer Academic Publishers, Norwell, MA, 1988.

[Sutton, 1992] R. S. Sutton, editor. *Machine Learning, Special issue on reinforcement learning*, volume 8(3/4). Kluwer Academic, Hingham, MA, 1992.