

# Hierarchical Learning of Reactive Behaviors in an Autonomous Mobile Robot

Olac Fuentes, Rajesh P. N. Rao and Michael Van Wie  
Computer Science Department  
University of Rochester  
Rochester, New York 14627, USA  
e-mail: {fuentes,rao,vanwie}@cs.rochester.edu

## Abstract

We describe an autonomous mobile robot that employs a simple sensorimotor learning algorithm at three different behavioral levels to achieve coherent goal-directed behavior. The robot autonomously navigates to a goal destination within an obstacle-ridden environment by using the learned behaviors of obstacle-detection, obstacle-avoidance, and beacon-following. These reactive behaviors are learned in a hierarchical manner by using a hillclimbing routine that attempts to find the optimal transfer function from perceptions to actions for each behavior. We present experimental results that show that each behavior was successfully learned by the robot within a reasonably short period of time. We conclude by discussing salient features of our approach and possible directions for future research.

## 1. Introduction

Traditionally, the task of developing a sensorimotor control architecture for a situated autonomous robot has been left to the human programmer of the robot. Prewiring robot behaviors by hand, however, becomes increasingly complex for robots with a large number of sensors and effectors, especially when they are performing sophisticated tasks that involve continuous interaction with the encompassing environment. In such cases, it is our belief that *hierarchical behavior-based decomposition* of the control architecture, as originally suggested by Brooks [1], can simplify the task of programming complex systems. In addition, it is desirable in many cases to endow the robot with the ability to adapt its constituent behaviors on-line in response to environmental stimuli by allowing it *to learn autonomously* the transfer function mapping sensory input into motor commands.

This work is supported by NSF research grants CDA-8822724, CDA-9401102, and CDA-9306454; ARPA grant MDA972-92-J-1012; and ONR grant N00014-93-1-0221.

For even moderately complex tasks and/or robots, the high dimensionality of the sensorimotor space makes learning difficult. One commonly used approach to make robot learning feasible despite the high dimensionality of the sensory space is to run the learning algorithm on a simulated environment (for example, [7]). However, in many situations, it is difficult or impossible to gather enough knowledge about the robot and its environment to build an accurate simulation. Moreover, some physical events, such as collisions, are difficult to simulate even when we have accurate knowledge of our environment. For these reasons, we believe that in order for the learned skills to be useful to the physical robot in its environment, all the learning and experimentation has to be carried out by the embodied physical robot itself. However, using a real robot has some drawbacks: given the slowness of real world experimentation and the limited computing power typically available to autonomous mobile robots, for the learning algorithms to be successfully applied, it is crucial that they converge within a reasonable number of trials and that they don't require large amounts of memory. The technique we present in this paper satisfies both requirements.

This paper describes an autonomous mobile robot that employs a simple sensorimotor learning algorithm at three different behavioral levels to achieve coherent goal-directed behavior. In particular, the robot solves the task of navigating to a goal destination (indicated by an infrared beacon) within an obstacle-ridden environment by using a set of learned behaviors for obstacle-detection, obstacle-avoidance, and beacon-following. The behaviors themselves are learned individually through simple heuristic hill-climbing. Instead of building a model of its environment, our algorithm uses a set of perceptual goals to guide its search.

## 2. Task Description

The task to be learned by the robot (figure 1) is one of navigation and obstacle-avoidance. Specifically, we expect the robot to learn appropriate sensorimotor strategies for navigating between two points in an obstacle-ridden environment.

Three classes of sensory input are available to the robot:

- **Bump Sensors:** Realized using digital microswitches, these sensors indicate whether the robot is physically touching an obstacle. Five of these sensors, placed at different locations around the robot, are used for learning the *obstacle-detection* behavior. In particular, the robot is expected to learn to back up when its front bump sensors are active, to turn left when the right bump sensor is active, and so on.
- **Photosensors:** Three shielded photoresistors placed in a tripod configuration are used to give advance warning of an approaching obstacle, taking advantage of the fact that the obstacles have a darker color than the floor. The inputs from these sensors are used for learning the *obstacle-avoidance* behavior; they are expected to allow the robot to steer clear of obstacles detected in its path.
- **Infrared detectors:** These sensors, when used in conjunction with infrared detection software, indicate the strength of the modulated infrared light in a small spread along their lines of sight. Four of these sensors are used to learn the high-level behavior of navigating toward the goal position, which is a source of infrared transmission.

The above sensory repertoire is supplemented by two effectors consisting of a drive motor attached to the robot's back axle and a servo motor at the front that is used for steering.

The environment is as shown in figure 2, with a scattering of obstacles in an eight-foot square arena and infrared beacons at the near and far corners.

## 3. Behavior-Based Task Decomposition

Since the robot is equipped with twelve sensors and two effectors, the learning task consists of finding a mapping from the 12-dimensional sensory space to the 2-dimensional motor space that optimizes the robot's performance of the task. The number of different perceptions the robot may encounter grows exponentially with the number of sensors it possesses, thereby making the task of hardwiring behaviors cumbersome and error-prone.

One way of circumventing this "curse of dimension-

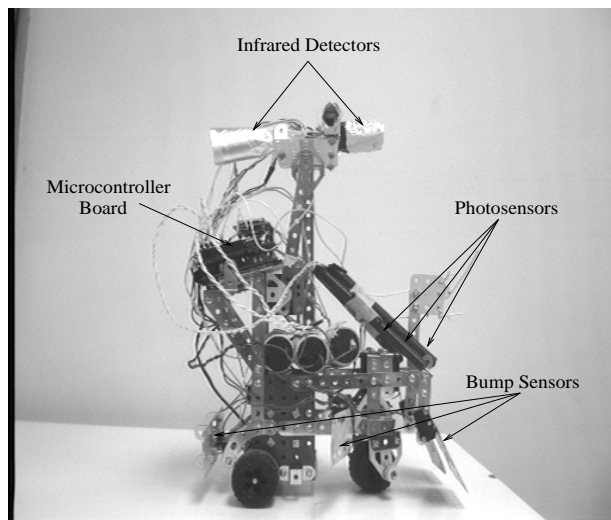


Figure 1: The Robot Used for the Experiments.

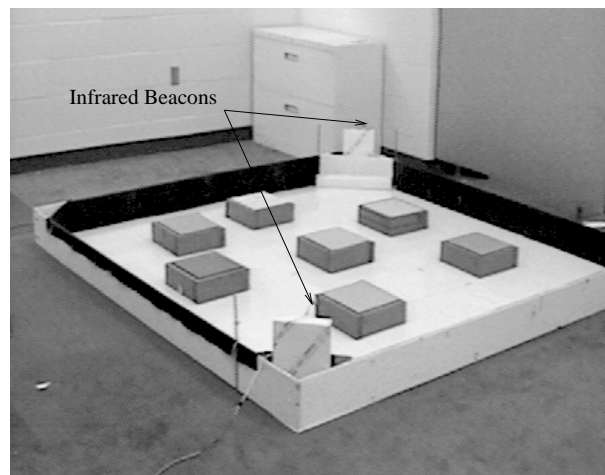


Figure 2: The Robot Arena

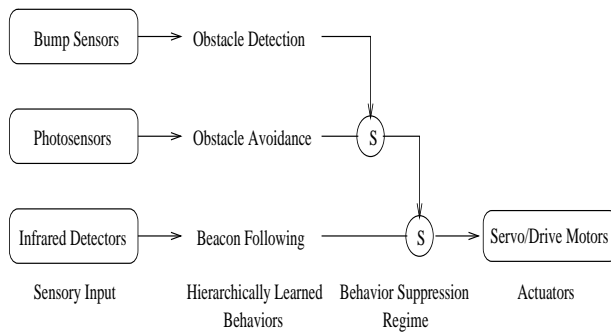


Figure 3: Block Diagram of the Robot Control Architecture

ality” is to divide the task into hierarchical layers of control. Each layer is responsible for one aspect of the goal behavior: at the first level is the obstacle-detection behavior, which controls the robot when it collides with an obstacle; at the second is obstacle-avoidance, which uses the light-sensors to steer away from obstacles before colliding with them; and at the third is beacon-following, which uses infrared detectors to direct the robot toward the current goal beacon. This approach is reminiscent of Brooks’s subsumption architecture [1]. Figure 3 illustrates our robot’s three-level hierarchical architecture.

Hierarchical partitioning of the sensory space allows the robot to learn the sensorimotor mapping that corresponds to each layer independent of the other layers. This greatly reduces the search space and allows for an implementation where all the learning can be done by physically experimenting with the world, instead of by relying on simulation. By not relying on simulation, we avoid the risk of learning a policy that works well in simulation but fails to transfer to the real world.

#### 4. Learning Reactive Behaviors Using Perceptual Goals

To learn each constituent behavior, we use a relatively simple hillclimbing technique. Since we only keep in memory a policy that encodes a series of statements of the form *perception*  $\rightarrow$  *action*, the method can be implemented using little memory. This is in contrast to some machine learning techniques recently applied to mobile robotics (for example, genetic programming [5], genetic algorithms [2], reinforcement learning [3], and neural networks [4]) that usually require the storage of considerable amounts of information.

Let  $A$  be the set of discrete actions that the robot can perform, and  $P$  be the set of relevant perceptions that the robot can obtain from its sensors. A policy is a mapping  $m : P \rightarrow A$  that defines the action  $m(p) \in A$  to be taken when confronted with the sensory stimulus  $p \in P$ .

Our algorithm uses a set of perceptual goals to guide its search. To every perception  $p$  we assign a numeric value  $v_p$  measuring the desirability or “goodness” of the situations where that perception normally occurs. For example, a perceptual input where one or more bump sensors is active will have a low  $v$ , because it occurs in the undesirable situation when the robot crashes into an obstacle, while a perceptual input where no bump sensors are active will have a high  $v$ , because it indicates the robot is clear.

The task of the learning mechanism is to learn a policy  $m$  that will take the robot from “bad” to “good” perceptions and maintain it in good percep-

tions when they are found. We achieve this by computing a heuristic metric  $h(p)$  that measures how often, on average, the action taken in situation  $p$  has resulted in perceptions that are more desirable than  $p$ . For every *perception-action* pair in the current policy, we keep a running average of heuristic values  $h$  and replace those entries in the policy that are judged to be inadequate (*i.e.* for which the average falls below a pre-specified threshold). To facilitate the maintenance of our running average, we keep a count  $n(p)$  of the number of times each perception  $p$  occurs, and a sum of the heuristic values we have so far obtained.

Each action’s heuristic value  $h$  is composed of two parts: one that measures the desirability of the current perceptual state, and another that measures the change in desirability from the previous state to the current state. The relative importance of  $h$ ’s two constituents is controlled through learning rates  $\alpha$ , which is a multiplier to the value of the current state, and  $\beta$ , which is a multiplier to the value of the most recent change in state. High values of  $\alpha$  will make the maintenance of good perceptual state relatively more important; high values of  $\beta$  will make positive-valued state change relatively more important.

The learning algorithm used for each level can be defined as follows:

1. Randomly initialize  $m$
2. Initialize heuristic value and occurrence counter ( $\forall p \in P$ )  $h(p) = 0, n(p) = 0$
3. Repeat until convergence:
  - (a) Get perceptual input  $p$  from sensors
  - (b) Perform action  $m(p)$
  - (c) Get resulting perceptual input  $r$  from sensors
  - (d) Adjust heuristic value
 
$$h(p) = \frac{n(p)}{n(p)+1}h(p) + \frac{1}{n(p)+1}(\alpha v_r + \beta(v_r - v_p))$$
  - (e) Update occurrence counter
 
$$n(p) = n(p) + 1$$
  - (f) if  $h(p) < threshold$  then replace  $m_p$  by a randomly chosen action  $q \in A$  and reinitialize  $h(p)$  and  $n(p)$

At the obstacle-detection and obstacle-avoidance levels,  $v_p = 1$  if  $p$  represents a perception where the robot is not crashing into an obstacle (*i.e.* no bump sensor is active) and  $v_p = -1$  otherwise. Because changes in the value of the perceptual state here tell us little relative to the knowledge of whether the robot is running into a wall, we set  $\alpha$  to 1 and  $\beta$  to 0.

At the beacon-following level,  $v$  is positive for the case where the beacon is seen by the front infrared

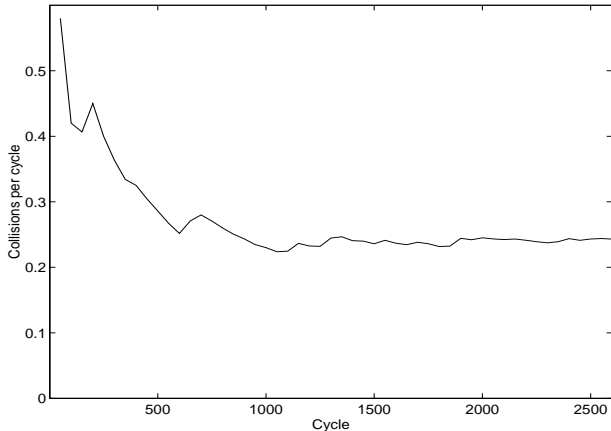


Figure 4: **Obstacle Detection.** The plot shows the average collisions per cycle as a function of the number of cycles.

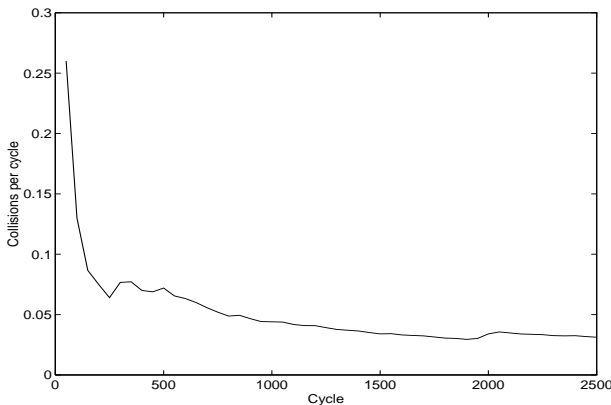


Figure 5: **Obstacle Avoidance.** The average collisions per cycle plotted as a function of the number of cycles.

detector, negative if it is seen by the back infrared detector and zero if it is seen by the side detectors. Here we consider the learning rates  $\alpha$  and  $\beta$  equally important, so we set them each to  $\frac{1}{2}$ . Note that leaving the value of  $\beta$  at 0 for this level would unfairly reward behaviors that keep the robot pointed more-or-less toward the goal but do nothing to advance it in that direction.

## 5. Experimental Results

In our experiments, the robot runs through the three levels of behavior, first learning to detect collisions with obstacles, then learning to avoid such collisions, and, finally, learning to navigate from goal to goal. Once the algorithm obtains adequate performance as specified by pre-set criteria, it switches behaviors and

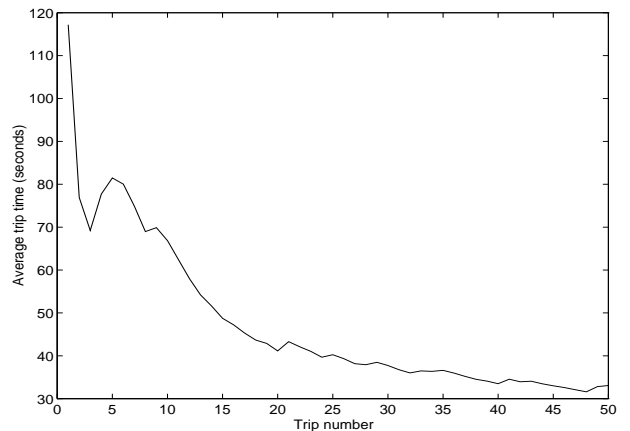


Figure 6: **Beacon Following.** The plot shows the average time per trip from one beacon to the other as a function of the trip number.

begins learning at the next level. For example, when, in the obstacle-detection behavior, the frequency with which the robot collides with obstacles drops below a given threshold, the algorithm proceeds to the obstacle-avoidance behavior.

For each behavior, we plot performance against time. Figure 4 plots collisions per robot control cycle versus time for the lowest level behavior, where the robot can feel but not see obstacles; figure 5 plots collisions-per-cycle versus time for the second level behavior, where the robot can both feel and see obstacles; and figure 6 plots average time-per-trip against trip number during the beacon-following behavior.

Figure 4 shows the performance of the robot in the obstacle-detection behavior. Collisions-per-cycle drop sharply until it reaches a stable value of approximately .25, beyond which point the (blind) robot cannot improve. After a few hundred cycles, the robot has learned the appropriate actions to take when it crashes into an obstacle. Once the robot has achieved a good level of performance in the obstacle-detection behavior, it switches to the next level behavior, obstacle-avoidance. The results for this behavior are shown in figure 5. Collisions-per-cycle again drop sharply, starting this time with the final value from the first behavior, and eventually reaching a new minimum of about .05. As in the previous case, after a few hundred cycles the robot successfully learns a policy that results in significantly fewer collisions. It should be noted that given the finite turning radius of the robot and the cluttered environment, collisions cannot be completely eliminated.

Figure 6 shows the results of beacon-following, the highest level behavior. The graph plots the average time spent by the robot on a trip between the beacons

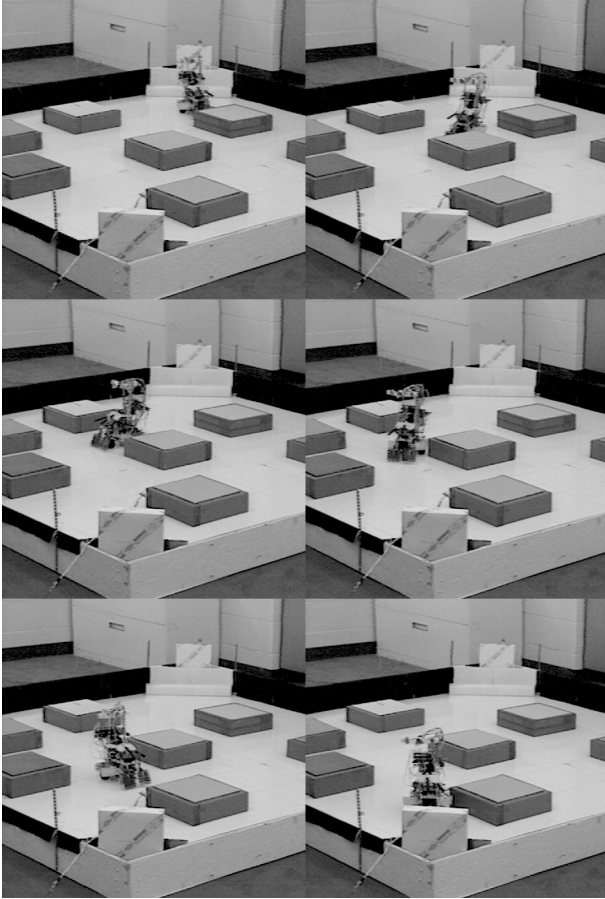


Figure 7: The Obstacle Avoidance/Beacon Following Behavior of the Robot after Hierarchical Learning.

as a function of time. As in the other behaviors, it can be seen that the robot quickly learns a policy that successfully performs the task (in this case, the task of homing to the location of the goal beacon).

Figure 7 depicts the behavior toward the end of the learning period, when all three behaviors are active.

## 6. Conclusions

We have shown that a simple heuristic hillclimbing strategy can be effectively used for learning useful reactive behaviors in an autonomous mobile robot. Our method results in considerable savings of memory space over other learning methods such as genetic programming, reinforcement learning and neural networks since we require the storage of only a single policy and a small history of perceptions.

The reduction in dimensionality allowed by hierarchical task decomposition speeds convergence of the learning algorithm. Constituent behaviors were quick to converge to an optimal value – typically, in less than half an hour. We believe such decompositions

can be found for most complex tasks.

The algorithm's quick convergence and low resource requirements make it ideal for implementation on real robots.

Possible future work includes further experiments regarding integration of additional behaviors and autonomous learning of coordination among behaviors [6].

## References

- [1] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–22, April 1986.
- [2] M. Colombetti and M. Dorigo. Learning to control an autonomous robot by distributed genetic algorithms. In J. A. Meyer, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, pages 305–312. Cambridge, MA: MIT Press, December 1992.
- [3] D. Gachet, M. Salchis, L. Moreno, and J. R. Pimentel. Learning emergent tasks for an autonomous mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 290–297, 1994.
- [4] B. J. A. Krose and M. Eecen. A self-organizing representation of sensor space for mobile robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 9–14, 1994.
- [5] M. A. Lewis, A. H. Fagg, and A. Solidum. Genetic programming approach to the construction of a neural network for control of a walking robot. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992.
- [6] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *Proceedings of the 1990 AAAI Conference*, 1990.
- [7] D. Pierce and B. Kuipers. Learning hill-climbing functions as a strategy for generating behaviors in a mobile robot. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 327–336, 1991.