HIERARCHICAL LEARNING OF REACTIVE BEHAVIORS IN AN AUTONOMOUS MOBILE ROBOT

Olac Fuentes
Laboratorio de Inteligencia Artificial
Centro de Investigación en Computación
Instituto Politécnico Nacional
México D.F., México
e-mail: fuentes@jsbach.cic.ipn.mx

Rajesh P. N. Rao and Michael Van Wie
Computer Science Department
University of Rochester
Rochester, New York
U.S.A.
e-mail: {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 simple hillclimbing routine that attempts to find the optimal transfer function from perceptions to actions for each behavior. We present experimental results which 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.

Keywords: Robot learning, behavior-based robotics, robot navigation.

1 INTRODUCTION

Traditionally, the task of developing a sensorimotor control architecture for a situated autonomous robot was left to the human programmer of the robot. Prewiring robot behaviors by hand however becomes increasingly complex for robots with 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 considerable simplification can be achieved by employing a *hierarchical behavior-based decomposition* of the control architecture as originally suggested by Brooks [1]. In addition, it is desirable in many cases to endow the robot with the ability to adapt its constituent behaviors online in response to environmental stimuli by allowing it to *autonomously learn* the transfer function mapping sensory input into motor commands.

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, [6]). 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 extremely difficult to simulate even when there is complete knowledge. For these reasons, we believe that in order for the learned skills to be applicable by 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 in 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 be-

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

havior. 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 by using a simple heuristic hillclimbing technique.

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 tripodal 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.

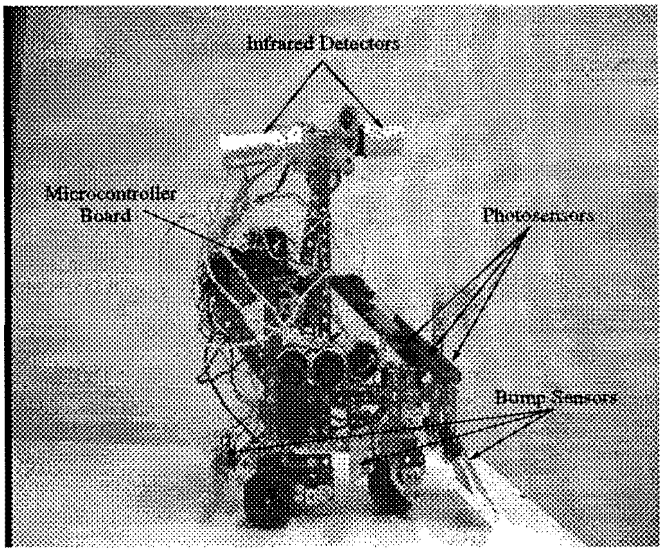

Figure 1: The robot used for the experiments.

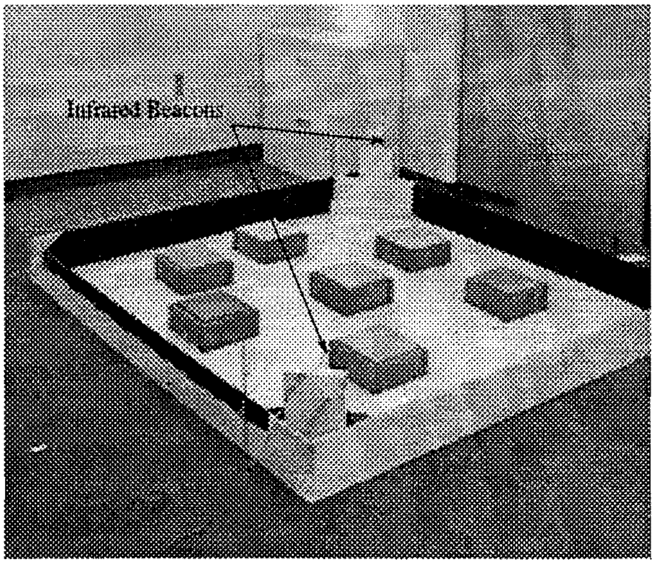

Figure 2: The robot arena

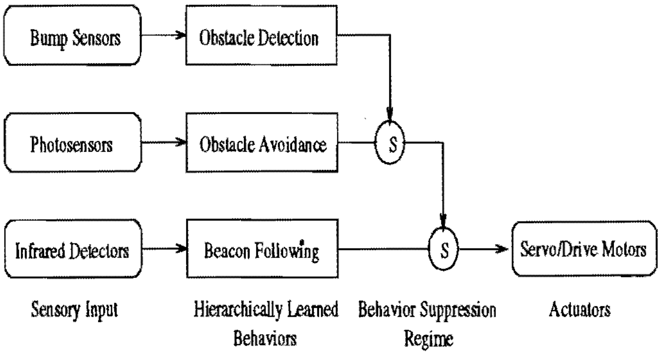

Figure 3: Block diagram of the robot control architecture

3 Behavior-based task decomposition

Since the robot is equipped with twelve different 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 extremely cumbersome and error-prone.

One way of circumventing this "curse of dimensionality" is to divide the given task into several layers of control such that the first layer consists of an elementary level of performance (say avoid continuous contact with obstacles), with each subsequent layer improving upon the performance obtained by the previous ones. If we chose this partition carefully, we can also arrange things in such a way that the first layer uses only a subset of the sensors available and each subsequent layer uses a superset of the sensors used by the lower layers. This is reminiscent of Brooks' subsumption architecture [1]. Figure 3 illustrates the simple three-level hierarchical architecture used in our robot.

Hierarchical partitioning of the sensory space allows the robot to learn the sensorimotor mapping corresponding 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 relying on simulation. By not relying on simulation we avoid the danger of learning a policy that works well only in simulation and can not be transfered 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 → action*, the method can be implemented using very little memory. This is in contrast to some machine learning techniques recently applied to mobile robotics (for example, genetic programming [4], reinforcement learning [2], and neural networks [3]) that usually require the storage of considerable amounts of information.

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

To every perception p we also assign a numeric value v_p measuring the desirability or "goodness" of the situations were that perception normally occurs. For example, a perceptual input indicating one or more pressed bump sensors has a low v, since its occurs in the undesirable situation when the robot crashes into an obstacle, while having no bump sensors pressed will have a high v, since 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 perceptions 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 heuristic value h and replace those entires in the policy that are judged to be inadequate (*i.e.* for which h falls under a prespecified threshold.)

The heuristic hillclimbing 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 $h(p) = \frac{n(p)}{n(p)+1} h(p) + \frac{1}{n(p)+1} (\alpha(v_r - v_p) + \beta v_r)$
   (e) Update occurrence counter $n(p) = n(p) + 1$
   (f) if $h(p) <$ *threshold* 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.* none of the bump sensors is depressed) and $v(p) = -1$ otherwise.

At the beacon-following level, v is positive for the case where the beacon is seen by the front infrared detector, negative if it is seen by the back infrared detector and zero if it is seen by the side detectors.

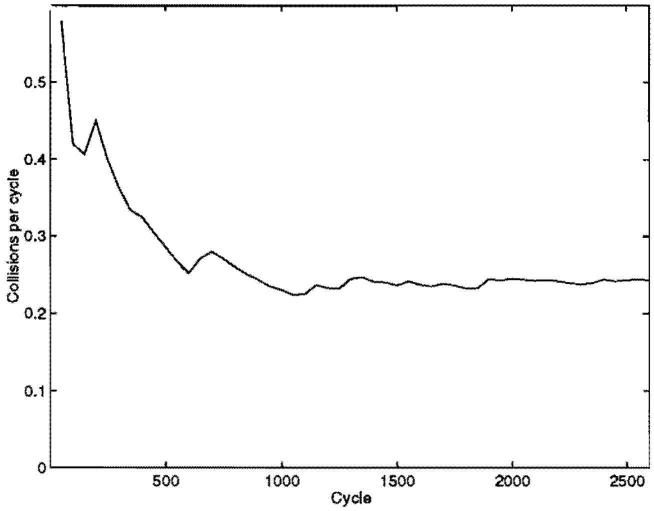

Figure 4: **Obstacle Detection.** The plot shows the average collisions per cycle as a function of the number of cycles.

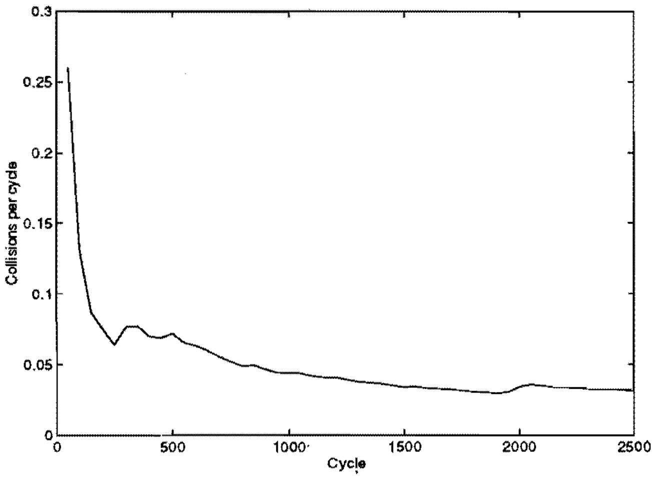

Figure 5: **Obstacle Avoidance.** The average collisions per cycle plotted as a function of the number of cycles.

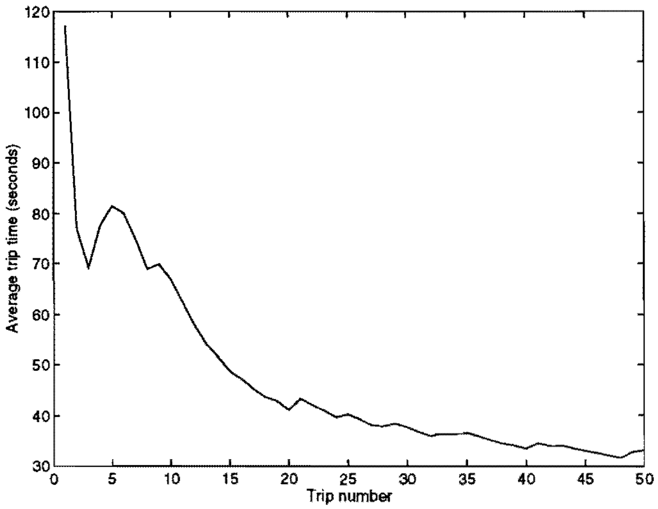

Figure 6: **Beacon Following.** The plot shows the average time per trip from one beacon to the other as a function of the trip number.

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 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 behavoir, 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 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

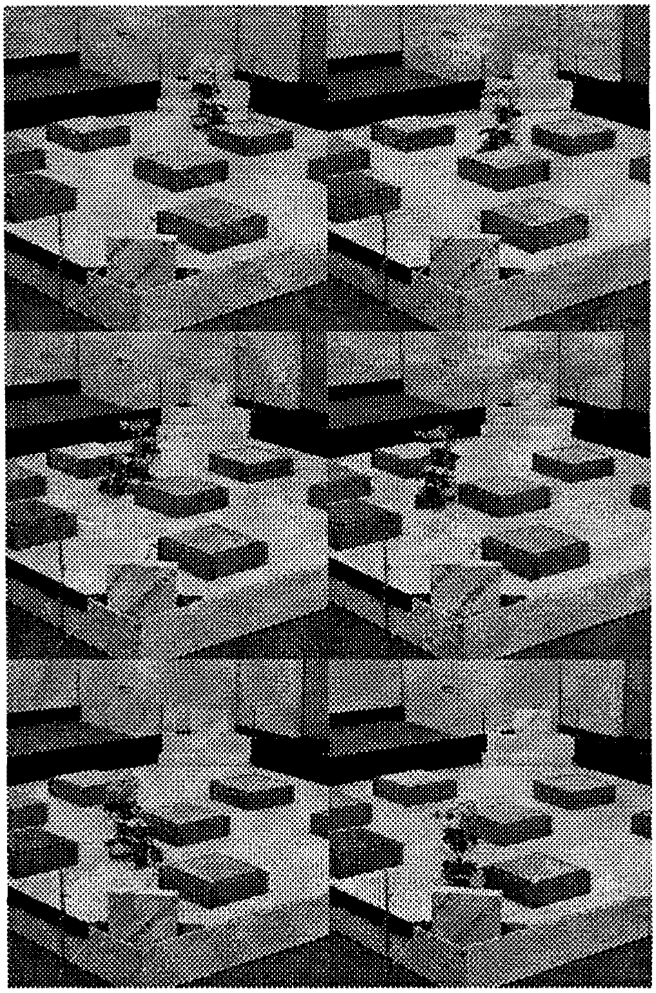

Figure 7: The Obstacle Avoidance/Beacon following behavior of the robot after hierarchical learning.

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 small history of perceptions for determining credit assignment followed by a subsequent stochastic change in the perception-to-action mapping.

Current work involves further experiments regarding the assignment of credit to vector elements, integration of additional behaviors, and possible autonomous learning of coordination between behaviors (cf. [5]).

References

[1] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–22, April 1986.

[2] D. Gachet, M.A. Salchis, L. Moreno, and J. R. Pimentel. Learning emergent tasks for an autonomous mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots an Systems*, pages 290–297, 1994.

[3] Ben J. A. Krose and Marc Eecen. A self-organizing representation of sensor space for mobile robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots an Systems*, pages 9–14, 1994.

[4] 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.

[5] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In *Proceedings of the 1990 AAAI Conference*, 1990.

[6] David Pierce and Benjamin Kuipers. Learning hillclimbing functions as a strategy for generating behaviors in a mobile robot. In *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behavior*, pages 327–336. Cambridge, MA: MIT Press, 1991.

Olac Fuentes. Recived his Ph. D. in Computer Science from University of Rochester in May of 1997. His research interests include machine learning, robotics and computer vision. Dr. Fuentes is a profesor at Centro de Investigación en Computación, of the Instituto Politécnico Nacional in Mexico City, Mexico.