A Situated Vacuuming Robot

**D. Bruemmer, R. Dickson, J. Dilatush, D. Lewis, H. Mateyak
M. Mirarchi, M. Morton, J. Tracy, A. Vorobiev, and L. Meeden**

Computer Science Program, Swarthmore College
Swarthmore, PA 19081
meeden@cs.swarthmore.edu

We undertook this project as an opportunity to explore design ideals of the embodied approach. At our disposal was a Pioneer robot and its Saphira software (ActivMedia 1996). Similar to subsumption architecture (Brooks 1986), the Saphira software tackles the dilemma of how to implement layered control design within a system which is inherently centralized. Brooks saw each layer as a simple and almost independent computational entity and, likewise, Saphira allows us to create a hierarchy of behaviors that each have the capacity to function simultaneously and yet asynchronously. Just as Brooks proposed a means by which one level can subsume a lower level by inhibiting its output, so behaviors can each be assigned a priority.

The main strength of Saphira's behavioral approach is not its ability to subsume or inhibit, but rather its capacity to blend behaviors. Non-conflicting behaviors run simultaneously and independently. If behaviors do conflict, their output can be combined in a number of ways. For example, if *Turn90Degrees* has highest priority and gives commands on the turn channel and *Constant_Velocity* is of lower priority and gives commands on the speed channel, then both behaviors will run completely independently. However, if *Turn90Degrees* gives commands on the speed channel, those commands override the activity of *Constant_Velocity.* If instead, both *Constant_Velocity* and *Turn90Degrees* have equal priorities then their activities will be blended, allowing smooth transitions between them.

Vacuuming Strategies

We are considering three different approaches to achieving full coverage for a given room. One approach utilizes genetic programming to generate an algorithm that effectively vacuums the room, while the other two focus on designing behaviors for the robot to traverse the floor in a pre-specified pattern (the North/South and Concentric Squares methods). We plan on comparing all three approaches in a variety of room environments to measure their relative utilities.

North/South. The robot begins in the southwest corner of the room, facing north. The robot then traces a path of north-south lines through the room, while avoiding large obstacles (see Figure 1). When the robot is in its *North* behavior, it simply travels north in a straight line until it either senses an obstacle directly in front of it or senses that it has just passed an object immediately on its left. If it senses an obstacle directly in front of it, the robot moves east one robot's width, turns to face south, and enters its *South* behavior. During *North*, if the robot senses that it has just passed an object immediately on its left, the robot switches to a *TraverseObstacle* behavior in which it turns to face east and follows the top edge of the object on its left. When the robot senses that this edge has dropped off, it turns to face north and returns to the *North* behavior, vacuuming the area of the room that lies north of the piece of furniture. During *South*, the robot moves due south until it senses an obstacle in front of it, in which case it moves east one robot's width, faces north, and changes to the *North* behavior. In designing this algorithm, we attempted to rely as little as possible on computing the robot's estimated position within a map. Instead, we used only the robot's sensor readings as criteria for determining the robot's path.

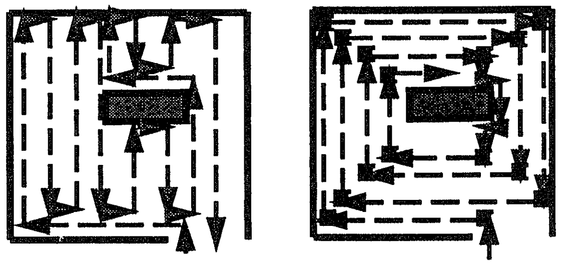

Figure 1: Illustrations of the North/South (left) and Concentric Square (right) approaches.

Concentric Squares. The first step of this method is to represent the structure of a room in the internal coordinate system of the Saphira interface. The next step is to locate a series of goals in the coordinate system. The goals should be positioned so that, if the robot were to move in a straight line from each goal to the next goal in the series, it would cover all the space in the room (see Figure 1, note that the goals are depicted as small squares). There are four behaviors that characterize the robot's action as it moves through the series of goals: *Avoid* turns the robot away from objects before it hits them, *Straight* turns the robot back to the straight line between its last goal and its next goal, *Seek* turns the robot toward its current goal and replaces the current goal with the next goal when the current goal is achieved, and *Wander* is the default behavior.

The combination of these behaviors leads to the following action sequence in the case of an obstacle: first, the robot slows and turns to avoid collision; next, it wanders until it is free of the obstacle; finally, it resumes the course it was following before it encountered the obstacle. This sequence of behaviors has the advantage that the space on the opposite side of an obstacle is not neglected once the obstacle has been avoided. The approach as a whole, however, is risky because of the inaccuracy of the robot's dead reckoning system.

Genetic Programming. In GP (Koza 1994), the idea is to first generate a collection of random programs, building from a pool of relatively simple functions, to control the robot's behavior. Each program is given a fitness—a measure of how well it performs. Akin to natural evolution, this population of random programs undergoes a simulated version of natural selection and reproduction to produce a new and hopefully better population of programs. In GP, the computer generates the programs rather than human designers. It is our hope that this approach may create innovative control strategies.

Special Considerations

Aside from designing algorithms to vacuum the floor, we have had to tackle a few other problems. Two of those problems are the inaccuracy of dead reckoning, and the difficulty of short distance obstacle avoidance using only the sonar readings of the Pioneer robot.

Dead Reckoning. The Saphira software keeps an estimate of the robot's global position. While good enough to correlate sonar readings over time, this estimate is not sufficiently accurate to keep the robot's position on a map over a long interval. Our code extends the accuracy of this estimate by comparing sonar readings with the arrangement of items near the robot on a map. First, the program attempts to fit a straight line to the point locations of recent sonar readings. It discards points until the remaining data can be fit with acceptably small error. If a fit is found without discarding too many points, the program assumes that a wall has been detected. It then scans the map, trying to

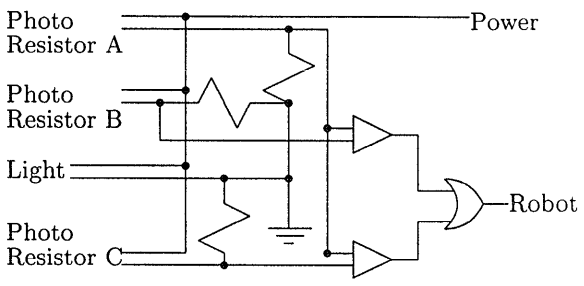

Figure 2: Circuit schematic

find a line segment with position and orientation, relative to the robot's estimated situation, most similar to that of the observed line. If the map segment found is similar enough, it is assumed to correspond to the observed segment. Any disparity between the two is therefore assumed to represent the difference between the actual and estimated position of the robot, and the estimate is adjusted accordingly.

Close-range obstacle avoidance. We found that the sonar sensors provided with the robot were insufficient to reliably detect objects within one foot of the robot. We decided to add visible light sensors based on photo resistors, instead of infrared, because of cost considerations. Eight sensor modules are attached to the robot, two per side. Each sensor module consists of three photo resistors, three standard resistors, a light source, two comparators, and an OR gate. All components are mounted on a vertical member attached to the side of the robot. One photo resistor is mounted facing up on the top of assembly. The other two photo resistors are mounted facing horizontally, at the top and bottom. The light is mounted horizontally in the middle. The components are then connected as detailed in Figure 2. (Photo Resistors A, B, and C correspond to vertical, top, and bottom.) Under standard lighting conditions this assembly can sense walls up to four inches away, under half light, the range extends to one foot.

References

ActivMedia, Inc. 1996. Pioneer 1 Mobile Robot Saphira Software Manual, Version 4.1.2.

Brooks, R. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, March 1986, 14–23.

Koza, J. R. 1994. Introduction to Genetic Programming. In *Advances in Genetic Programming*, 21–42. MIT Press, Cambridge, MA.