SWEEP STRATEGIES FOR A SENSORY-DRIVEN, BEHAVIOR-BASED VACUUM CLEANING AGENT

by
Keith L. Doty and Reid R. Harrison

Machine Intelligence Laboratory
University of Florida, Gainesville, FL

AAAI 1993 Fall Symposium Series
Instantiating Real-World Agents
Research Triangle Park, Raleigh, NC
October 22-24, 1993

ABSTRACT

In the Machine Intelligence Laboratory, University of Florida, we have built a small autonomous robot and programmed it to exhibit various reactive behaviors. The robot, named Gator, performs area coverage in an interior room by combining distinct behaviors. Gator has 26 sensors of which only 7 are used in the coverage algorithms examined here. Gator's behaviors allow it to avoid obstacles, follow walls, seek open areas and break out of confined areas. In the limited number of experiments performed here it appears that the best coverage algorithm among those tried consisted of a random walk with a probability of 0.05 for following a wall after each encounter with an obstacle. After 20 minutes about 85% of the floor space was covered with an efficiency close to 85%.

INTRODUCTION

Earlier work at the Machine Intelligence Laboratory, University of Florida, illustrated the application of swarm robots to materials handling in a manufacturing workcell [5]. The simulation results reported in [3] encouraged us to construct autonomous platforms to physically embody the theoretical model and test it under more realistic conditions. This effort is currently under way and will be reported elsewhere. Our mobile robot platform, while designed for a different application, appears to offer the capability to realize key behaviors of a vacuuming robot with regard to area coverage. Area coverage defines a general behavior and applies to a variety of problems [7]. Our report focuses on the area coverage problem. Construction of an autonomous vacuuming tool offers a challenging engineering task, but lack of a satisfactory area coverage algorithm will render such a tool ineffective.

Autonomous vacuuming presents a challenging task well suited for sensory-driven, behavior-based, reactive agents. Although our approach resonates with the Brooks paradigm [4], we do not use the subsumption architecture due to its restrictive hierarchy [2],[3],[6]. We have created an agent that achieves competency at a number of tasks through a synthesis of several reactive behaviors [6],[8].

Our robot exhibits several types of wandering behavior similar to those simulated and implemented by Anderson and Donath [1] with qualitatively corresponding results. Within this context, we have investigated area coverage algorithms for a single robot with combined reactive behaviors. Our robot does not yet use minimal maps, similar to the topological maps described by Mataric [9] but it does have one behavior, namely *claustrophobia*, that incorporates short-term memory or an *ephemeral state* , a term coined by Gat[6].

AUTONOMOUS VACUUM CLEANING AGENTS

The assumed goal is to develop an autonomous mobile agent for vacuuming enclosed areas with obstacles. This assumption rules other types of solutions such as self-cleaning air-hockey floors with negative pressure at the holes instead of positive pressure. The next few paragraphs details further assumptions and the approach taken in this work toward the realization of the stated goal.

Assumptions about the Environment

We assume that the environment is a closed-off, interior room with a relatively smooth, level surface and that furniture in the room appears like obstacles to the proximity sensors. We have discovered that real chairs present a challenging problem to proximity sensors, especially pedestal office chairs with radiating legs. Vacuuming under and around such chairs presents considerable difficulties. We have chosen to side-step this difficulty by making the furniture-obstacle assumption.

Autonomous Vacuum Functional Requirements

A vacuum cleaning robot should, of course, clean all surface areas in the room not occupied by furniture. Our model does not include a furniture moving policy, and so those areas will not be touched. The robot should not damage furniture while performing its function and should optimize the parameters of time, energy consumption and capital investment.

Implementation Consideration

Even the reduced requirements provide challenging engineering and algorithmic problems. One must minimally consider

1. The vacuum tool platform construction, operation and power source,
2. The robot's processor architecture, sensor suite and interface structure, and
3. The robot agent's primitive behaviors and emergent functionality.

Our premise is that explorations into the primitive behaviors which lead to the emergent function of vacuuming can be separated from the other two problems. Consequently, one can explore appropriate vacuuming behaviors with a physically simpler autonomous agent.

Area Coverage

Sweeping corresponds to a type of wandering that will cause a robot to cover all parts of a room without missing any exposed areas. We perceive the sweeping behavior as a synthesis of simpler movement behaviors: following walls, traveling back and forth across open spaces, random wandering etc. In the experiments to be described later, these were precisely the behaviors used to provide area coverage with qualified success.

Limitations

IR detectors do not provide a reliable measure of absolute distance, nor, with the power levels used, long distance detection (less than 50 cm). Further, IR readings vary not only with distance, but also with the reflectivity of the obstacle. In spite of these limitations, our preliminary experience indicates IR detectors do provide the requisite sensor information for a vacuum-cleaning agent to avoid collisions with obstacles and to follow walls.

We limit our robot to one microprocessor to reduce hardware complexity and cost. Due to the limited processing and memory resources, our robot does not build sophisticated maps of its environment. Instead, it will worry only about objects in its immediate proximity or in an ephemeral state [6]. Its various behaviors will be combined to produce floor vacuuming, i.e. area coverage, as an emergent functionality.

Vacuum Cleaning Robots

Our primary assumption is that a vacuum cleaning robot must transport the vacuum tool on a sensory-driven, behavior-based mobile platform. While vacuum cleaning with swarm robots represents a viable alternative, our goal to physically implement the area coverage paradigm limits our initial efforts to a single platform. We are in the process of building several other platforms and will soon explore multi-agent approaches to the problem as well.

We call our small mobile robots Munchkins (Toto, this doesn't look like Kansas!). Munchkins are constructed from LEGO™ building blocks which provide great flexibility, sophistication and ease of mechanical design. The Munchkin in this paper (Figure 1), named *Gator*, is controlled by one MC68HC11 microcontroller and uses less than 2 Kbytes of code to accomplish its tasks.

Two bi-directional DC motors drive, respectively, the left and right drive tracks of the robot. Gator travels about 0.345 ft /sec (105 mm/s) and sweeps 3.125 inches and so it covers about 107 square feet of area in 20 minutes.

Gator also possess a 2-DOF arm capable of grasping small objects and lifting them 7 cm above the ground. The manipulator capability was not used in this work.

Gator measures 27 cm long, 12 cm wide, and 20 cm tall (see Figure 1). It is powered by six AA NiCd batteries and can run for approximately 45 minutes on a charge.

Gator supports a variety of sensors. Table 1 itemizes the sensor suite available. The robot is outfitted with two forward-pointing spring whiskers which serve as flexible contact sensors (Figure 1). We have also installed IR sensors around Gator's waist to provide proximity detection so that Gator can actually avoid contact with objects detected in its path. In the experiments reported here, only the Proximity and Dead Reckoning sensors were employed.

Table 1 Munchkin Sensor Suite

Sensor Type	Function	Location	Number
Infrared (IR)	Proximity	Periphery	7
Infrared (IR)	Grip Detection	Gripper Claw	1
Infrared (IR)	Beacon Detection	Front	8
Contact Switches	Collision Detection	Four corners	4
Shaft Encoders	Dead Reckoning	Front wheels	2
Limit Switches	Actuator stops	2-DOF Gripper	4

PRIMITIVE BEHAVIORS

Machine behaviors constitute the central focus of this research. The following behaviors, which have all been reported elsewhere by the authors and others were taken as primitives. Our goal was to determine how effective these behaviors are in solving the area coverage problem.

Collision Avoidance

For collision avoidance two levels of proximity sensitivity were employed. High proximity sensitivity makes the robot *shy* away from obstacles and walls and, hence, to *favor* open areas while the low proximity sensitivity makes the robot *bold* and allows it to traverse narrow passages and explore more confined areas.

For either low or high sensitivity, obstacle detection derives from the sensor-based conditional:

if IR_detect = *true* *then* obstacle := *true*

We call the derived signal "obstacle" an indicator. Elsewhere, one notes the name "virtual sensor". To us the semantic content of "virtual sensor" in current usage is either too broad or ill defined. We admit the term is appealing, however, and we would like for the research community to settle on a precise definition or discard the notion.

Random Sweep

Random sweep constitutes the base line coverage algorithm. Essentially, the robot moves forward until an obstacle is detected, at which time it turns at a randomly chosen angle between $\pm 180^{\circ}$. The algorithm for random sweep is,

```
/* Random Sweep */
do {
   if  obstacle = false
    then  go forward
    else  turn Random_Uniform( -180°, 180°)
   }
```

Wall Following

Side viewing low and high sensitivity IR proximity detectors allow the robot to follow walls. The robot attempts to stay between the low and high sensitivity distances. *Wall following* permits the robot to circle furniture and follow along the room walls. Walls the robot follows can be on either the right or left side of the robot.

Plow Sweep

This *plow sweep* algorithm attempts to drive the robot about the room in manner corresponding to plowing a field. However, when it encounters an object it turns at right angles and attempts to *plow* again in the new direction.

Claustrophobia

If the robot mode corresponds to the high proximity sensitivity state and five or more object detections occur before the robot moves one foot, the robot switches to the low proximity sensitivity. By switching to the low sensitivity level, the robot can escape its confinement by enabling it to pass through narrow passages. The objective is too keep the robot from spinning around in place in confined areas. This behavior is called *claustrophobic* because the robot does not tolerate confinement.

The conditional for *claustrophobia* equals,

if obstacle_count = 5 *and* wheel has not moved forward 1 foot *then* claustrophobia := *true*

Combined Behaviors

The primitive behaviors alone do not adequately provide area coverage, although *random sweep* alone provides impressive coverage. Here we described combined behaviors which we have tested. In the next section we will discuss specific experiments utilizing these behaviors.

The following pseudo-code indicates *random sweep with wall following*: This combined behavior typically exhibits random sweep behavior. However, when an obstacle is detected, the robot, with probability of p, will follow the object as a wall for a random distance d before resorting back to *random sweep* behavior. In our experiments $p = 0.05$ and $6 \leq d \leq 20$ feet.

```
\* CB1 (Combined Behavior One) : Random Sweep with
Wall Following *\
do {
   if  obstacle = false
    then  go forward
    else
      Random Select
         1-p : turn Random_Uniform( -180°, 180°)
         p :  {turn Random_Binary(-90°,90°) ;
                 follow wall for
                 Random_Uniform( 6 feet, 20 feet)  }
   }
```

In *random-sweep-with-wall-following* behavior, the proximity sensors may be operated in low or high

sensitivity mode. In low sensitivity mode the robot appears *bold* and explores more confined areas. In the high sensitivity mode the robot *shys* away from confined areas and stays in the open. The next behavior combines both features. The robot is shy $\frac{1}{3}$ of the time in our experiments which employs this behavior.

```
\* CB2: Random-sweep alternating between shy and
bold-with-wall-following *\

do {
   /*Bold with wall following*/
    proximity_sensitivity := low
   repeat
      if obstacle = false
      then go forward
      else
       Random Select
         0.95: turn Random_Uniform( -180°, 180°)
         0.05: {turn Random_Binary(-90°,90°);
                follow wall for
              Random_Uniform( 6 feet, 20 feet) }
   until 2 minutes elapse

   /*Shy*/
    proximity_sensitivity := high
   repeat
      if obstacle = false
      then go forward
      else turn Random_Uniform( -180°, 180°)
   until 1 minute elapses
   }
```

The previous behavior tended to trap the robot for a minute when switching from low to high proximity sensitivity in confined areas. To avoid this waste of time and energy, a claustrophobic short-term memory or ephemeral state behavior was added. The modified algorithm is listed below.

```
\* CB3: Random sweep alternating between 1) shy
behavior with claustrophobia and 2) bold behavior with
wall following *\

do {
   /*Bold with wall following*/
    proximity_sensitivity := low
   repeat
      if obstacle = false
      then go forward
      else
       Random Select
         0.95: turn Random_Uniform( -180°, 180°)
         0.05: {turn Random_Binary(-90°,90°);
                follow wall for
              Random_Uniform( 6 feet, 20 feet) }
   until 2 minutes elapse

   /*Shy with claustrophobia*/
    proximity_sensitivity := high
   repeat
      if obstacle = false
      then go forward
      else turn Random_Uniform( -180°, 180°)
   until 1 minute elapses or claustrophobia = true
   }
```

Table 2 lists the actual size of the program, in bytes, of the different behaviors. In all cases this code includes software for monitoring the sensors, processing the sensor data and controlling the motors.

Table 2 Code Size for Gator's Behaviors

Behavior	Bytes
Random Sweep	838
CB1: Random Sweep with Wall Following	1027
Plow Sweep	916
Plow Sweep with Wall Following	1162
CB2 = shy + bold CB1	1138
CB3 = CB2 with Claustrophobia	1167

Area Coverage Experiments

We performed a total of 20 area coverage experiments, of which 5 are reported here. The area to be covered by Gator equaled a 10' × 10' walled region of the Machine Intelligence Laboratory floor space containing five obstacles representing furniture (Figures 2-6). The walls and obstacles were painted white to provide uniform IR readings. While not totally necessary, it did make the experiments more manageable. In more realistic settings, the IR sensor algorithms would have to deal with surfaces with significantly different reflectivity.

We used open-shutter photography to record Gator's travels. A green LED attached to the top-central part of Gator traces out a light path on the photograph as Gator moves about. These Gator alleys appear as yellow traces on all but Figure 2, where we had used a red LED. Two red LEDs, one on each side of the front undercarriage, illuminate the vacuum sweep area. The red LEDs generate a red paint-brush effect in the image, indicating the area swept. Although partially blocked in some directions, the red-painted area provide an effective way of determining the total amount of floor space covered by the robot.

To properly view the photographs, imagine the yellow light traces as suspended above the floor. Further, imagine the projection of a light trace onto the floor as falling in the middle of its corresponding "red-paint" sweep area. The green areas on the floor represent those spots not swept by Gator during the experiment. The arcs of light on the

photographs indicate that Gator's proximity sensors have detected an obstacle and forced a turn.

During the 20 minute experiments Gator covers about 107 square feet of surface of area. The exposed area equals 93 square feet. So the ratio of the red-paint area to the green area times $\frac{93}{107}$ provides a respectable measure of sweep efficiency.

Qualitative Analysis of the Experiments

The photograph in Figure 2 indicates Gator's behavior with the proximity detectors at high sensitivity. Elapsed time of the experiment : 20 minutes. The left side of the room, which is accessible via narrow passageways between furniture items, and significant areas along the right wall and the lower wall have been totally missed. Gator covered about 40 % of the room , hence, the efficiency of this run approximately equals $40\ \frac{93}{107}\ \% \approx$ 35%.

After several other experiments we opted to employ two strategies to get increased coverage: we 1) decreased the sensitivity of the proximity sensors used for collision detection and 2) incorporated *wall following* behavior. The photograph in Figure 3 illustrates a run at low proximity sensitivity of the combined behavior algorithm *CB1* (*random-sweep* with *wall following*). The behavior executed for 20 minutes. Of our 20 runs this was the best. Gator accessed all regions of the room and covered about 80% of the exposed surface area with an efficiency of 80 $\frac{93}{107}\ \% \approx 70\%$.

The experimental results in Figure 4 shows the effects of combining the strategies used for the experiments illustrated in Figures 1 and 2 (*CB2)*. Intuitively, we believed the *shy* behavior would cover the open areas well and the *bold* behavior the more confined areas. The weak results, however, were not anticipated. Observe the bright circles of light. They indicate that Gator was trapped in a confined area. We deduced that this phenomena results when, in a confined area, Gator switches from low to high proximity sensitivity. The picture dramatically implies the inefficiency of this algorithm by the large percentage of green area. Since each circle indicates up to a minute of lost sweeping time, it was important to eliminate the behavior leading to this result. This was done by adding the *claustrophobic* behavior to *CB2* to create *CB3*. At this point we still believed combined *shy* and *bold* behavior to be a good basis for an efficient covering behavior.

As a base line for comparison we ran the *random sweep* behavior for 40 minutes (Figure 5). Next, we ran *CB3* for 40 minutes. No visible spinning in one spot can be seen in the photograph, so *claustrophobia* eliminated the trapped behavior in confined areas. The marginally better performance of *CB3* over the *random sweep* , however, surprised us. The other surprising result was that the coverage was not much better than the 20 minute run of *CB1*. Another run of *CB1*, not shown here, was far less effective, so we consider the experiment of Figure 3 an exceptionally lucky run.

Although we have not taken a statistically significant number of runs, we also have not attempted to measure the coverage more precisely than just visual estimation.

CONCLUSIONS

We approached the problem of autonomous vacuuming from a low-level standpoint, using a robot equipped with simple, low-cost sensors and an 8-bit microcontroller with 2K of EEPROM for software. We set up a 10' x 10' area containing five obstacles to serve as an environment for the robot during a series of tests. Each test recorded the movement of the robot over a specific time interval.

According to our estimations, a purely random sweep of movement covers the room with about 60% efficiency. The addition of a wall following behavior, triggered with probability 0.05 at each obstacle detection, seems to increase performance to about 70 % efficiency. Increasing the sensitivity of the object detectors traps the robot in wide open spaces. Decreasing the sensitivity of the object detectors allows the robot to wander through narrow corridors into new open spaces. The wall following behavior also seems to "drag" the robot to distant parts of the room. In an attempt to mix thorough open space coverage with full-room coverage, we combined these behaviors. This strategy provided coverage that was marginally better than the random walk behavior. We implemented a *claustrophobia* indicator that successfully freed the robot from narrow corridors and corners.

To reduce the difference between the 70% efficiency of area coverage with the techniques developed here and the 95% or better efficiency of a human with a Hoover, will probably require more sophisticated sensors, short-term memory based strategies and sensors providing feedback information about how successfully the task is being performed.

REFERENCES

[1] Tracy L. Anderson and Max Donath, "Animal behavior as a paradigm for developing robot autonomy," in Designing Autonomous Agents edited by Pattie Maes, MIT Press, 1990.

[2] Akram A. Bou-Ghannam and Keith L. Doty, "A CLIPS Implementation of a Knowledge-Based Distributed Control of an Autonomous Mobile Robot," SPIE Proceedings, Orlando, Florida, April, 1991.

[3] Akram A. Bou-Ghannam, Intelligent Autonomous Systems: Controlling Reactive Behaviors with Consistent World Modeling and Reasoning, Ph.D. Dissertation, University of Florida, 1991.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.