Making a Clean Sweep: Behavior Based Vacuuming

Douglas C. MacKenzie Tucker R. Balch

Mobile Robot Laboratory
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280 U.S.A.
doug@cc.gatech.edu

Position Paper for AAAI Fall Symposium,
Instantiating Real-World Agents

I. Introduction

This position paper discusses how the techniques and building blocks in use at the Georgia Tech Mobile Robot Lab could be used to construct an autonomous vacuum cleaner. Previous research at Georgia Tech using the Autonomous Robot Architecture (AuRA)[2, 4] has demonstrated robust operation in dynamic and partially modeled environments[3, 12]. Although much of this work has been deployed and tested within the manufacturing domain, the inherent modularity of the reactive component of the AuRA architecture facilitates major retargeting of the domain with minimal new development[1]. This paper will focus on how the existing work at Georgia Tech could be enhanced to perform the vacuuming task.

II. Primitive behaviors

A new language which specifies how behaviors may be described and coordinated has been developed at the Georgia Tech Mobile Robotics Laboratory. See [13] for a more complete discussion. That grammar is used here to describe a vacuuming robot.

Motor schemas are the basic action units in AuRA. Motor schemas instantiated with one or more embedded perceptual schemas are primitive sensorimotor behaviors. A primitive sensorimotor behavior b_1 is a reactive channel converting sensations to actions. Figure 1 shows b_1, a primitive behavior. It uses one sensation stream, a perceptual module, and a motor module to generate a stream of actions. The sensor samples the environment and generates a stream of sensations S. New sensations are generated when required by perception modules. The perceptual module P acts as a feature detector. It extracts features using sensations from one or more sensors S_i and generates a stream of action-oriented features F. Features are generated as required by motor modules. A motor module M combines the input from one or more perceptual modules F_i to generate a stream of actions A as output.

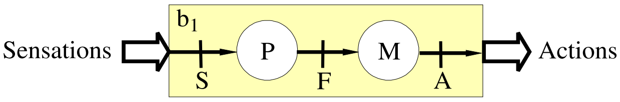

Fig. 1. A simple primitive behavior

Several primitive sensorimotor behaviors are used in the construction of the reactive control for the vacuuming robot. Some of these behaviors have been developed and evaluated in other work [1, 2]. A brief overview of each will be presented here.

- **Avoid-static-obstacle(find-obstacle)**
  The avoid-static-obstacle behavior is used to keep the robot at a safe distance from non-threatening obstacles. A repulsive vector is generated from obstacles perceived to be closer than a threshold distance to the robot. The embedded perceptual module find-obstacle reports the locations of obstacles within sensor range.

- **Noise()**
  The noise behavior generates a vector with a random direction. The strength and persistence of the noise are set as parameters. The randomness helps prevent the robot from getting stuck in local minima or maxima, acting as a sort of reactive "grease". Noise does not require any sensory input.

- **Move-to-goal(detect-dirt)**
  The move-to-goal behavior generates a vector towards a specific location. For the vacuuming robot the embedded perceptual module detect-dirt directs the robot towards unvacuumed areas.

- **Probe(detect-current-heading)**
  When active, this motor behavior keeps the robot moving along its current direction.

- **Vacuum-dirt**
  When vacuum-dirt is activated, the vacuum motor turns on. Any dirt under the robot is sucked up. In simulation, a grid-based map is marked to indicate that the local area has been vacuumed.

Four perceptual modules are utilized by the motor behaviors and coordination processes. These perceptual modules may be embedded within the motor behaviors described above, or they may be combined with a null motor behavior so that their output may be utilized by a coordination process.

- **Detect-dirt**
  A perceptual module which reports if any areas in need of vacuuming are visible. Here it is implemented using a grid-based map. To simulate the limitations of real sensors, *Detect-dirt* can only find dirt within a limited range. Move-to-goal uses detect-dirt to set a goal to move to.

- **Over-dirt**
  A perceptual trigger which reports if the area under the robot is dirty. The simulation implementation uses the grid-based map described above. Over-dirt is utilized by one of the coordination processes to ensure that useful vacuuming is taking place. If not, a transition to another behavior is made.

- **Find-obstacle**
  Find-obstacle reports locations of obstacles to the avoid-obstacle motor behavior.

- **Detect-current-heading**
  Detect-current-heading detects and reports the heading of the robot.

III. Assemblages of behaviors

The coordination between active motor behaviors defines the emergent behavior of the robot. Here we present a formal description of this coordination. The primitive sensorimotor behaviors will each be considered trivial assemblages. Assemblages can also be constructed from coordinated groups of two or more assemblages. This recursive definition allows creation of increasingly complex assemblages[11] grounded in the primitive sensorimotor behaviors (the trivial assemblages).

To generate a non-trivial overt behavior requires effective use of multiple primitive behaviors. We define the *coordination operator* @ as a generic mechanism that groups two or more assemblages into a larger assemblage. Equation 1 formalizes the usage of the coordination operator to form the assemblage A' from the group of $b_1, \ldots, b_n$ primitive sensorimotor behaviors and a coordination process $@_1$.

$$A' \longrightarrow [@_1 \; b_1, b_2, \cdots, b_n] \tag{1}$$

The coordination operator must consume the multiple action streams and generate a single action stream for the new assemblage (specific coordination processes are described below). This definition draws on the Gapps/Rex[8, 9, 10] idea of specifying reactive systems in a high level manner. However, it is a generic operator based on the RS assemblage construct[11], allowing recursive definitions and arbitrarily complex coordination procedures. Eventually the robot designer must specify instances of coordination operators. These specific instantiations are indicated with a subscript. Figure 2 shows how assemblage A_3 is constructed using the coordination operator $@_1$ and behaviors b_1 and b_2.

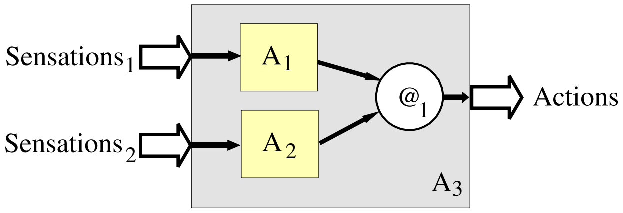

Fig. 2. Coordination operator creating assemblage A_3 from behaviors b_1 and b_2 and coordination operator $@_1$

Competition and cooperation are two example coordination processes. Competitive coordination uses an n to 1 multiplexer to pass the action stream for one assemblage through as the output for the coordinated group, effectively arbitrating among the competitive set of behaviors. The process of determining which of the n assemblages to select can use a variety of techniques. Examples include encoding a procedure in a finite state automaton, using activation networks, and assigning priorities to each stream. Architectures using competition mechanisms include spreading activation nets[14], and subsumption architecture[7].

Cooperative coordination uses a function $f(A_1, A_2, \ldots, A_n)$ to merge the output stream of each assemblage into one output action stream for the group. Examples include vector summation in

AuRA[4] architecture, and the weighted combination of inputs in neural networks. Arbitrary coordination operators can be constructed using combinations of these classes.

IV. The vacuuming task

When performing a complex task such as vacuuming, a robot's operation must conform to various situations and environmental conditions. Consider the case when the local area has been vacuumed, but distant areas are still dirty. The robot must have appropriate strategies for discovering dirty areas, moving to them and cleaning them. There might also be various vacuuming behaviors for different environments. When vacuuming an empty hallway, for example, it is appropriate for the robot to move rapidly, with little resources devoted to obstacle detection and avoidance. However, when operating in a dining room, slow deliberate motions are necessary to prevent inadvertent damage to expensive furnishings. It must also cope with dynamic objects (people, pets) and unpredictable environments (toys, shoes, etc).

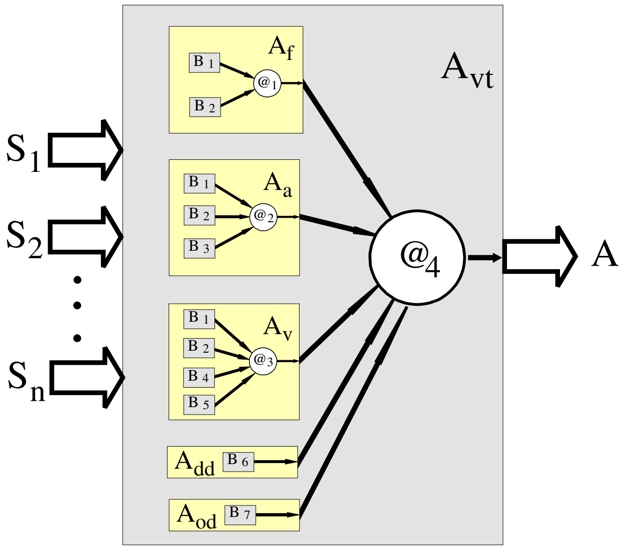

Fig. 3. The vacuuming assemblage

To manifest this behavior, we will define distinct operating states for the robot and use a competitive coordination operator to control the high-level process while using cooperative coordination within the low-level assemblages. The use of states allows greater optimization and specificity of individual operating configurations since a transition to a different operating state can be provided when the conditions required for the active state cease.

In other work we have developed a set of generic tasks for multi-agent teams to perform [5]. One such task is *graze*. To complete the task, the team of robots must move over a specified area while simultaneously collecting small objects which have been uniformly scattered about the environment. Robots able to perform this generic task could easily perform specific tasks such as mowing lawns, or vacuuming a house.

The approach taken in the multi-agent work is adapted here for the vacuuming task. We define three top level states of operation: forage, acquire and vacuum (additional states such as mop-floor could easily be added later). Figure 3 shows a graphical representation of the vacuuming task assemblage A_{vt}. It is constructed from the forage A_f, acquire A_a, and vacuum A_v assemblages using an FSA as the coordination operator $@_4$, to appropriately invoke A_f, A_a, or A_v. $@_4$ utilizes two additional assemblages, A_{dd} (detect-dirt) and A_{od} (over-dirt), as perceptual inputs to select the appropriate motor behavior.

- A_f **(Forage)**
  While in the **forage** state the robot explores its world looking for areas needing to be vacuumed. When the detect-dirt assemblage A_{dd} finds a dirty area, a transition to the **acquire** state occurs. The primitive behaviors of A_f are coordinated by the operator $@_1$, which sums their outputs. Behaviors active in the **forage** state are:
  - b_1 avoid-static-obstacles(find-obstacle)
  - b_2 noise()
- A_a **(Acquire)**
  While in the **acquire** state the robot moves towards the dirty area that has been discovered. Once that area is reached, or if the robot comes across a new dirty area, the over-dirt assemblage A_{od} causes a transition to the **vacuum** state. The coordination process $@_2$ sums the motor behavior outputs. Behaviors active in the **acquire** state are:
  - b_1 avoid-static-obstacles(find-obstacle)
  - b_2 noise()
  - b_3 move-to-goal(detect-dirt)
- A_v **(Vacuum)**
  While in the **vacuum** state the robot vacuums the floor. If the *over-dirt* assemblage A_{od} reports that the area is not dirty, a transition is made back to the **forage** state. The behaviors of A_v are coordinated by $@_3$, which sums the motor outputs. Behaviors active in the **vacuum** state are:
  - b_1 avoid-static-obstacles(find-obstacle)

- b_2 noise()
- b_4 probe(detect-current-heading)
- b_5 vacuum-dirt()

- A_{dd} **(Detect-dirt)**
An assemblage used solely for its perceptual output. When in the **forage** state the overall coordination operator $@_4$ will transition to the **acquire** state if A_{dd} discovers a dirty area. There is only one behavior active in the **detect-dirt** assemblage:

  - b_6 null-motor(detect-dirt)

  Null-motor allows the embedded perceptual module detect-dirt to be invoked without any resultant robot motor action.

- A_{od} **(Over-dirt)**
Another assemblage used solely for its perceptual output. When in the **vacuum** state the overall coordination operator $@_4$ will transition to the **forage** state if A_{od} reports there is no dirt to vacuum up. There is only one behavior active in the **over-dirt** assemblage:

  - b_7 null-motor(over-dirt)

  Null-motor allows the embedded perceptual module over-dirt to be invoked without any resultant robot motor action.

Each of the assemblages is formally described below. The overall vacuum behavior is expressed in (7).

$$A_f = [@_1 b_1, b_2] \tag{2}$$

$$A_a = [@_2 b_1, b_2, b_3] \tag{3}$$

$$A_v = [@_3 b_1, b_2, b_4, b_5] \tag{4}$$

$$A_{dd} = [b_6] \tag{5}$$

$$A_{od} = [b_7] \tag{6}$$

$$A_{vt} = [@_4 A_f, A_a, A_v, A_{dd}, A_{od}] \tag{7}$$

V. Simulation Results

The system described above was implemented in simulation on a Sun Sparc under the X-windows environment. Figure 4 shows a sample run. The area to be vacuumed is a 64 unit by 64 unit square with 15% obstacle coverage. The obstacles are represented by black circles. The robot can vacuum a swath 1 unit wide. The robot's path is traced by a solid line or a dashed line depending on whether it is vacuuming, or looking for dirty areas, respectively.

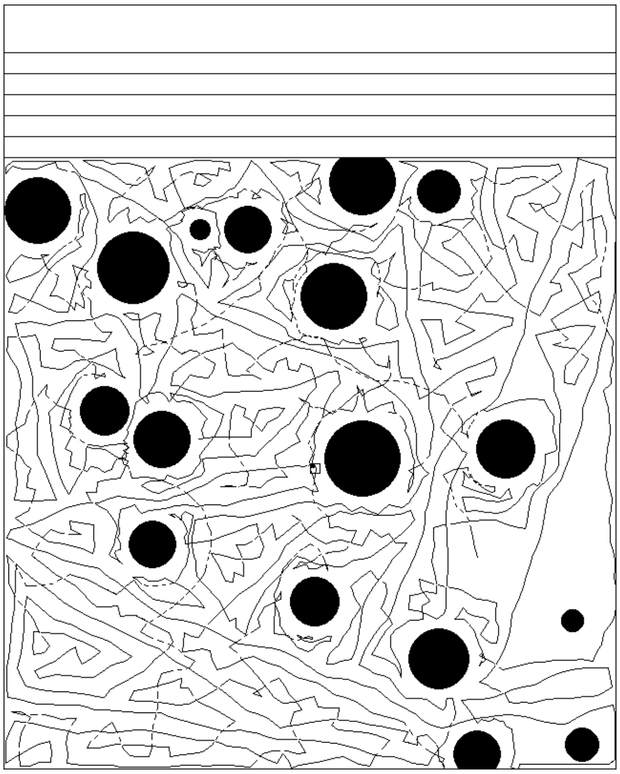

Fig. 4. Simulation of an agent vacuuming a room.

For this particular example, the agent required 7692 time units to complete the task. Of that time, 71% was spent actually vacuuming. The remaining 29% was spent either searching for or moving towards a "dirty" area.

VI. Multi-agent Vacuuming

In separate research our group has investigated reactive control for teams of robots[5, 6]. Since the vacuuming task is implemented within the existing multi-agent simulation system, it is easy to extend the vacuum task simulation to multiple agents.

A simulation of three agents vacuuming the same area as before is shown in Figures 5 and 6. Each of these agents is identical to the single agent described above. Even though there is no explicit communication between the agents, implicit communication occurs through the environment. The agents cooperate by avoiding areas other agents have already vacuumed. The three agents together were able to complete the task in 1554 time units, or 2.9 times as fast as one agent.

In other multi-agent tasks we have found that simple communication between agents significantly improves cooperation as measured by time to complete

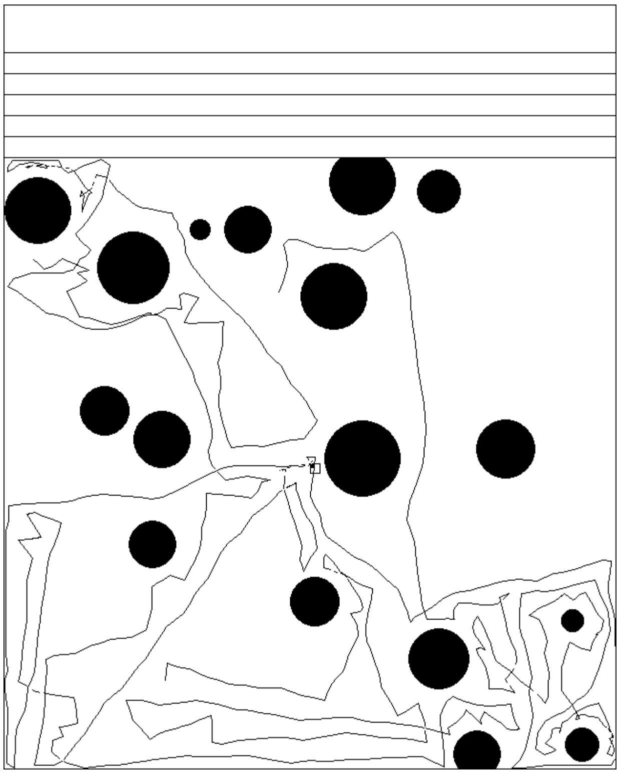

Fig. 5. Simulation of three agents vacuuming a room. This image shows the agents' initial progress.

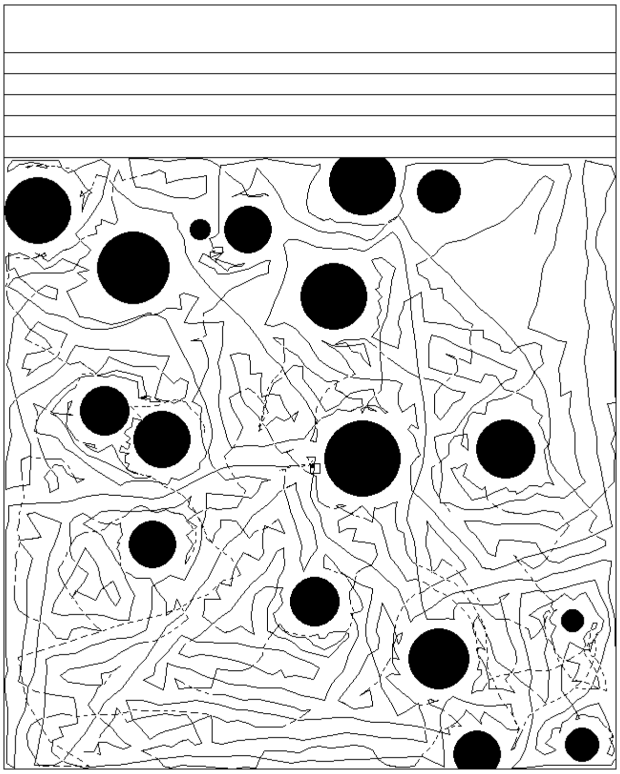

Fig. 6. Simulation of an three agents vacuuming a room. This image shows the completed simulation run.

the task[5]. The vacuuming task simulation has been extended to allow communication, but comparative performance results are not yet available.

VII. Summary

The AuRA architecture is flexible and extensible, allowing a wide range of tasks to be accomplished with minimal development effort. New problem domains are approached by combining existing primitive behaviors in new ways or by adding new primitive behaviors. These behaviors and assemblages of behaviors are described using the grammar introduced here.

The vacuuming task offered an opportunity to demonstrate AuRA's flexibility. The task was solved by selectively combining five primitive sensorimotor behaviors using the generic coordination operator. The solution was extended further with multiple cooperating agents.

References

[1] Arkin, R.C., *et. al.* , "Buzz: An Instantiation of a Schema-Based Reactive Robotic System", *Proc. International Conference on Intelligent Autonomous Systems: IAS-3*, Pittsburgh, PA., Pg 418-427, 1993.

[2] Arkin, R.C., "Motor Schema Based Mobile Robot Navigation", *International Journal of Robotics Research*, Vol 8(4), Pg 92-112, 1989.

[3] Arkin, R.C., Murphy, R.R., Pearson, M. and Vaughn, D., "Mobile Robot Docking Operations in a Manufacturing Environment: Progress in Visual Perceptual Strategies", *Proc. IEEE International Workshop on Intelligent Robots and Systems '89*, Tsukuba, Japan, 1989, Pg 147-154.

[4] Arkin, R.C., "Towards Cosmopolitan Robots: Intelligent Navigation of a Mobile Robot in Extended Man-made Environments", *Ph.D. Dissertation, COINS TR 87-80*, Univ. of Massachusetts, 1987.

[5] Arkin, R.C., Balch, T., Nitz, E., "Communication of Behavioral State in Multi-agent Retrieval Tasks", *Proc. 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, 1993, Vol. 1, Pg. 678.

[6] Arkin, R.C., Hobbs, J.D., "Dimensions of Communication and Social Organization in Multi-Agent Robotic Systems", *Proc. Simulation of Adaptive Behavior 92*, Honolulu, HI, Dec. 1992.

[7] Brooks, R., "A Robust Layered Control System for a Mobile Robot", *IEEE Jour. of Robotics and Auto.*, Vol RA-2, no. 1, Pg. 14-23, 1986.

[8] Kaelbling, L.P. and Rosenschein, S.J., "Action and Planning in Embedded Agents", *Robotics and Autonomous Systems*, Vol. 6, 1990, Pg. 35-48. Also in *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, P. Maes Editor, MIT Press, 1990.

[9] Kaelbling, L.P., "Goals as Parallel Program Specifications", *Proc. 1988 AAAI Conference* , St. Paul, MN, 1988., Vol. 1, Pg. 60-65.

[10] Kaelbling, L.P., "REX Programmer's Manual", SRI International Technical Note 381, May, 1986.

[11] Lyons, D.M., and Arbib, M.A., "A Formal Model of Computation for Sensory-Based Robotics", *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 3, June 1989.

[12] MacKenzie, D. and Arkin, R.C., "Perceptual Support for Ballistic Motion in Docking for a Mobile Robot", *Proc. SPIE Conference on Mobile Robots VI*, Nov. 1991, Boston, MA.

[13] MacKenzie, D. and Arkin, R.C., "Formal Specification for Behavior-Based Mobile Robots", to appear *Proc. SPIE Conference on Mobile Robots VIII*, Sept. 1993, Boston, MA.

[14] Maes, P., "Situated Agents Can Have Goals", *Robotics and Autonomous Systems*, Vol. 6, 1990, Pg. 49-70. Also in *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, P. Maes Editor, MIT Press, 1990.