A Pair of Heterogeneous Agents in a Unique Vehicle for Object Motion

Philippe Morignot
ICS-FORTH STEP-C, Vassilika Vouton, P.O.Box 1385
GR 711 10 Heraklion, Crete, Greece
Phone: +30 81391600 Fax: +30 81391601
E-mail: morignot@ics.forth.gr

Olivier Aycard, François Charpillet[0]
CRIN-CNRS / INRIA-Lorraine, Bat. LORIA, B.P. 239
54500 Vandoeuvre-les-Nancy, France
Phone: +33 3 83592084 Fax: +33 3 83413079
E-mail: {aycard, charp}@loria.fr

Keywords: Intelligent control, hybrid systems.

Abstract

We present a multi-agent architecture for controlling a mobile robot in an unpredictible environment. This architecture has been developed with the objective of coordinating the various competences of the robot (e.g., perception, navigation, planning). The architecture is made up of two agents: the first one specialized for cognitive tasks, the second one dedicated to control of the robot's physical devices. This two-agent architecture guarantees a good robustness of the system, as the navigation modules can run independently of the cognitive ones.

1 Introduction

Being able to design a robot that can both "sense/act" and "reason" at the same time in a coordinated way is a very important goal for both Robotics and AI. In a dynamic and unpredictable environment, a whole spectrum of reactions can be expected from a robot: from a minor bump on its trajectory to avoid an unexpected obstacle, to a major revision of the plan which is intended to carry out a given mission. Moreover, small communication bandwidth and/or low communication speed can limit heplful human intervention: decisions about reactions strictly have to be made onboard. These two requirements of reaction width and of onboard decision making constitute the first step towards the *autonomy* of a robot.

Embedding symbolic computation into robots has been studied by researchers for decades. Fikes, for example, uses a symbolic planner inside the Shakey robot in the early 70's [8]. But apart from the fact that task planning, for example, is an NP-complete problem [4], embedding a reasonably efficient symbolic reasoning capability in a robot leads to difficult architectural issues *per se*: time spent reasoning may be time lost for sensing/reacting. For example, the robot might plan for a task which is already obsolete due to fast changes in the environment — hence making that whole planning session useless, in retrospect.

In another view, Brooks argues that keeping an explicit model of the environment inside a robot may be useless because the robot is *situated* in its environment [3]. (Similarly, living organisms are situated in their natural environment through evolution.) The architecture of robots' software should be thought of as a combinatorial circuit and a time circuitry — both being hand-programmed at design time. However, coupling onboard reasoning with physical moving still seems to be required due to the discrete aspect of robot (re)actions, to the various abstraction levels required for them and to their multiple meanings [5] For example, the robot may act in several ways: navigating to reach some location; navigating to perform bee-like figures to warn other robots; moving objects by pushing them; moving objects by manipulating them with a clamp, with objects having complex uses and functions.

In another view towards adressing architecural issues, it has been proposed to keep the environment's symbolic model inside the robot but to have it run in parallel with sensing/reacting parts [9] [11]. If both activities run on the same processor (even using real-time operating systems), the sensing/reacting capability of the robot is still weakened, but interesting realizations in natural/non-engineered environments have been built [9].

To reinforce this latter view, a paradigm of AI coined *multi-agent* proposes to have problems solved by a a set of autonomous entities (*agents*). The *behavior* of this *society* of *agents* (terms used by analogy with human and animal societies) emerges from the individual activities of these agents and from their mutual interaction [7]. In this paper, we propose a multi-agent architecture for controlling a mobile robot which can reason (e.g., plan) and sense/act at the same time. We first present the design of the multi-agent system. We then present the specific agents for object motion. An implementation on a mobile robot is presented and discussed.

[0]Thanks to Vincent Chevrier for useful comments on earlier drafts of this paper and to the members of the RFIA group for useful discussions. The first author has been supported at CRIN-CNRS/INRIA-Lorraine by an Esprit III contract and at ICS-FORTH by an ERCIM fellowship. The second author has been supported by MSER fellowship from French government.

2 System Design

Multi-Agent. The mobile robot materializes the agents' society: the behavior of the robot *is* the emerging behavior of this society. Various number of agents of a multi-agent system (or, closely, various number of levels of a unique agent) has been proposed: One [16], two [8], three [9] or n [3]. The system presented in this paper stands between the extreme many-simple-agent approach and the extreme single-complex-agent one: a small number of reasonably complex agents are modeled. We propose (i) an agent architecture, (ii) common component types for agents, (iii) a role for each agent in the society, (iv) a communication protocol for collaboration.

Architecture of One Agent. Each agent is composed of independent modules (*behaviors*), a mechanism for choosing which module to execute next (*controller*) and shared memory (*blackboard*) [10]. A behavior is made up of two parts: *triggering conditions* determine the executability of the behavior, the second part contains the behavior's source code itself.

The controller is a best-first execution cycle on the set of instantaneously-executable behaviors. At each cycle, it checks which behavior is executable (*trigger condition checking*), sorts the executable behaviors by decreasing importance (*agenda management*), and gives control to the most important one (*execution*). It is the behavior's responsibility to give control back to the controller, so as to make the next cycle possible.

At each cycle the executable behaviors that were less important than the executed one are forgotten (hence the term best-first *execution cycle*, for a best-first *search* would keep them). Since all behaviors are considered at each cycle for executability check, a behavior which is declared executable at a given cycle can be declared executable at the next cycle, if its triggering conditions are not affected by the executed behavior.

The sort criterion for evaluating the importance of executable behaviors can be specified in a dedicated part of the shared memory (*control plan*).

An *event* is a record of a change in the shared memory. A memory change can be initiated by the execution of a behavior or by reception of a message from another agent (which, in turn, comes from the execution of its own behaviors). Events, behaviors and other internal objects are stored in another dedicated part of the shared memory.

The controller matches behaviors' triggering conditions against events: This prevents the controller from uselessly polling an unchanged shared memory, hence it reduces the controller's CPU consumption for non-behavioral use.

3 Specialization for Object Motion

For the application of object motion, the presented system is composed of two agents. These agents have heterogeneous semantics and isomorphic structures — hence they play complementary roles in this society. One agent is in contact with the environment through sensors and effectors. It carries out a behavior for action (in a sense similar to *walking*, *running*) or perception (e.g., noticing an open area on the right). Another agent manipulates the interpretation of previous physical behaviors. We informally refer to the former agent as a *physical* agent and to the latter agent as a *cognitive* agent.

The cognitive agent sends action description to the physical agent. The physical agent sends events to the cognitive agent. An event can be perceptual (e.g., open door on the right, stuck) or related to the agent itself (e.g., task performed, low battery).

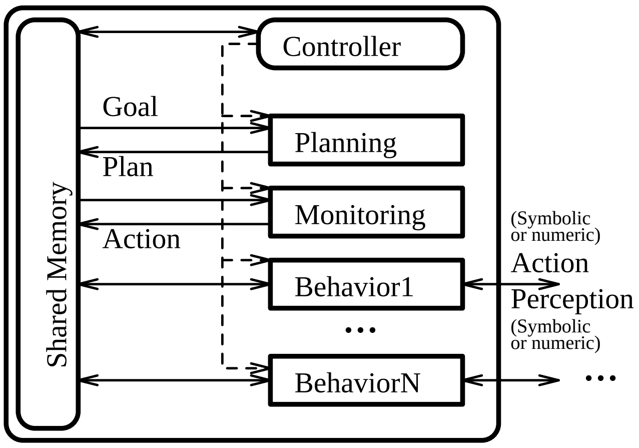


Figure 1: The software architecture of each agent.

To attempt at achieving given task, a mobile robot must anticipate: its intended actions influence the current action. This temporal aspect leads to give at least two behaviors to an agent, as shown in Figure 1: monitoring (i.e., computation about the agent's present) and planning (i.e., computation about the agent's future).

3.1 Cognitive Agent

Task Planning. The triggering condition of this behavior is the presence in the shared memory of a (conjunctive) goal which has no plan satisfying it.

This behavior consists of finding a sequence of action descriptions (a *plan*) that transforms an initial situation description to another in which a given conjunction of *goals* is satisfied. The initial situation description typically is the current situation of the robot, the goals altogether represent a *task* to achieve. This behavior uses a set of descriptions of the actions that the robot can physically perform (see Figure 2). An action is represented as pre-, post-conditions [8] and preservation conditions (i.e., preconditions which must hold *during* the action's execution) [15].

```
(deftemplate (drop ?robot ?object ?location)
   :preservations ((robot ?robot)
                   (height ?object ?height)
                   (at ?robot ?location)
                   (clamp-height ?height))
   :preconditions ((at ?object ?robot)
                   (:not (is-clamp-open)))
   :postconditions ((at ?object ?location)
                    (:not (at ?object ?robot))
                    (:not (is-clamp-open)))
   :location ?location)
```

Figure 2: Description of a `drop` action.

The initial situation description and the goals form an initial plan, which is incrementally improved during the process. The task planner handles partial-ordered partially-instantiated plans: two actions in parallel mean that they are unordered, an uninstantiated variable means that it can be instantiated to more than one constant (the domain of a variable is pruned during the process). Arithmetic equations can be maintained by representing integers as function symbols in actions' pre- and postconditions [13]; this enables to represent limitations on the capacity of the robot.

Task planning is a search in the space of partially-ordered partially-instantiated plans [4]. A best-first search, for example, ensures completeness. A plan is a solution to a given planning problem (i.e., initial plan plus action descriptions) iff it contains no unsatisfied precondition or preservation condition. The satisfaction of a precondition (or, similarly, of a preservation condition) is computed in polynomial time with a modal truth criterion (MTC) [4]. This ensures correctness of the generated plans.

Monitoring. The triggering condition of this behavior is the presence in the shared memory of a plan which contains unexecuted action descriptions.

This behavior attempts at having a plan executed by sending (a simplification of) its *current* action description to the physical agent — this decoupling makes the cognitive agent available for reasoning on other tasks. Such action description is declared executed when a message of successful completion is received from the physical agent. A failure message leads to replanning from the situation just before that action description — this situation is computed by simulating the execution of action descriptions from the initial situation, i.e., adding positive postconditions to it and retracting negative ones from it [8].

An action description can be sent to the physical agent before reception of any message related to the previous action description, if the preservation conditions of one of them do not conflict with the post conditions of the other one, according to the MTC algorithm.

```
1. If event is performed-action and event source is
   physical agent,
       a. If the current plan has no unexecuted actions,
          Then If there exists at least another plan,
               Then select another plan as current.
               Else exit from this behavior.
       b. Unstack the current action from the current
   plan.
       c. Send it to the physical agent.
2. If event type is goal and event source is physical agent,
       store it in the shared memory.
3. If event type is percept and event source is physical
   agent,
       a. If the event negates a preservation condition
   of the currently executed action,
          Then stop this action and turn this condition
   into a goal.
       b. Store it in the shared memory.
```

Figure 3: Rules for monitoring.

More precisely, this behavior reacts to events (sent by the physical agent), by choosing a plan (step 1.a of Figure 3), taking the current action of that plan (step 1.b) and sending it to the physical agent (step 1.c).

The monitor also filters the perceptual events from the physical agent (step 2). For example, a goal event which is perceived in the environment (e.g., requests from humans) is recognized, which may lead the cognitive agent to plan or replan (see previous paragraph). The monitor also recognizes an self-related event which conflicts with a preservation condition of currently-executed action descriptions (step 3); If so, it turns the threatened preservation condition into a goal (which, again, may lead the cognitive agent to plan or replan).

3.2 Physical Agent

Path Planning. This triggering condition of this behavior is the presence of a motion action description in the shared memory of the physical agent.

This behavior uses a *topographic* map and a *topological* map of the environment. Both maps are stored in the shared memory of the physical agent. The topographic map records fixed obstacles: each point is a binary value can-go/cannot-go — the environment is assumed to be plane. A node of the topological map corresponds to a *salient* location of the topographic map (see Figure 5 for an example). A location is salient iff it can be detected in 2D with precision by the robot's sensors. When the path between two salient locations is unique, an edge is added between these two nodes. The topological map is manually extracted from the topographic map in a preprocessing phase.

Path-planning involves: (1) finding the salient point sp_1 which is the closest to the current location; (2) finding the salient point sp_2 which is the closest one to the goal location; (3) searching for sp_2, starting at sp_1, in the topological map. For example, an A^* search ensures completeness and produces a path from sp_1 to sp_2 which is minimal according to a cost function (e.g., path length).

Navigating. The triggering condition of this behavior is the presence of a primitive motion action description in the shared memory of the physical agent.

The robot has three wheels for translation. Turns can be made by rotating the wheels around a vertical axis (non-holonomic motion). Environment is sensed through a 20-bumper ring (emergency stop) and a 16 sonar/infra-red ring (regular navigation).

Different sensor zones are defined around the robot (e.g., left, front, right); each zone gives a preferred rotation angle and speed for the next cycle; a fuzzy controller merges the different advice (i.e., obstacle avoidance on the left/front/right) to compute the actual values [1]. Navigation stops either (1) when the next salient location is reached (The method for building the topological map from the topographic map guarantees its existence) or (2) when the goal location is estimated to be reached using odometry (Again, the method for building the topological map from the topographic map guarantees the minimal use of this navigation mode.) Salient location detection uses the variation of a sonar/infra-red sensor value between two time points [12] (e.g., open door or corridor on the right).

Grabbing/Dropping. The triggering condition of this behavior is the presence of a grab/drop action description in the shared memory of the physical agent.

A clamp, at the back of the robot, is composed of two parallel bars that move in a horizontal plane (see Figure 4). This clamp can move vertically, can open/close, can detect resistance when closing (hence meaning that an object is being grabbed if $d \neq 0$) and can rotate. (The upper part of the robot is a rotatable turret on which the upper part of the clamp's rail is fixed.)

The plan for grabbing is: (1) move the clamp vertically to reach the target-object's height (the *object's height* refers to the height at which the object should be grabbed, see Figure 2, the actual height of the object usually is higher); (2) open the clamp; (3) close the clamp until resistance is detected; if the clamp is closed and no resistance has been detected on the way, re-open the clamp until $d = d_{max}$, send a warning message to the physical monitor and go to step (3); (4) move the clamp up. *A priori* actions (1) and (2) are unordered.

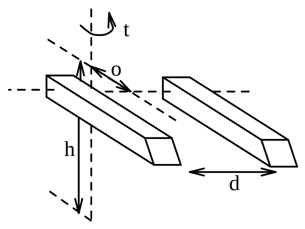


Figure 4: A two-finger clamp rotates (angle t) around the turret's axis (at distance o from the clamp), moves vertically (distance h from the floor, $0 \leq h \leq h_{max}$) and opens/closes (distance d, $0 \leq d \leq d_{max}$).

The plan for dropping is: (1) move the clamp down to reach the target-object's height; (2) open the clamp until $d = d_{max}$; (3) move the clamp down. For safety purposes, the robot with an empty clamp navigates with the clamp in low position $h = 0$. Objects are assumed to have their lower part less wide than d_{max}.

Monitoring. The triggering condition of this behavior is the presence of a plan in the agent's shared memory which contains unexecuted action descriptions. The only difference with the cognitive monitor is that the only filtered percept is the completion or failure of an action — such events are propagated to the cognitive agent.

A plan here either is a structure simpler than these for the cognitive monitor (i.e., sequence of salient locations) or is hand-coded in an action.

4 Experiments

The two-agent architecture has been implemented on a Nomad 200 mobile robot [6]. It is composed of a base and a turret, which can rotate independently. The base is composed of three wheels and is protected by a 20-bumper ring. The turret's horizontal cross-section is an uniform 16-side polygon, with an infrared and ultrasonic sensor on each vertical plane (i.e., on each side of the polygon). The turret also carries a stereo CCD-camera and a pair CCD-camera / laser. An onboard pentium PC under the Linux operating system communicates with the sensors/effectors. It also communicates with the local computer network of the laboratory via a radio link.

We tested the two-agent architecture in scenarios where the mobile robot gathers soda cans in an indoor environment (the second floor of the CRIN-CNRS laboratory, see Figure 5). Objects *not* mentionned in the topographic map include: shelves, door frames, fire extinguishers, cardboard boxes and people. This environment has not been engineered for mobile robots in particular.

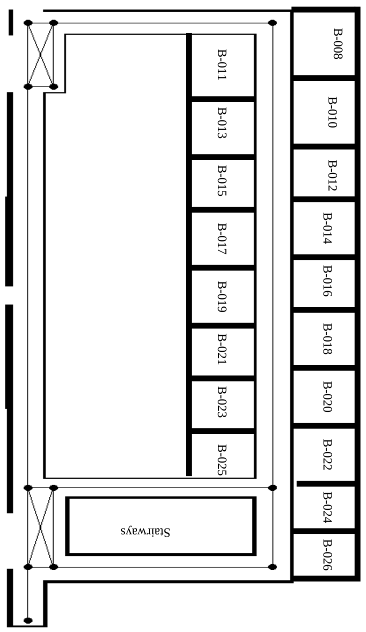


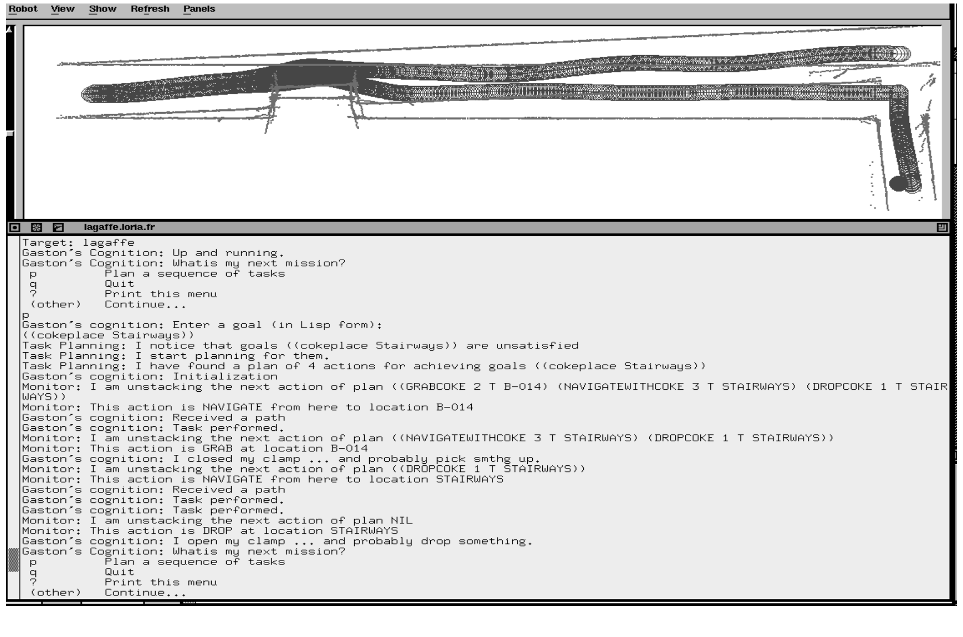
Figure 5: A topographic map. (Nodes and edges of the topological maps are shown as dots and thin lines.)

A goal (a mission) is sent on the fly to the cognitive agent: typically, bringing a soda can to some location — the initial locations of the robot and of the can are included in the model of the environment (topographic, topological and symbolic) which is maintained by the robot. Figure 6 and 7 show typical traces of the robot's motion and of the cognitive / physical agents' activity.

5 Discussion

1. Again, our point is not to demonstrate the performance of a particular behavior but to demonstrate the usefulness of a multi-agent architecture for controlling a unique vehicle. However, in cases actually happening in our scenarios in our environment, task planning is faster than action execution (less than one second for task planning [13], several seconds for action execution [1]). But due to the polynomial or exponential complexity of each behavior, opposite observations could be made in close scenarios (e.g., safely moving seven stacked boxes with one intermediate location only (problem known as the *Towers of Hanoi*)). For the general case, domain-dependent behaviors can be added to each one of the two agents and can mask default-case behaviors (e.g., using priorities in line 1.*a*.3 in Figure **3** or in similar lines of the physical monitor).

2. The plans executed by the physical monitor are mostly hand-programmed (except that navigation can require path planning). Extending the best-first execution cycle that currently implements a monitor by using a language such as PRS-Lite [14], for representing such plans or *procedures*, would help expressing the knowledge on use conditions of sensors/effectors and associated behaviors. Thus this would (1) help making these behaviors reusable for instances of our architecture on other robots, (2) improve the *reaction capabilities* while the cognitive agent deliberates (e.g., replans), hence nicely enabling a form of graceful degradation [3].

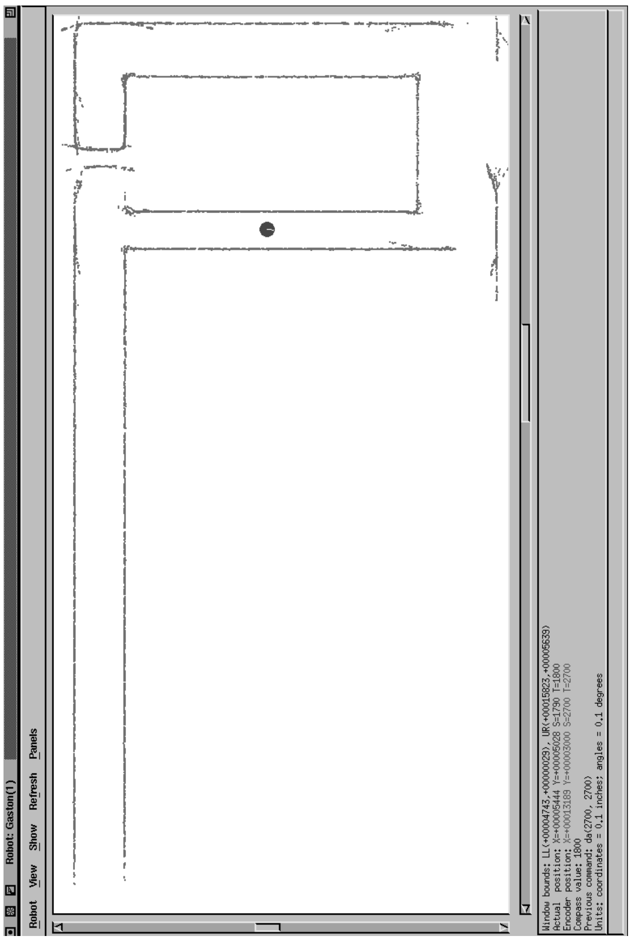


```
Target: lagaffe
Gaston's Cognition: Up and running.
Gaston's Cognition: Whatis my next mission?
 p          Plan a sequence of tasks
 q          Quit
 ?          Print this menu
 (other)    Continue...
p
Gaston's cognition: Enter a goal (in Lisp form):
((cokeplace Stairways))
Task Planning: I notice that goals ((cokeplace Stairways)) are unsatisfied
Task Planning: I start planning for them.
Task Planning: I have found a plan of 4 actions for achieving goals ((cokeplace Stairways))
Gaston's cognition: Initialization
Monitor: I am unstacking the next action of plan ((GRABCOKE 2 T B-014) (NAVIGATEWITHCOKE 3 T STAIRWAYS) (DROPCOKE 1 T STAIR
WAYS))
Monitor: This action is NAVIGATE from here to location B-014
Gaston's cognition: Received a path
Gaston's cognition: Task performed.
Monitor: I am unstacking the next action of plan ((NAVIGATEWITHCOKE 3 T STAIRWAYS) (DROPCOKE 1 T STAIRWAYS))
Monitor: This action is GRAB at location B-014
Gaston's cognition: I closed my clamp ... and probably pick smthg up.
Monitor: I am unstacking the next action of plan ((DROPCOKE 1 T STAIRWAYS))
Monitor: This action is NAVIGATE from here to location STAIRWAYS
Gaston's cognition: Received a path
Gaston's cognition: Task performed.
Gaston's cognition: Task performed.
Monitor: I am unstacking the next action of plan NIL
Monitor: This action is DROP at location STAIRWAYS
Gaston's cognition: I open my clamp ... and probably drop something.
Gaston's Cognition: Whatis my next mission?
 p          Plan a sequence of tasks
 q          Quit
 ?          Print this menu
 (other)    Continue...
```

Figure 6: Trace for soda can gathering.

3. We were not able to find a behavior which was neither cognitive nor physical, so an agent actually refers to a behavior *type*, either cognitive or physical. Now, if an agent refers to a *function* [7], more than two such agents can be represented in our architecture: each controller acts as an operating system assigning time slices to (aggregations of parts of different) behaviors/agents, their conflicting roles being handled by use of a MTC on their description, or more generally by mutual *negociation*. For example, an "obstacle avoidance" functional agent may involve navigating, path-planning and task planning behaviors, with potential negociation between path-planning and task-planning (e.g., opening a door (which may require fetching keys) vs. taking a longer path to go around it (through another open door)).

Ongoing work involves using Hidden Markov Models for navigation to cope with reflection of sonar/infrared sensors on obstacles [2], and using the stereo CCD-camera for active 3D sensing.



```
Window bounds: LL(+00004743,+00000029), UR(+00015823,+00005639)
Actual  position: X=+00005444 Y=+00005028 S=1790 T=1800
Encoder position: X=+00013189 Y=+00003000 S=2700 T=2700
Compass value: 1800
Previous command: da(2700, 2700)
Units: coordinates = 0.1 inches; angles = 0.1 degrees
```

Figure 7: Trace for soda can gathering. (An unexpected obstacle in front of office B-024 leads the robot to go around the stairways.)

References

[1] Olivier Aycard. A Two-Level Fuzzy Controller for Mobile Robot Navigation. Technical report, CRIN-CNRS/INRIA-Lorraine, Nancy, France, 1996.

[2] Olivier Aycard, Jean-François Mari, and François Charpillet. Place Learning and Recognition using Hidden Markov Models. Technical report, CRIN-CNRS/INRIA-Lorraine, Nancy, France, 1996.

[3] Rodney Brooks. Intelligence without Reason. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 979–984, Detroit, MI, 1991.

[4] David Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333–377, 1987.

[5] Raja Chatila. Deliberation and reactivity in autonomous robots. *Robotics and Autonomous Systems*, 16:197–211, 1995.

[6] David Zhu et al. Nomad 200 user s manual. Technical report, Nomadics Technologies Inc., Mountain View, CA, 1996.

[7] Jacques Ferber. *Les Systèmes Multi-Agents: Vers une Intelligence Collective.* InterÉditions, Paris, 1995.

[8] Richard E. Fikes and Nils J. Nilsson. STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:198–208, 1971.

[9] Erann Gat. Integrating Planning and Reacting in a Heterogeneous Asymchronous Architecture for Controlling a Real-world Mobile Robot. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Mateo, CA, 1992. Morgan Kaufmann.

[10] Barbara Hayes-Roth. A Blackboard Architecture for Control. *Artificial Intelligence*, 26:251–321, 1985.

[11] Barbara Hayes-Roth, Karl Pfleger, Philippe Lalanda, Philippe Morignot, and Marko Balabanovic. A Domain-Specific Software Architecture for Adaptive Intelligent Agents. *IEEE Transactions on Software Engineering*, 21(4):288–301, April 1995.

[12] D. Kortenkamp, L. Douglas Baker, and T. Weymouth. Using gateways to build a route map. In *proceedings of the 1992 IEEE International Conference on Intelligent Robots and Systems*, 1992.

[13] Philippe Morignot. Embedded Planning. In Kristian Hammond, editor, *Proceedings of the Second International Conference on A.I. Planning Systems*, pages 128–133, Chicago, IL, 1994.

[14] Karen L. Myers. A Procedural Knowledge Approach to Task-Level Control. In *Proceedings of the Third International Conference on A.I. Planning Systems*, Edimburgh, UK, May 1996.

[15] Edwin Pednault. *The Synthesis of Plans.* PhD thesis, E.E. Dept., Stanford, Palo Alto, California, 1986.

[16] Marcel J. Schoppers. Universal Plans for Reactive Robots in Unpredictible Environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1039–1046, 1987.