DERVISH
An Office-Navigating Robot

Illah Nourbakhsh, Rob Powers, and Stan Birchfield

■ DERVISH won the Office Delivery event of the 1994 Robot Competition and Exhibition, held as part of the Thirteenth National Conferennce on Artificial Intelligence. Although the contest required DERVISH to navigate in an artificial office environment, the official goal of the contest was to push the technology of robot navigation in real office buildings with minimal domain information. DERVISH navigates reliably using retractable assumptions that simplify the planning problem. In this article, we present a short description of DERVISH's hardware and low-level motion modules. We then discuss this assumptive system in more detail.

The judges are ready, and DERVISH knows its mission: Exit the room in which it has been placed, navigate down the artificial hallways to one of the goal room's two doors, and enter the goal room. DERVISH begins by exiting the room. Turning to align itself with the hallway, it begins to move toward the near door of the goal room, which is just a few feet in front. This run should be easy, so the robot thinks. Oops! DERVISH sees a hallway on its left and knows that the hallway is opposite the door. DERVISH must have missed the door. DERVISH backs up to where the door must be and searches back and forth a few feet, looking for an opening. Finding none, DERVISH decides that this door must be closed; replans to the far door of the goal room; and begins to move down the hallway again, heading for the far door. Uh-oh! A large object is in front, blocking the robot's path. DERVISH searches to find a way around the object but to no avail; it must be a blockade, so DERVISH plans to use another hallway. After successfully traveling to the other side of the office building, DERVISH reaches the final hallway, slowly but surely approaching the far door. DERVISH reaches the door and keeps going. The audience gasps in despair, thinking for an instant that DERVISH has missed the door, but DERVISH stops, backs up to the door, and enters the room. The crowd cheers.

So it was that DERVISH won the Office Delivery event of the 1994 Robot Competition and Exhibition, held as part of the Thirteenth National Conference on Artificial Intelligence (AAAI-94). DERVISH was the only robot to successfully complete this fourth and final trial.

Although the contest required DERVISH to navigate in an artificial office environment, the official goal of the contest was much broader: to push the technology of robot navigation in real office buildings with minimal domain information, such as a *topological map* (a connectivity map that contains no distance information) and approximate hallway and doorway widths. We designed DERVISH with this more ambitious goal in mind, and as a result, DERVISH navigates well in real office environments, including three separate office buildings at Stanford University.

To succeed, DERVISH must never become unrecoverably lost despite its sensory and effectory error. To this end, DERVISH maintains its sense of position in the form of a *state set* that contains all the states that the robot believes are possible. This state set is *progressed*, or updated, whenever the robot discerns a new percept. DERVISH uses an *assumptive system* that plans under the assumption that the robot is in the most likely state in the state set. It then executes the resultant path while it continually updates the state set until either the goal is reached, or the current most likely state is no longer on the path, at which point it replans.

In this article, we present a short descrip-

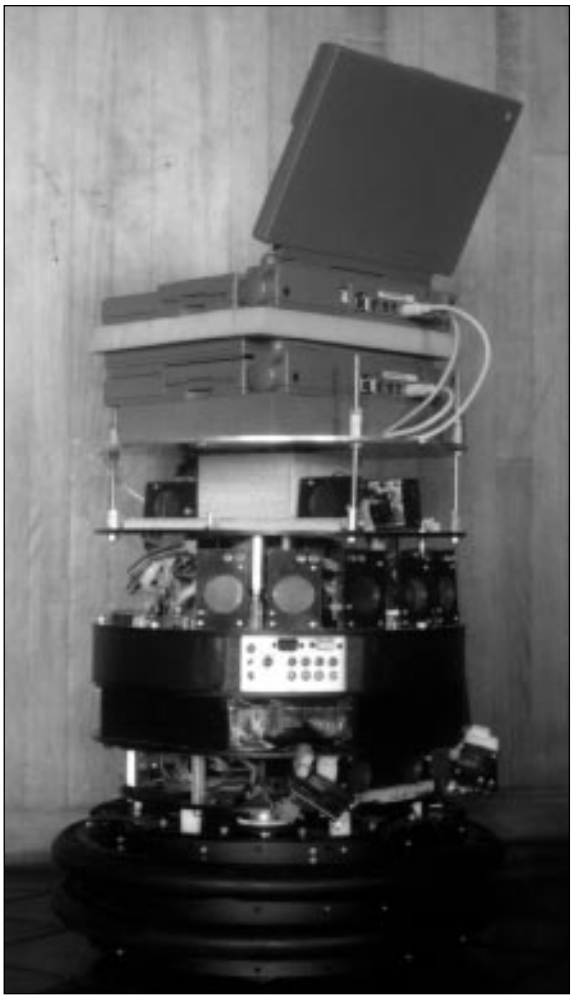

Figure 1. DERVISH.

tion of DERVISH's hardware and low-level motion modules. We then discuss state-set progression and the assumptive-planning mechanism in more detail.

Hardware

Our original robot, a NOMAD 100, was equipped with the traditional, planar, 16-sonar ring. The problem with planar sensor rings is that robots and office buildings are not planar. Therefore, any robot with such a sensor configuration is subject to both tripping over short objects below the ring and being decapitated by tall objects (for example, ledges, shelves, and tables) above the ring. We abandoned radial symmetry to create the nontraditional sonar configuration shown in figure 1.

To detect obstacles, DERVISH uses four separate sonar clusters. The largest cluster consists of five sonars pointing straight ahead: three at the height of the original ring and two near the base of the robot to detect small obstacles such as paper cups. Additional clusters point left and right at 45-degree angles. A final cluster, containing two sonars near the base of the robot pointing upward, is able to detect obstacles at any level, thereby preventing decapitation. A weakness of sonar is that at small angles, a sonar's sound wave reflects off the surface without producing a return echo; to minimize this effect, we vary the angles of the sonars within each cluster.

In addition to the sonar clusters, DERVISH has a pair of sonars on each side, perpendicular to its direction of travel, and rotational and translational encoders. These sensors are essential to DERVISH's ability to localize in hallways because they facilitate the detection of parallel walls and the computation of the robot's angle with respect to the walls (an extremely useful parameter in an office environment). Unlike most mobile robots but like most humans, DERVISH has no sensors facing backward.

DERVISH's brain is an on-board MACINTOSH POWERBOOK running Lisp that communicates by serial link with the robot. A second on-board POWERBOOK is dedicated to singing and speech. We have found that using purely on-board computation has two key advantages: (1) commands to the robot are never lost because of electromagnetic interference from outside sources and (2) the robot's range of motion is not restricted by the range of a radio modem.

Low-Level Control

Equipped with its new sonar configuration, DERVISH is capable of wandering aimlessly and safely throughout an office building, but aimless wandering does not an office robot make. According to the contest rules, the robot begins in a known room at an unspecified position and orientation within the room, and its goal is to navigate to another room whose identity is known in terms of the topological map. We decompose this problem into three distinct tasks: (1) leaving the initial room, (2) navigating through the hallways, and (3) entering the goal room.

The module for leaving the room is a wall-following routine whose actions are predominantly determined by current sensor readings. This module terminates when DERVISH either detects that it has passed through a doorway or later determines that it is moving in a long, straight hallway. When the module terminates, the robot can determine its position

and orientation by analyzing the topological map because there is only one door to the initial room, according to the contest rules.

Later, when the robot finally reaches the node just outside the goal room, the *enter-room module* is called. This simple procedure aligns the robot with the doorway and then moves a prespecified distance into the room.

The most complex module is the *hallway navigator*, which moves the robot from its current position to a specified hallway node using a path given in terms of intersecting hallways. As DERVISH moves down a hallway, this module incorporates raw sensor data to avoid obstacles in the robot's path and to test for the presence of closed doors, open doors, and open hallways on either side. For example, DERVISH detects closed doors as consistent four-inch depressions in the wall. If DERVISH detects any of these three features, it combines it with the feature on the other side (nothing if no feature was detected) to generate a percept pair. For a more detailed description of the hallway navigator, see the sidebar.

The hallway navigator displays the ability to execute a single abstract action to move down the hallway while it detects multiple percept pairs. Each percept pair allows DERVISH to update its state set and then choose the most likely new state as its newly assumed current state. As long as this state remains on the path being taken by the current plan, execution of the path continues uninterrupted. Control is relinquished only if the robot reaches the goal or if the newly assumed state is incompatible with the current plan.

In contrast to many roboticists, we believe that current simulators are not sufficiently accurate models of the real world to be of value for the development of low-level motion code; therefore, we developed and tested these three modules exclusively in real office buildings on the Stanford campus. Every office building has characteristics that are peculiar to the particular environment; by testing in three separate office buildings, we challenged the robustness of our solution.

Figure 2 depicts the interaction of these three modules with both the state set and the *dictator*, which is the higher-level component responsible for planning and invoking the motion modules. The arc from the hallway navigator module to the dictator enumerates the possible return conditions of the hallway navigator, which the dictator uses to decide whether to replan or transfer control to the enter-room module.

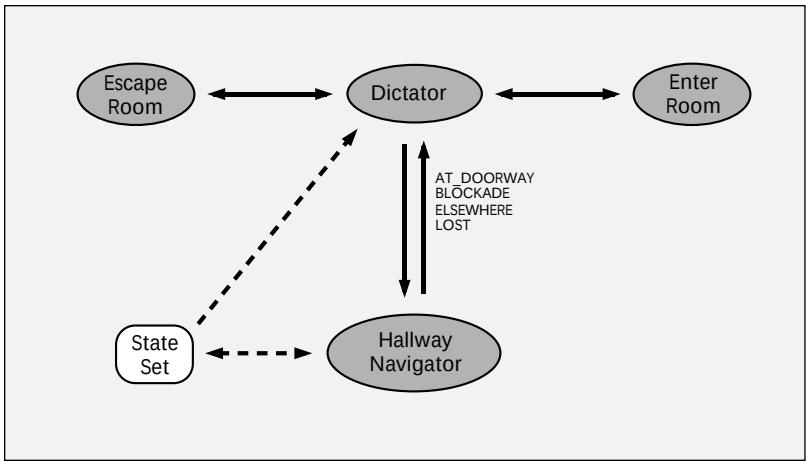

Figure 2. Control Diagram.

High-Level Control

Although the hallway navigator afforded robust motion in several office buildings, DERVISH made frequent mistakes recognizing features in the real world. To navigate robustly, DERVISH needed a higher level of control that would take potential sensory errors into account. We implemented such a system by designing an abstract representation that allowed DERVISH to associate certainty factors with possible world states and interleave planning and execution.

Abstraction

The real world is so finely grained that any attempt to plan using a highly detailed model of reality is doomed because of enormous computational complexity. DERVISH avoids this pitfall by reasoning about an abstraction of the real world that is based on a quantization of the world into a set of states; each state corresponds to a node on the topological map or a hallway segment between two nodes. Figure 3 depicts the quantization of a hallway fragment. The area encompassing a topological node is labeled with a number, whereas the area between two topological nodes is labeled with a dashed number. The state set represents DERVISH's positional uncertainty because it captures all the actual positions at which the robot believes it could be located. (We assume no uncertainty about DERVISH's orientation once it has left the initial room.) Associated with each state is a *certainty factor,* which represents the relative likelihood of the robot being in the corresponding state.

Note that the robot's notion of state consists only of information about its own location. For example, the state does not contain information about whether particular doors

The Hallway Navigator

The following pseudocode and its explanation describe in detail the execution of one leg of a path, that is, the navigation down one hallway to an intersection or doorway. The final paragraph explains the logic that takes place in between two path legs. Taken together, the explanations describe the hallway navigator.

```
while (!Termination) {
        Hallway_Localize()
        Set_Velocity()
        if New_Percept() Update_State_Set()
        Termination =
        Discern_Termination_Condition()
}
```

Hallway_Localize(): While it is moving down the hallway, the robot establishes its sense of position in the hallway by maintaining three parameters: (1) distance to the left wall, (2) distance to the right wall, and (3) angle with respect to the hallway. DERVISH's unusual sonar configuration greatly facilitates this task because two sonars point directly at each wall and are parallel to each other. The distances to the left and right walls are updated using the robot's encoders if the current distances detected by sonar differ from the expected values by more than two inches.

Set_Velocity(): At each time step, the hallway navigator must set the robot's translational and rotational velocities. The translational velocity is determined simply by the amount of open space in front of the robot. The rotational velocity involves combining several possibly competing objectives: (1) avoid obstacles, (2) remain aligned with the hallway, and (3) stay a reasonable distance from the walls. In addition, a governor limits the angle of the robot with respect to the hallway to be less than a certain threshold (between 17 and 30 degrees depending on the environment). The governor's primary purpose is to minimize the side sonars' specular reflection with hallway walls. If the governor prevents the robot from navigating around a large obstacle, the robot stops and makes an explicit search to determine whether the obstacle completely blocks its path.

if New_Percept() Update_States(): As DERVISH moves down the hallway, it continually compares its side sonar readings with the expected readings based on its projected hallway position. The expected readings are generally accurate within one or two inches, allowing DERVISH to look for closed doors as consistent four-inch depressions in a wall. Open doors and hallways are defined as a string of sonar readings that are consistently longer than expected. DERVISH distinguishes between the two by measuring the width of the opening using its translational encoders. Percepts from the left and the right sides are combined into a percept pair, which the robot uses to progress the state set.

Discern_Termination_Condition(): The loop terminates (that is, the termination flag is set) when any one of four termination conditions is discerned:

First, in the *At_Turn_Node* termination condition, the robot has successfully reached the end of a path leg.

Second, in the *elsewhere* termination condition, the robot is no longer on the path. (This condition can occur either when the robot overshoots a turn or, more rarely, when it relocalizes to another hallway.)

Third, in the *blockade* termination condition, an obstacle completely blocks the robot's path.

Fourth, in the *lost* termination condition, the state set has become empty (that is, there are no possible states). This condition is extremely rare.

If At_Turn_Node is discerned, the hallway navigator aligns the robot with the new direction of travel for the next path leg, and the loop begins again; if there are no more path legs to be traversed, the hallway navigator returns At_Doorway to the dictator. If one of the remaining conditions is encountered, it is immediately returned to the dictator.

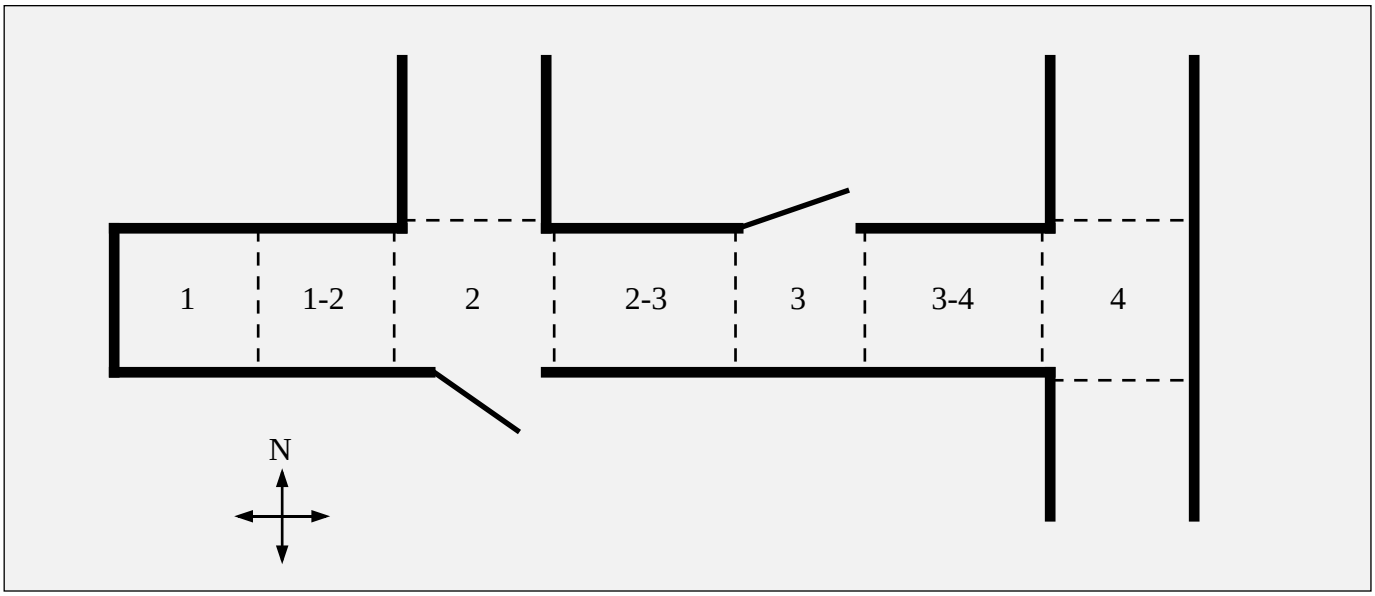

Figure 3. The Quantization of a Hall.

are open or closed. Including more information would help reduce positional uncertainty over time but would also greatly increase the size of the state set. Experimentally, we have found that DERVISH performs well without this additional information.

State-Set Progression

As the robot moves down a hall, whenever it discerns a percept pair, it updates the state set by progressing each state based on the percept pair and the current direction of travel. Progression of a state involves removing it from the state set and replacing it with all possible subsequent states.

For example, consider a robot with perfect sensors facing east, with the initial state set {1-2, 2-3} (figure 3). If this robot were to detect a hallway on its left and a door on its right, the percept pair would uniquely determine its position; that is, its state set would be the singleton set {2}. Possible state 1-2 would progress to 2, but 2-3 would progress to the *empty set* (that is, the percept pair is impossible if the world state was 2-3).

Of course, DERVISH does not have perfect sensors; so, we augment state-set progression with certainty factors that are computed using a certainty matrix and the probability of doors being closed. For each of the five world features that the robot can encounter, the certainty matrix assigns a likelihood that each of the three possible percepts (or *nothing*) will be detected. For example, using the certainty matrix that follows, if DERVISH is beside an open hallway, the likelihood of mistakenly recognizing it as an open door is 0.10. For each new environment, we empirically generate a certainty matrix and determine the closed-door probability. Interestingly, the success of state-set progression is not dependent on precise values in the certainty matrix. The certainty matrix for the Computer Science Department is shown in Table 1. In this example, the probability of any particular door being closed is 0.6. Entries in the matrix can be zero, allowing the state set to collapse when certain percepts are discerned; however, robustness is limited because in the real world, anything is possible.

Now we can reexamine the earlier example with a more realistic robot using this certainty matrix. Once again, assume that the original state set is {1-2, 2-3}. However, now, the states have certainty factors of 1.0 and 0.2, respectively. (Only the relative magnitudes of certainty factors are important.) How will the state-set progress if the robot simultaneously detects an open hallway on its left and an open door on its right?

State 2-3 will progress potentially to states 3, 3-4, and 4. However, states 3 and 3-4 can be eliminated because the likelihood of detecting an open door when the actual feature is a wall is zero. The likelihood of being in state 4 is the product of (1) the initial certainty factor for state 2-3, 0.2; (2) the likelihood of not detecting anything at node 3; and (3) the likelihood of detecting a hallway on the left and a door on the right at node 4.

The second likelihood occurs only if DERVISH fails to detect the door on its left at

	Wall	Closed Door	Open Door	Open Hallway	Foyer
Nothing detected	.70	.40	.05	.001	.30
Closed door detected	.30	.60	0	0	.05
Open door detected	0	0	.90	.10	.15
Open hallway detected	0	0	.001	.90	.50

Table 1. Certainty Matrix for the Computer Science Department at Stanford University.

node 3 (either closed or open), [(0.6)(.4) + (1 – 0.6)(.05)], and correctly detects nothing on its right, .70.

The third likelihood occurs if DERVISH correctly identifies the open hallway on its left at node 4, .90, and mistakes the right hallway for an open door, .10.

The final formula, (0.2)[(0.6)(0.4) + (0.4)(0.05)](0.7)[(0.9)(0.1)], yields a certainty factor of 0.003276 for state 4.

State 1-2 will potentially progress to states 2, 2-3, 3, 3-4, and 4. Again, states 2-3, 3, and 3-4 can all be eliminated because the likelihood of detecting an open door when a wall is present is zero. The likelihood for state 2 is the product of the initial certainty, (1.0); the likelihood of detecting the door on its right as an open door, [(0.6)(0) + (0.4)(0.9)]; and the likelihood of correctly detecting an open hallway on its left, 0.9. The certainty factor for being at state 2 is then (1.0)(0.4)(0.9)(0.9) = 0.324. In addition, 1-2 will progress to state 4 with a certainty factor of 4.259×10^{-6}, which is added to the certainty factor in the preceding paragraph to bring the total for state 4 to 0.00328. DERVISH would then believe strongly that it is in state 2 but retain the remote possibility that it is in state 4.

Assumptive Planning

Given our state-set representation, a traditional approach to planning would be to conduct an exhaustive search in the state-set transition diagram to create a full conditional plan that would prescribe action sequences for every possible series of perceptual input. Unfortunately, the cost of complete conditional planning is prohibitive for real-world robot navigation. Some roboticists bypass this problem by forgoing planning altogether, using entirely reactive systems (Slack 1993; Brooks 1986). Still others plan once and then execute the plan steps using reactive components (Gat 1992; Nilsson 1992). Our solution, however, addresses uncertainty by allowing for multiple planning instances, effectively interleaving planning and execution (Genesereth and Nourbakhsh 1993; Olawsky, Krebsback, and Gini 1993; Hsu 1990). In our assumptive-planning strategy, DERVISH plans by assuming that it is in the most likely state in its initial state set, then finding a sequential solution to the goal. Then, DERVISH executes the plan while it progresses its state set until the current most likely state either is at the goal or is inconsistent with the path.

To understand our motivation for this approach, consider an analogy to the way a human with a car might navigate to a goal. First, he/she will use an internal map to construct a (possibly partial) path to the goal, represented as a series of intersecting roads. Then, he/she will drive along each road and turn at each appropriate intersection until he/she reaches the destination.

Now let's examine the driver's thinking process more closely during the drive down a particular road. He/she does not waste time trying to determine the car's precise location along the road at every instant; he/she is satisfied with a rough idea of position and the knowledge that the car is still headed in the correct direction (that is, making progress toward the goal).

What happens if the driver sees a street sign? If it is an unfamiliar street, he/she simply continues driving. Although he/she is familiar with the street, he/she will update

his/her sense of position. The act of driving is not interrupted as long as the driver believes that he/she is making progress toward the goal, even if he/she cannot remember passing a familiar grocery store. However, if the sign convinces the driver that his/her assumption about his/her current location is wrong, he/she will replan. This replanning can occur if the driver overshot a turn or is not on the assumed road, perhaps because of an earlier wrong turn.

This analogy is almost identical to DERVISH's thought process. Because the robot knows its initial and final locations and has a map of the world, it constructs only one path to the goal and begins to traverse this path. Whenever DERVISH detects a percept, it looks at the topological map to determine its most likely position. As long as this new state corresponds to a node along the path, it continues execution without interruption, even if it missed some earlier expected percepts. DERVISH stops and replans when it is no longer on the path because it either has overshot the turn or is actually on a different hallway than previously assumed.

DERVISH is extremely robust, in part because state-set progression prevents the robot from becoming lost as a result of false percepts. A false or missing percept can cause DERVISH to temporarily assume the wrong state, but the correct state is retained in the state set and becomes the most likely state after one or more correctly detected percepts. In practice, DERVISH is often off by one or two nodes when it is moving down the hallway because its sensors are unreliable.

Why does this method work so well? The robot's positional uncertainty is almost always limited to the particular hallway in which it is located. In addition, because of the reliability of the hallway detector, the robot quickly detects when it has overshot a turn. Therefore, the current action, which is to move forward, usually moves the robot toward the goal. This action typically terminates in two situations: (1) when the robot reaches an intersection and (2) when the robot believes it is just outside the goal room. The first situation rarely results in an incorrect subsequent plan because the reliability of the open hallway detector tends to condense the state set at intersections. In the other situation, the robot examines the state set to see if the second most-probable state is also highly likely. If so, the robot proceeds to the end of the hallway to collapse its state set and then returns to the goal door.

The most convincing argument against using full state-set conditional planning is its extremely high computational overhead. Whereas an assumptive planner must search $O(a^k)$ states, a conditional planner must search $O(a^k p^k)$ states, where there are a possible actions and p possible percepts, and the plan length is k. Because we have 4 actions, our planner must search 4^{10} states for a 10-step plan. For the same plan length, because we have 15 percepts, a conditional planner would be required to search $4^{10}15^{10} = 60^{10}$ states, which is an increase of more than 11 orders of magnitude. For a detailed description of conditional planning, interleaved planning, and execution and their respective costs, see Genesereth and Nourbakhsh (1993).

There are, of course, disadvantages to assumptive planning. First, if the world is dangerous (for example, if it contains staircases), then, clearly, the assumptive approach can be deadly. More precisely, assumptive planning is dangerous any time the robot can take an irreversible action before realizing that its assumption is wrong; one partial solution is to modify the path planner to avoid dangerous regions. Another serious disadvantage is that an assumptive system usually cannot guarantee an upper bound on the execution-path length. Of course, in a real world with imperfect sensors and effectors, full conditional planning is usually unable to make any reachability guarantee either. In practice, assumptive planning produces near-optimal execution lengths if the initial assumption is correct and if replanning is rare (that is, if the sensors operate reasonably well).

Conclusion

DERVISH navigates well in spite of incomplete knowledge about the environment because it maintains a state set that explicitly captures its uncertainty and because it can retract its assumptions if they are found to be invalid. One current weakness is that our state set does not actually capture all the information DERVISH has available from the world. A more complete approach should incorporate metric information as well as the state of previously observed doors.

Another weakness is that DERVISH is unable to question many of its assumptions. A truly powerful robot navigator should be able to retract assumptions at multiple levels of abstraction; for example, DERVISH ought to be able to question not only its position but also the values of the certainty matrix, the stated width of doorways, the integrity of the topological map, and so on.

Although DERVISH's assumptive planner works well in general, in some environments, actions can have irreversible consequences. One way to address this shortcoming is to create a hybrid system that combines assumptive planning with an analysis of the state space to avoid dangerous action sequences.

DERVISH's sonar configuration allows it to perform well in office buildings. However, sonars suffer from both an array of inherent weaknesses and a lack of richness. The next step in our research agenda is the Bookstore Project, which will combine the assumptive planning system with purely vision-based sensors that provide a more complete picture of the world. We plan to design a system that can navigate the Stanford campus, coexisting with high-speed bicyclists and curious tourists and crossing a busy campus to fetch books from the Stanford Bookstore.

Acknowledgments

We would like to thank Michael Genesereth for his role in supporting the DERVISH Project. Thanks are also owed to Steve Ketchpel, Reid Simmons, Bromley Birchfield, and Terry Winograd for their helpful comments and suggestions.

References

Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1): 14–23.

Gat, E. 1992. Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. In Proceedings of the Tenth National Conference on Artificial Intelligence, 809–815. Menlo Park, Calif.: American Association for Artificial Intelligence.

Genesereth, M., and Nourbakhsh, I. 1993. Time-Saving Tips for Problem Solving with Incomplete Information. In Proceedings of the Eleventh National Conference on Artificial Intelligence, 724–730. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hsu, J. 1990. Partial Planning with Incomplete Information. Presented at the AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments, 27–29 March, Stanford, California.

Nilsson, N. 1992. Toward Agent Programs with Circuit Semantics, Technical Report, STAN-CS-92-1412, Computer Science Department, Stanford University.

Olawsky, D.; Krebsback, K.; and Gini, M. 1993. An Analysis of Sensor-Based Task Planning, Technical Report, 93-94, Computer Science Department, University of Minnesota.

Slack, M. G. 1993. Navigation Templates: Mediating Qualitative Guidance and Quantitative Control in Mobile Robots. *IEEE Transactions on Systems, Man, and Cybernetics* 23(2): 452–466.

Illah Nourbakhsh is a graduate student in the Stanford University Computer Science Department. His research has included protein-structure prediction under the GENOME Project, the interleaving of planning and execution, software reuse, and mobile robot navigation. He is a member of the Logic Group at Stanford.

Rob Powers received his Bachelor's degree in computer science from Brown University in 1993 and his Master's in computer science from Stanford University in 1994. He is currently working with the Logic Group at Stanford, focusing primarily on autonomous mobile robotics.

Stan Birchfield is a graduate student in electrical engineering at Stanford University. He is sponsored by a National Science Foundation Graduate Fellowship and received his bachelor of science in electrical engineering from Clemson University in 1993. His research is aimed at determining three-dimensional world structure from vision.