# YODA

## The Young Observant Discovery Agent

*Wei-Min Shen, Jafar Adibi, Bonghan Cho,*
*Gal Kaminka, Jihie Kim, Behnam Salemi, and Sheila Tejada*

■ The YODA Robot Project at the University of Southern California/Information Sciences Institute consists of a group of young researchers who share a passion for autonomous systems that can bootstrap its knowledge from real environments by exploration, experimentation, learning, and discovery. Our goal is to create a mobile agent that can autonomously learn from its environment based on its own actions, percepts, and missions. Our participation in the Fifth Annual AAAI Mobile Robot Competition and Exhibition, held as part of the Thirteenth National Conference on Artificial Intelligence, served as the first milestone in advancing us toward this goal. YODA's software architecture is a hierarchy of abstraction layers, ranging from a set of behaviors at the bottom layer to a dynamic, mission-oriented planner at the top. The planner uses a map of the environment to determine a sequence of goals to be accomplished by the robot and delegates the detailed executions to the set of behaviors at the lower layer. This abstraction architecture has proven robust in dynamic and noisy environments, as shown by YODA's performance at the robot competition.

The suspense is high. We stare intensely at the robot with one eye, keeping the other one out for any surprises. As YODA approaches the director's office, it seems to be moving slower than ever before. It looks for the door and slowly starts moving into the room. Our minds seem to be sharing the same thought—"YODA, don't fail us now." YODA announces the room for the meeting and then the time: "The meeting will start in one minute." Perfect! We scream, and it is all over. YODA's final run in the Fifth Annual AAAI Mobile Robot Competition and Exhibition (held as part of the Thirteenth National Conference on Artificial Intelligence [AAAI-96]) was perfect—an exciting climax to our six months of hard work.

The YODA team was formed when a few of us felt the urge to do something with the big Denning robot at the Information Sciences Institute (ISI). The final push occurred when Rodney Brooks came to the University of Southern California (USC) and showed the video clips of his robots at the Massachusetts Institute of Technology (MIT). These clips demonstrated some interesting ideas about AI and looked like a lot of fun. Our goal became to transform our then-lifeless robot into YODA (figure 1), an autonomous agent that would learn to explore and interact in a real environment.

We decided that the Office Navigation event in the robot competition was to be our first milestone in working toward this goal. It would provide us a context in which to direct our efforts. We developed a general architecture that would allow YODA to perform the competition task and accommodate the learning and discovery tasks that we would later add. The following sections describe this architecture in more detail and provide an account of YODA's performance at the competition and the challenges that we faced there.

## General Architecture

The current YODA system comprises a Denning MRV-3 mobile robot and an on-board portable personal computer. The robot is a three-wheel cylindrical system with separate motors for motion and steering. It is equipped with 24 long-range sonar sensors, 3 cameras for stereo vision, a speaker for sound emission, and a voice-recognition system. The communication between the robot and the control


*Figure 1. YODA Wandering the Halls of the Information Sciences Institute.*

computer is accomplished through an RS232 serial port using a remote programming interface (Denning 1989). The robot is controlled by a set of commands, and the sensor readings include sonar ranges, motor status, and position vectors (vision was not used in this competition). As with any real sensing device, the sensor readings from the robot are not always reliable, which poses challenges for building a robust system.

YODA's software is implemented in MCL2.0 on a MACINTOSH POWERBOOK computer. The control architecture (figure 2) consists of three layers and is designed to integrate deliberate planning with reactive behaviors. The top layer is a dynamic planner that can find the shortest path between any pair of rooms on the map. Because the empty conference rooms are not known at the start, the robot must have the capability of finding the shortest path to each conference room until an empty room is found, then plan the shortest route between the professor and director rooms. At the middle layer of the architecture, each shortest path found by the planner is expressed as a sequence of behavioral actions with appropriately assigned parameters. YODA's four generic behaviors are (1) passing through a doorway, (2) traveling to a landmark, (3) audio communicating, and (4) detecting an empty room. Each of these behaviors is implemented at the bottom layer of the architecture in terms of the basic actions (forward, backward, and turn) and perceptions (sonar vectors, *x-y* locations, and angles).

Notice that the main idea behind this architecture is abstraction, with each layer being an abstraction of the layer that is immediately below. The top layer, as shown in figure 2, only reasons about the relationships between rooms; so, when YODA starts out at the director's room, the dynamic planner decides which conference room to visit first. The landmark planner expands the high-level plan by determining the route between rooms in terms of landmarks, such as doorways, hallways, and corners. Once the route has been planned, then the behavior controller is called to move the robot safely from landmark to landmark. This configuration was a large contribution to the building of a robust performance system, as demonstrated by YODA's success in the competition.

## Dynamic Planner

On the top layer of the architecture, the dynamic planner determines all the mission-oriented long-term behaviors of the robot. For the Office Navigation event, there are two mission-oriented behaviors or goals: (1) find an empty room and (2) notify the professors of the meeting time and place. To accomplish these goals, the planner must find the shortest path between a set of rooms as well as determine the necessary actions to interact with the environment. The planner needs to be dynamic because it must decide the current plan based on information that it is acquiring from the environment. For example, when trying to find the empty conference room, YODA needs to move from its current room to the nearest conference room and then detect if the room is empty. If it is occupied, then the robot moves to the nearest unchecked

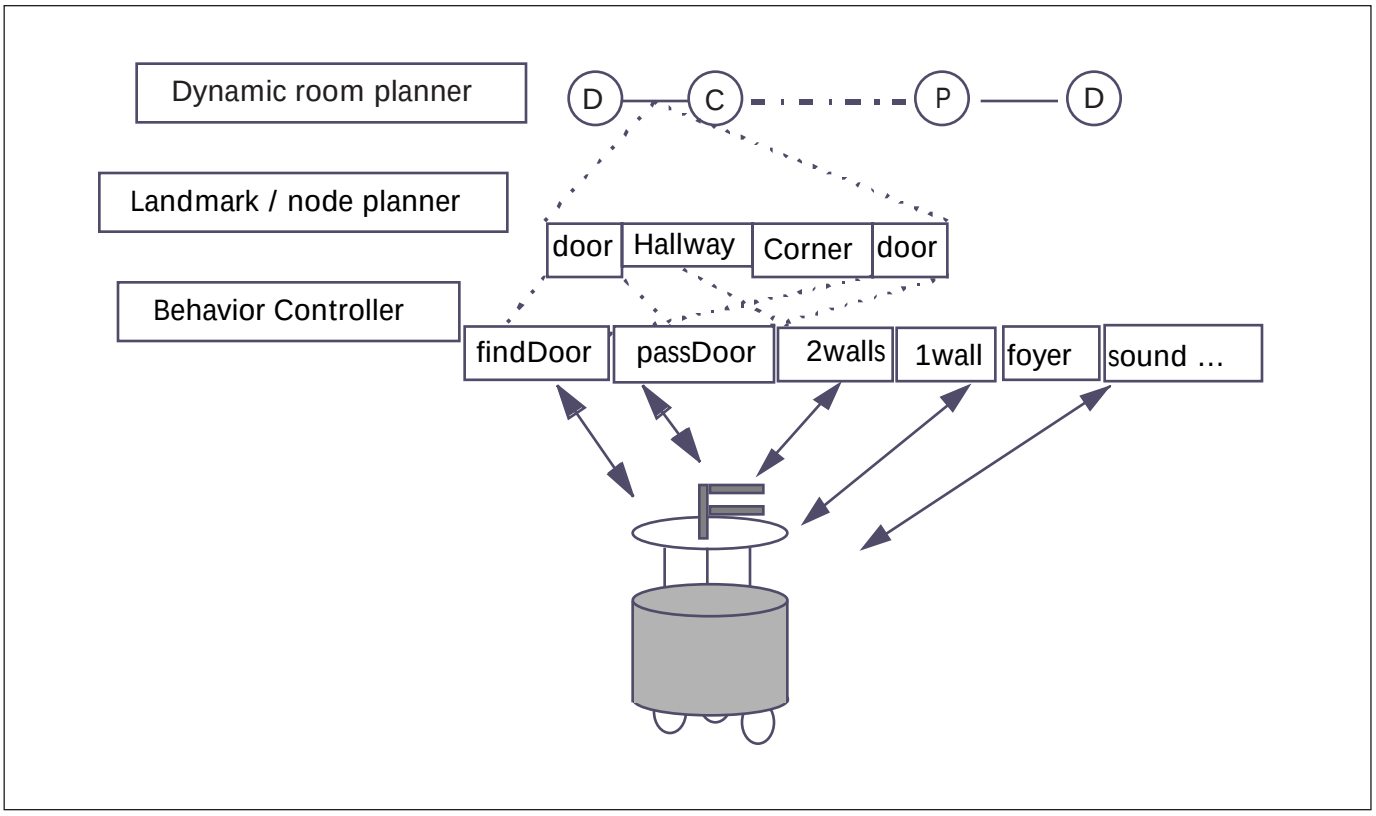



*Figure 2. The Three Abstraction Layers of YODA's Control Architecture.*

conference room. However, if the room is empty, then the current plan is to satisfy the next goal of finding the shortest route to notify the professors.

The current plan is determined using the information acquired from the environment in conjunction with a set of tables that provide the shortest-path information corresponding to the current situation. These tables are built from parsing the input map. The *input map* (figure 3) consists of a list of records, one record for each location or node. A *record* contains the node type (corridor, room, foyer), adjacent nodes, and the distances to adjacent nodes. The planner builds three tables by parsing this map. It first computes the shortest paths among the conference and professor rooms based on the connections and distances of the nodes. These paths are stored in a table called the *path table*. Each path consists of a list of nodes and the length of the path. Once the path table is created, the system then builds the *notify table* by computing the shortest route to visit the professors' rooms. The notify table consists of a list of all the nodes in the route and the route length. It can be used to notify the professors given the empty conference room. Finally, the scenario table is built based on these two tables.

A *scenario* is a permutation of the set of conference rooms. Each scenario denotes the order in which the conference rooms are visited. For example, given three conference rooms, C1, C2, and C3, one of the permutations is (C1, C2, C3), meaning that C1 is visited first, then C2, then C3. The *scenario table* records the total route lengths for the different possibilities of empty conference rooms for each scenario. For this example scenario, the planner computes the total route lengths for three cases: (1) the total route length for visiting C1 first and then the professors' rooms, assuming C1 is empty; (2) the total route length for visiting C1 first, then C2, and then the professors' rooms, assuming C1 is occupied, and C2 is empty; and, finally, (3) the total route length for visiting C1 first, then C2, then C3, and then the professors' rooms, assuming C1 and C2 are occupied, and C3 is empty. These three route lengths are stored in the table with the scenario. The number of cases depends on the number of conference rooms.

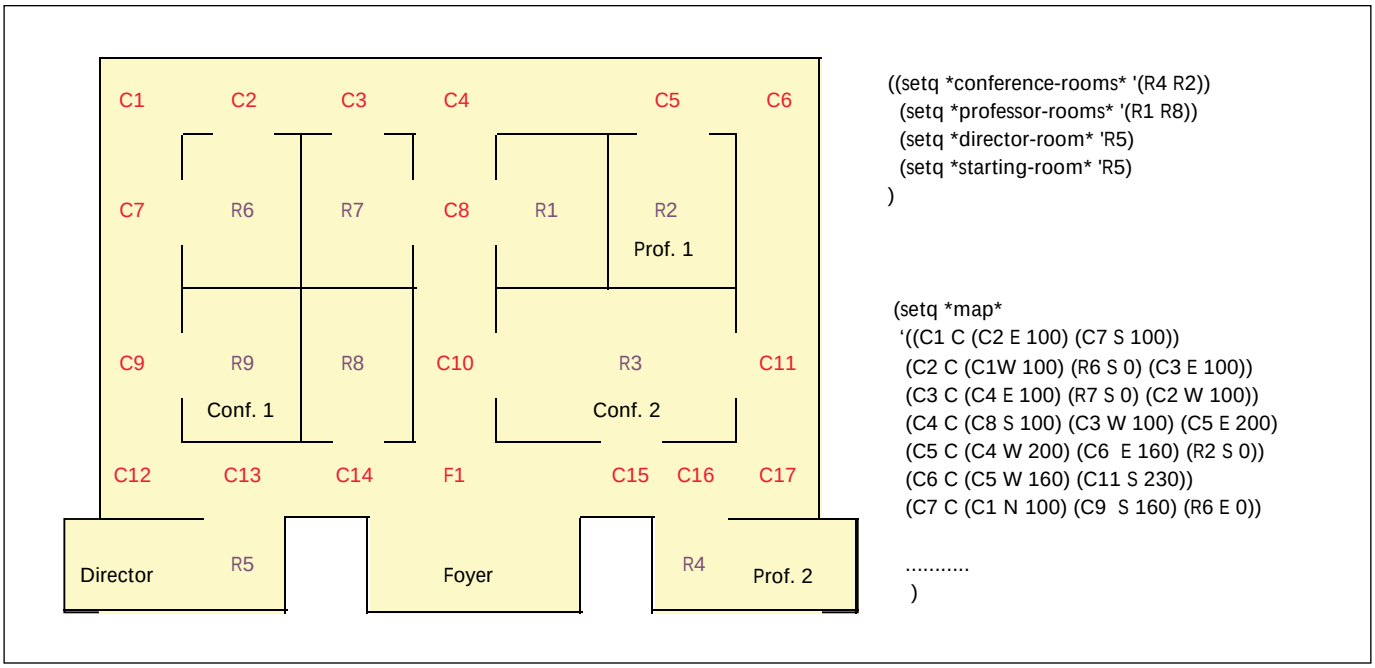



*Figure 3. An Example of the Input Map.*

Given the scenario table, there are at least three ways to select one of the scenarios: We can select the scenario with the minimum total route length when the first conference room is empty. In the given example, the system will select the first scenario shown in figure 4. The second strategy selects the scenario that has the minimum total route length for the case where only the last conference room in the sequence is empty. In the given example, this strategy will select the third scenario. The third strategy is to select the scenario with an average minimum, which is the second scenario in the example. We used the first strategy to select a scenario for the competition. By building the tables from bottom to top (from path table to scenario table), we not only avoid redundant computations in the future (during execution) but also save recomputations while we build the tables.

## Landmark Planner

At the middle layer of the architecture, the *landmark planner* reasons about each plan found by the high-level planner in terms of behavioral actions. This layer also controls the execution of the high-level plan and the time-estimation task involved in notifying the professors. Once a scenario is selected, the scenario is executed as travel through a sequence of conference rooms. When a conference room is found empty, the plan execution is based on a sequence of professors' rooms that were already planned as the shortest path. The room-to-room traveling, in turn, is executed as travel through a sequence of nodes in the room-to-room path.

There are various types of navigation between two nodes. The landmark planner expands the high-level plan into a set of low-level navigation behaviors depending on the types of the two nodes. For example, "passing through the doorway" is the navigation type necessary to connect from a hallway type to a room type and "recognizing a landmark" to connect from a hallway type to a hallway type (or foyer). Although most low-level behaviors are specified at this level of the execution hierarchy, some behaviors, such as "detecting an empty room," have already been specified in the high-level plan.

This hierarchical plan execution enables the plan to be safely recovered in case of a crash. The plan can be executed from the point of the crash instead of the beginning. The hierarchical execution keeps the current status hierarchically (for example, the current room, the current node) so that the point of the execution at the time of the crash can easily be located in the whole sequence of the overall plan.

Our time estimation is based on the time data recorded during the plan execution. YO-

DA uses the time data from its past activities (from the beginning of a competition run to the point where the estimation is needed for notifying the professors). The key idea for accurate estimation is the use of multiple types of time data. The collected time data are organized by the various types of robot behavior (for example, "recognizing a landmark," "passing though a doorway") and used for estimating the execution time for each type occurring in the future plan. A default value is used for unavailable data. The estimation might require interpolation or averaging techniques to adapt the data for parameterized behaviors. For example, the data for the behavior of recognizing a landmark includes a distance and a time. When we have more than two data items for recognizing a landmark, we calculate an average speed.

## Behavior Controller

This subsection describes the design rationale and implementation of each behavioral action. They are, in many ways, similar to the behavior-based systems reported in Arkin (1987). Each behavior in itself is an independent program module that transitions the robot from one state to a desired state in a way that is as robust as we can achieve. To deal with imprecision in sensor readings, each behavior abstracts only the necessary perception information from the raw readings. The ability to focus attention and ignore those irrelevant sensor readings contributes greatly to the robustness of these behaviors.

**Passing through a Doorway** The basic idea behind the behavior of passing through a doorway is symmetry. When in the vicinity of a doorway, this behavior computes the sum of the sonar readings on both sides of the robot when heading toward the doorway. These sums are then compared to each other. If they are roughly equal to each other, then the robot concludes that it is at the center of the doorway, and it moves forward for a distance and computes the symmetry values again. Of course, at the same time, it also checks if the front sonar readings are sufficiently large so that there is indeed room to move forward. If the sum on one side is sufficiently larger than the other, then the robot first turns 45 degrees toward the side that has a larger sum and then forward for a short distance before doing another symmetry computation. What made this approach work was to dynamically determine how many sensors on each side to sum and what the threshold is for being symmetric. One heuristic we use is as follows: When the robot is far from the doorway, pay more attention to the sonars that are at the front. As the robot moves closer to the door, pay more attention to the sonars that are at the sides, and the threshold for symmetry should be lower.

1. (C1,C2,C3): (100,150,190)
2. (C3,C1,C2): (110,140,170)
3. (C2,C1,C3): (130,140,160)

*Figure 4. An Example of a Scenario Table.*

The symmetry idea works well even if the door width is very narrow compared to the size of YODA. As a cylinder of 90 centimeters (cm) in diameter, YODA has used this idea to successfully pass many doors 100 cm wide. This approach, of course, must rely on the fact that sonar readings from the walls adjacent to the door are reliable. At the competition, because most of the walls were made of painted cardboard boxes, the sonar beams that hit the walls at about a 45-degree angle were mostly bounced away; so, YODA could not see the walls and believed that it was going in a good direction. To compensate for this uncertainty, we used another strategy that ignored these unreliable readings and focused only on the direction where the door is. This strategy is not as precise as the symmetry idea, but it is good enough to pass doors that have a larger width. (Doors at the competition were 110 cm wide).

Another challenge is to know when you have actually passed out the doorway. The problem is that in some cases, the walls for the doorway are thin; so, when the robot passes the door, the sonars are not able to detect them. YODA uses the information stored in the map about the distance needed to pass out the room into the hallway. Once YODA has traveled the necessary distance, it assumes that it has passed the doorway, even though it did not detect it.

**Traveling to a Landmark** This behavior is designed to guide the robot from one location to another along a certain path. One problem is that the locations of the starting and the ending positions are only known to a certain extent because a large amount of er-

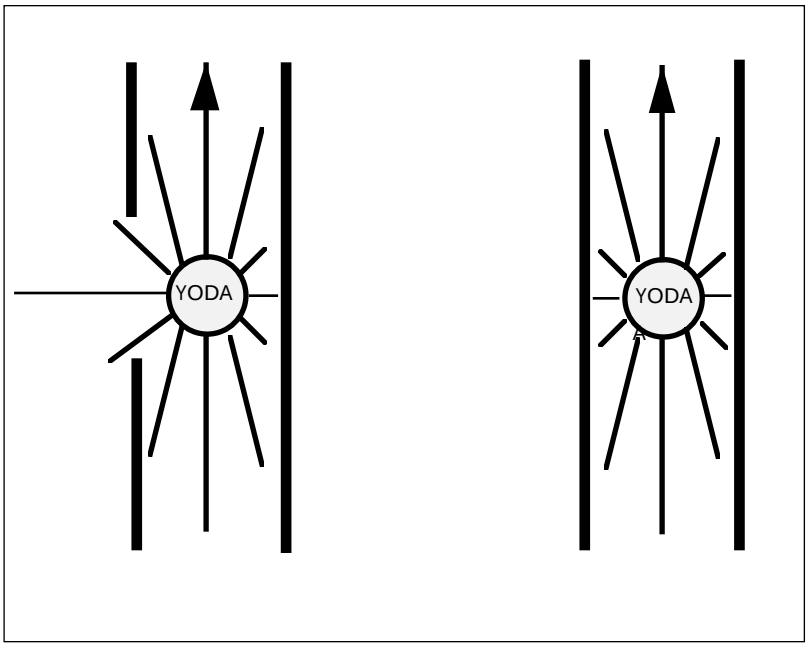



*Figure 5. A Signature is the Known Sensing Pattern for a Situation.*
A. An example of a signature for recognizing a doorway from a hallway.
B. A signature for recognizing a hallway.

rors can be accumulated from the previous actions that YODA has performed. Another problem is that there also might be obstacles along the path, whether they are static furniture or people who just walk by. The challenge is keeping the robot on the right path (such as a hallway) and at the same time avoiding obstacles. To overcome the uncertainty of goal location, we used the idea of finding a goal landmark as a way to detect whether the goal position has been reached or not. For example, if the goal position is a corner of a hallway, YODA uses the open space on the appropriate side as the landmark. If the goal position is at a door, then YODA actually looks for the door when the location to the goal position nears.

To navigate through certain terrain (such as a hallway or a foyer), we developed an idea called a *signature*. A signature is the known sensing pattern for a situation, and it is to be matched to the current sensor readings to determine the current position and orientation as well as find the best direction to go. For example, the signature for a hallway in term of the 24 sonar readings is a vector, such that the readings parallel to the hallway are large (figure 5a), but the readings against walls are small (figure 5b).

A signature has an orientation and can be rotated to match a given sonar vector. Thus, when YODA is in a hallway, it can rotate the hallway signature to find the best matching direction against the current sonar readings (figure 6). This direction will indicate the best direction to move to keep parallel with the walls. We call this "finding the best direction," and it is performed when the robot finds itself too close to the walls or obstacles.

To detect obstacles, YODA always keeps an eye on the direction it is moving. It has a virtual cushion space. This space is adjusted relative to its speed and orientation. For example, it has a large cushion in the direction it is moving, and it is even larger when the speed is high. If an obstacle is detected within this space in the moving direction, YODA stops immediately and then tries to find the best direction there. If an obstacle is detected in the cushion space but not directly in the moving direction, YODA concludes it is unsafe. It first slows down and then adjusts its direction according to the current signature.

The signature idea works well when the robot is following one wall (figure 5a) or two walls (figure 5b). However, there are no walls at a foyer, which caused problems for the signature approach. Strictly applying the signature can give you a direction that is not the best for reaching the goal, and the signature of foyer does not contribute much because all directions look the same. To overcome this problem, we used the idea of *virtual path*. Before YODA starts a movement from one place to the other, it first computes a path between the two places. Although this path can only be an estimation because of the sensor errors, whenever YODA finds itself in a position where there is no wall nearby, it tries to keep on the path as long as there are no obstacles. If some obstacles have forced YODA to deviate from the path, it detects the deviation and tries to come back to the path before moving any further. This way, we have a reliable way to recover from any deviation. At the competition, there were several occasions when this behavior saved YODA from becoming lost in the environment. YODA's movement control system is designed to bypass any obstacle or people during its movement toward a goal. However, if an obstacle is so large that the passage to a goal position is completely blocked, the robot stops and asks politely for people to move away.

**Audio Communicating** For the competition task, YODA needed to inform each of the professors of the time and place for the scheduled meeting as well as possibly interact with people who might be obstructing its path. Also, YODA was required to communi-

cate with the audience as much as possible, so that the audience could better understand the reasoning behind YODA's actions. YODA's voice is a collection of recorded phrases made by all the team members. This feature gives YODA a unique character.

Audio communication was also a helpful tool for debugging purposes. For example, when testing the passing through a doorway behavior, we noticed that if the walls of the doorway were thin, YODA would not say that it had passed the doorway, even though it was already in the middle of the hall. Thus, we concluded that YODA could not detect the thin doorways properly; so, we altered the behavior, as mentioned previously.

**Detecting an Empty Room** We use only the front sonars to detect whether a conference room is empty. Once YODA enters a room, it keeps computing the degree of changes in the sonar values. If the degree is greater than the given threshold, YODA considers the room not empty. Any movement of the occupants in the room, including vertical movement (walking toward YODA) or horizontal movement (walking side to side), is effective for this detection.

Although any movement of the occupants is effective for detection, to make YODA's behavior more pleasant and natural, we used a special strategy for the competition. We put a bowl of M&M candy on top of YODA and asked people in the room to come get some candy. This strategy ensured that if the room were occupied, someone in the room would move toward YODA, and the sonars would detect the movement. We assumed that everyone in the competition could be tempted by M&Ms, and they did indeed.

## The Competition

More than 20 teams—all with different backgrounds, robots, and approaches but with one unique aim to advance mobile robot technology—competed in the 1996 robot competition. Team YODA entered the exhibition hall two days before the competition. We have to admit we were not optimistic when we first arrived at the huge hall where all the teams were unloading their robots and setting up their equipment. At first, YODA had a serious hardware problem: It refused to boot up. After fixing the hardware problem, YODA was ready to test in the competition environment, and it took some time for YODA to adjust to the new environment.

The competition was conducted in three rounds: (1) preliminary, (2) semifinal, and (3) final. Although YODA had been tested in many different environments at USC/ISI, we soon discovered that YODA had problems operating in this new environment. The sonars did not work well because of the special materials used for the walls. The emitted sonar signal was bouncing away; so, YODA could not detect the walls at certain angles. This signal problem caused YODA difficulty when entering conference rooms and passing through the foyer. For the preliminary round, we solved this problem by covering the appropriate walls with a different material (YODA T-shirts). We also realized that YODA was too conservative in its time estimation. YODA would add about two minutes in anticipation of possible obstacles on its return route to the director's office. However, the distribution of obstacles was much lower than previously expected.

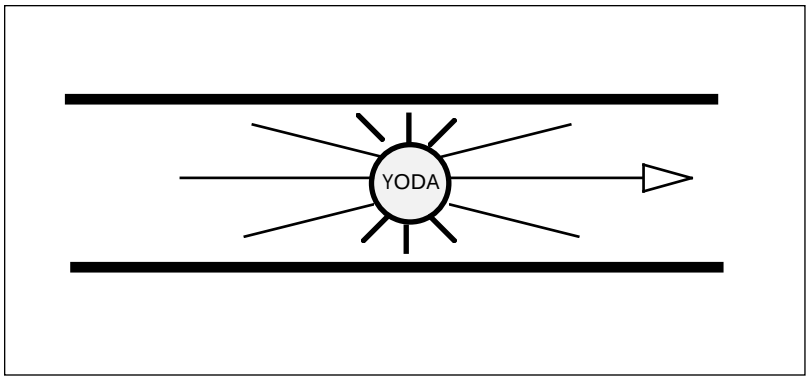



*Figure 6. The Signature for Recognizing a Hallway Shown in Figure 5b Has Been Rotated to Correspond with the Current Situation.*

To ready YODA for the semifinal, we spent the next 24 hours fixing these problems by changing several parts of code. We changed YODA's passing-through-a-doorway strategy as described earlier and used a more conservative approach to avoid hitting the side walls. Moreover, we eliminated the two-minute margin so that YODA can more accurately calculate its time estimation of the task. In the semifinal round, YODA performed perfectly and had no points deducted. YODA was ready for the final round.

During the final round, actor Alan Alda was filming for the PBS television program *Scientific American Frontiers*, which made us even more nervous. However, we enjoyed seeing Alan Alda interact with YODA. When YODA was checking to see if a conference room was empty, it was Alan Alda who walked up to YODA and took some M&Ms. The audience was also pleased with the use of different voices. You could never guess what voice YODA would use next. The rest of the story is history. YODA

performed the task perfectly and ended in the director's room, stating, "The meeting will start in one minute." Despite the fact that YODA was the biggest, heaviest, and oldest robot in the competition, a band of amateurs, working nights and weekends, was able to shape a lifeless robot into YODA, a robust autonomous agent. For the YODA team members, this feat was truly surprising and an extremely exciting experience.

## Related Work

YODA's abstraction architecture originates from a prediction-based architecture called LIVE (Shen 1991; Shen and Simon 1989) for integrating learning, planning, and action in autonomous learning from the environment (Shen 1994). It also bears many similarities to Erann Gat's (1992) ATLANTIS architecture, which integrates planning and reacting in a three-layer, heterogeneous asynchronous architecture for mobile robots. Different from ATLANTIS, however, YODA uses a closed-loop control theory (Goodwin and Sin 1984), as well as the idea of a signature, to maintain the awareness of its current states and control its low-level behaviors. Furthermore, YODA also uses landmarks to initiate and terminate the execution of sequences at the middle layer.

The use of signatures is closely related to Arkin's (1987) behavior-based approach and schema theories by Arbib (1981) and others. Our motivation for using declarative representations is to facilitate our long-term goals for learning models from the environment and collaborations among multiple heterogeneous agents. In this aspect, YODA seems to have a different philosophy about models of the world than Brooks's (1991, 1986) subsumption architecture.

### Acknowledgments

We would like to thank Ramakant Nevatia for providing us the Denning Robot. Special thanks to the various projects and people in the ISI Intelligent Systems Division for their moral support and their tolerance for sharing space (and occasionally "forces") with YODA.

### References

Arbib, M. 1981. Perceptual Structures and Distributed Motor Control. In *Handbook of Physiology—The Nervous System, II*, ed. V. B. Brooks, 1449–1465. Bethesda, Md.: American Physiological Society.

Arkin, R. C. 1987. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research* 8(4): 92–112.

Brooks, R. A. 1991. Intelligence without Representation. *Artificial Intelligence* 47(2): 139–160.

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

Denning. 1989. Denning MRV-3 Product Manual. Denning Mobile Robotics Inc., Wilmington, Massachusetts.

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.

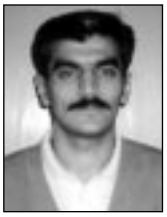
Goodwin, G. C., and Sin, K. S. 1984. *Adaptive Filtering, Prediction, and Control.* New York: Prentice Hall.

Shen, W. M. 1994. *Autonomous Learning from the Environment.* New York: W. H. Freeman.

Shen, W. M. 1991. LIVE: An Architecture for Autonomous Learning from the Environment. *ACM SIGART Bulletin* (Special Issue on Integrated Cognitive Architectures) 2(4): 151–155.

Shen, W. M., and Simon, H. A. 1989. Rule Creation and Rule Learning through Environmental Exploration. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 675–680. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

**Wei-Min Shen** is a research assistant professor in the Computer Science Department at the University of Southern California (USC) and a senior research scientist at the USC/Information Sciences Institute. He received his Ph.D. in computer science from Carnegie Mellon University in 1989 on the subject of autonomous learning. From 1989 to 1994, he was with the Microelectronic and Computer Technology Corporation, where he spent five years applying innovative research results to solving industrial problems. His current research interests include machine learning and discovery, autonomous agents and robots, intelligent information integration, and data mining. He has an international reputation in these research areas and has authored one book and about 45 technical papers in journals and conferences.



**Jafar Adibi** is a Ph.D. student in the Department of Computer Science at the University of Southern California (USC). He recived his B.S. in electrical engineering from Isfahan University of Technology, Isfahan, Iran, and his M.S. from USC. Currently, he is a graduate research assistant at the USC/Information Sciences Institute, where his current research interests include soft computing methods, knowledge discovery, and applications of AI in medicine.

**Bonghan Cho** is a Ph.D. candidate in the Department of Computer Science at the University of Southern California (USC). He received his M.S. from USC in 1989 and his B.S. from the Department of Computer Science and Statistics, Seoul National University, Korea, in 1987. He is currently working for the SOAR Project. His areas of interest include constraint-satisfaction problems, the scaling up of knowledge base systems, and computer networks.

**Gal Kaminka** is a graduate research assistant at the Information Sciences Institute, University of Southern California (USC), and a Ph.D. student in the Computer Science Department at USC. He completed his undergraduate education in computer science at the Open University of Israel. His interests are in the areas of agent modeling, agents that reason about themselves, failure and anomaly detection, and fuzzy set theory.

**Jihie Kim** is a computer scientist in the Information Sciences Institute at the University of Southern California (USC). She received her Ph.D. in computer science from USC in 1996 and her M.S. and B.S. in computer science from Seoul National University in 1990 and 1988, respectively. Her research interests include machine learning, intelligent agents, rule-based systems, knowledge-based systems for information retrieval, and electronic commerce.

**Behnam Salemi** is a graduate student in the Department of Computer Science at the University of Southern California and a graduate research assistant at the Information Sciences Institute. He received his B.S. in computer science from Shahid-Beheshti University, Tehran, Iran, in 1991. His research interests include autonomous learning and intelligent agents in the domains of robotics and education.

**Sheila Tejada** is a Ph.D. student in the Department of Computer Science at the University of Southern California and a graduate research assistant at the Information Sciences Institute. In 1993, she received her B.S. in computer science from the University of California at Los Angeles. Her research interests include machine learning, planning, intelligent agents, and data mining.