

YODA: The Young Observant Discovery Agent

**Wei-Min Shen, Jafar Adibi, Bonghan Cho, Gal Kaminka,
Jihie Kim, Behnam Salemi, Sheila Tejada**
Information Sciences Institute
University of Southern California
Email: robot-fans@isi.edu

Abstract

The YODA robot project at USC/ISI 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 AAI-96 Robot Competition served as the first milestone in advancing us towards 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 to be robust in dynamic and noisy environments, as shown by YODA's performance at the robot competition.

Introduction

The suspense is high. We stare intensely at the robot with one eye, while 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 AAI-96 Robot Competition was perfect -- an exciting climax for 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 MIT. They demonstrated some interesting ideas about artificial intelligence, 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 task in the robot competition was to be our first milestone in working towards this goal. It would provide us a context in which to direct

our efforts. We developed a general architecture which 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.



Figure 1

YODA wandering the halls of the Information Science Institute

General Architecture

The current YODA system is comprised of 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, three cameras for stereo vision, a speaker for sound emission, and a voice recognition system. The communication between the robot and the control computer is accomplished through an RS232 serial port using a remote programming interface [1]. 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 and this poses challenges for building a robust system.

YODA's software is implemented in MCL2.0 on a Macintosh Powerbook computer. The control architecture (see 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. Since 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: "Passing through a doorway," "Recognizing a landmark," "Audio communication," and "Detecting an empty room." Each of these behaviors are 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).

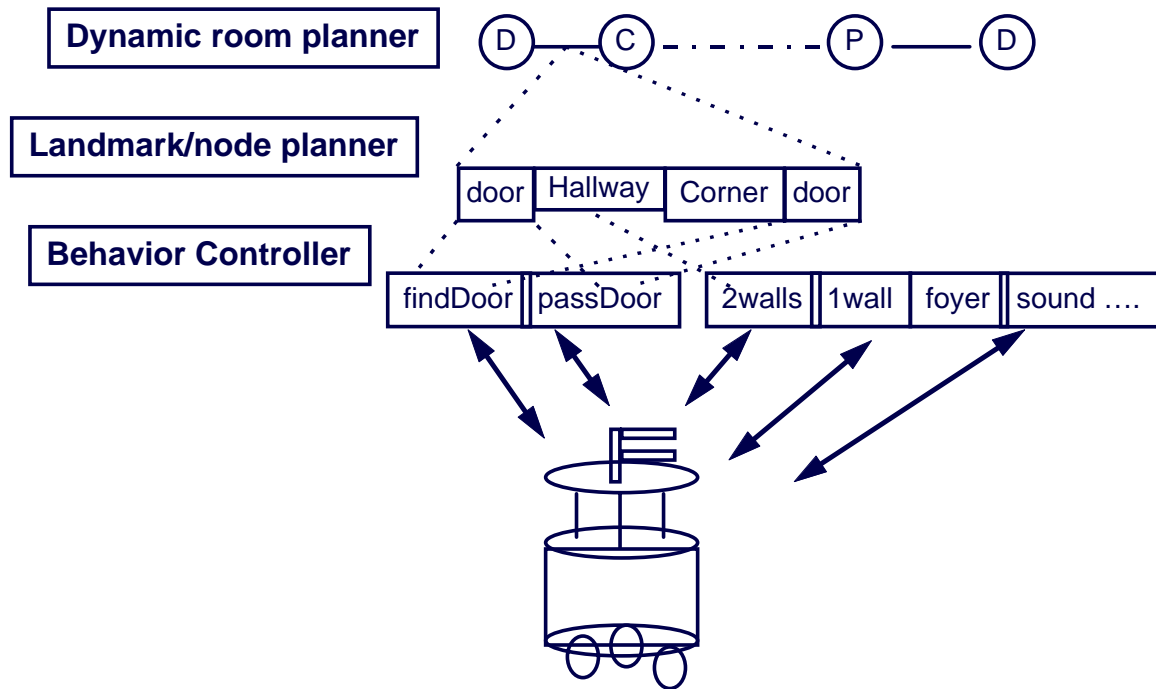


Figure 2

The three abstraction layers of YODA's control architecture

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 has a large contribution to building a robust performance system, as demonstrated by YODA's performance in the competition.

Dynamic Planner

On the top layer is the dynamic planner that determines all the mission-oriented long-term behaviors of the robot. For the office-navigation task there are two mission-oriented behaviors or goals: first, find an empty room, and then notify the professors of the meeting time and place. To accomplish these goals the planner must find the shortest the 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 conference room. But, if the room was empty, then the current plan would be 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 which provide the shortest path information corresponding to the current situation. These tables are built from parsing the input map. The input map 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 all of 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 all of the professor rooms. The table consists of a list of all the nodes in the route and the route length. The notify-table 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 all the professor rooms, assuming C1 is empty; (2) the total route length for visiting C1 first, then C2, and then all the professor 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 all the professor 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.

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

Figure 3

An example of a Scenario Table

Given the scenario table, there are at least three ways of selecting one of the scenarios. We can select the scenario that has 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 3. The second strategy selects the scenario that has the minimum total route length for the case that only the last conference room in the sequence is empty. In the given example this will select the third scenario. The third strategy is to select the scenario with an average minimum. This will select the second scenario in the example. We use 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 re-computations while building the tables.

Landmark Planner

At the middle layer, 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 traveling a sequence of conference rooms. When a conference room is found empty, the plan execution is based on a sequence of professor rooms which were already planned as the shortest path. The room-to-room traveling, in turn, is executed as traveling 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 called for the connection from a hallway type to a room type, and "Recognizing a landmark" for the connection from a hallway type to a hallway type (or foyer). While 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 from the very beginning. The hierarchical execution keeps the current status hierarchically (e.g., the current room, the current node, etc.) so that the point of the execution at the time of crash can be easily located in the whole sequence of overall plan.

Our time estimation is based the time data recorded during the plan execution. It uses the time data for the past robot activities (from the beginning 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 the data. The collected time data is organized by the various types of robot behavior (e.g., "Recognizing a landmark," "Passing through a doorway," etc.) 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 may require interpolation or averaging techniques to adapt the data for parameterized behaviors. For example, the data for "Recognizing a landmark" includes a distance and time. When we have more than two data items for "Recognizing a landmark," we calculate an average speed.

Behavior controller

This section describes the design rationale and implementation of each behavioral action. They are in many ways similar to the behavior-based systems reported in [2]. Each behavior in itself is an independent program module that transitions the robot from one state to a desired state in the most robust way that can be achieved. 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 sense 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 towards 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 will move forward for a distance and compute 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 will first turn 45 degrees towards the side that has a larger sum and then forward for a very short distance before doing another symmetry computation. What made this approach work is to dynamically determine how many sensors on each side to sum and what is the threshold 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, it needs to pay more attention to the sonars that are at the sides and the threshold for symmetry should be lower.

The symmetry idea works well even if the door width is very narrow compared to the size of YODA. Being a cylinder of diameter of 90cm, YODA has used this idea to successfully pass many doors of width 100cm. This, of course, must rely on the fact that sonar readings from the walls adjacent to the door are reliable. At the AAI competition, since most of the walls were made of painted cardboard boxes, the sonar beams that hit the walls at a 45 degree angle were mostly bounced away. So, YODA could not see the walls and believed that it is going in a good direction. To compensate this uncertainty, we used another strategy which 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 good enough to pass doors that have larger width. (Doors at the competition are 110cm wide).

Another challenge is to know when you have actually passed out of the doorway. The problem is that in some cases the walls for the doorway are very 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 of the room into the hallway. Once YODA has travelled the necessary distance it assumes that it has passed the doorway, even though it did not detect it.

Recognizing Landmarks

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 ending position are only known to a certain extent because a large amount of errors can be accumulated from the previous actions 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 how to keep the robot on the right path (such as a hallway) and at the same time to avoid obstacles. To overcome the uncertainty of goal location, we have 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 will use the open space on the appropriate side as the landmark. If the goal position is at a doorway, then YODA will actually look for the door when the location to the goal position nears.

To navigate through certain terrain (such as a hallway or foyer), we have 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 in order to determine the current position and orientation as well as finding 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 while the readings against walls are small (see Figure 4b).

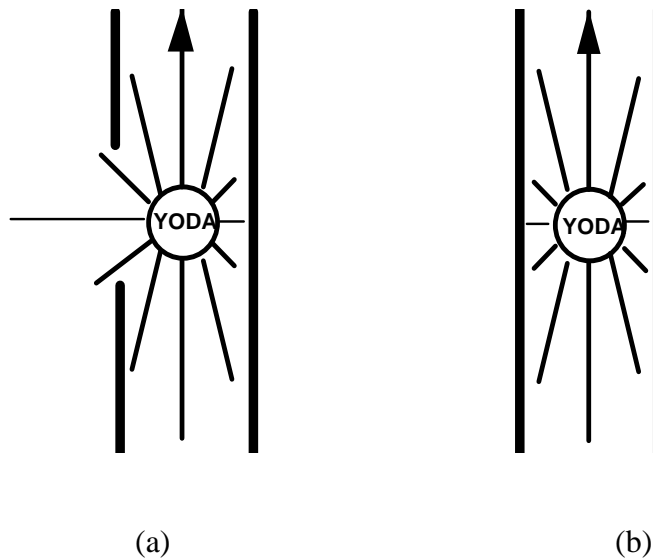


Figure 4

Figure 4a is an example of a signature for recognizing a doorway from a hallway. Figure 4b is a signature for recognizing a hallway.

The 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 5). This direction will indicate the best direction to move in order 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.

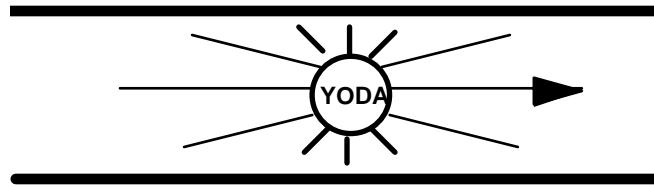


Figure 5

In Figure 5 the signature for recognizing a hallway shown in Figure 4b has been rotated to correspond with the current situation

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 its moving and it is even larger when the speed is high. If an obstacle is detected within this space in the moving direction, YODA will stop immediately and then try to find the best direction there. If an obstacle is detected in the cushion space but not directly in the moving direction, YODA will conclude “unsafe.” It will first slow down and then adjust its direction according to the current signature.

The signature idea works well when the robot is following one wall (Figure 4a) or two walls (Figure 4b). However, there are no walls at a foyer and this caused problems for the signature approach. Strictly applying the signature may 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 have used an 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 due to the sensor errors, whenever YODA finds itself in a position where there is no wall nearby, it will try to keep on the path as long as there is no obstacles. If some obstacles have forced YODA to deviate from the path, it will detect the deviation and try 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 are several occasions when this behavior has saved YODA from becoming lost in the environment. The movement control system of YODA is designed to bypass any obstacle or people during its movement towards a goal. However, if an obstacle is so large that the passage to a goal position is completely blocked, the robot will stop and ask politely for people to move away.

Audio Communication

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 communicate with the audience as much as possible, so that the audience could better understand the reasoning behind YODA’s actions. The voice of YODA is a collection of recorded phrases by all of 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 very thin YODA would not say that it had passed the doorway, even though it was already in the middle of the hall. This led us to believe that YODA could not detect the thin doorways properly, so we altered the behavior as mentioned previously.

Detecting an empty room

For this behavior we use only the front sonars to detect whether a conference room is empty. Once YODA has entered 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 towards YODA) or horizontal movement (walking side-to-side), will be effective for this detection.

Although any movement of the occupants are effective for detection, to make YODA’s behavior more pleasant and natural we employed 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 was to ensure that if the room was occupied someone in the room would move towards YODA, so the sonars would detect the movement. We assumed that everyone in the competition could be tempted by M&Ms.

The Competition

There were more than 20 teams competing in the 1996 robot competition -- all with different backgrounds, robots, and approaches, but with one unique aim to advance mobile robot technology. Team YODA entered the exhibition hall 2 days before the competition. We have to admit we were not very 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. It took some time for YODA to adjust to the new environment.

The competition was conducted in three rounds: preliminary, semi-final and final. Although YODA had been tested in many different environments at the USC/Information Sciences Institute, 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 caused problems when entering conference rooms and passing through the foyer. For the preliminary round we solved this problem by covering the appropriate walls with different material (YODA T-shirts). We also realized that YODA was too conservative in its time estimation. YODA would add about 2 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.

To ready YODA for the semi-final we spent the next 24 hours fixing these problems by changing several parts of code. We changed YODA's strategy of "Passing through a doorway", as described earlier, and used a more conservative approach to avoid hitting the side walls. Moreover, we eliminated the 2 minute margin which enabled YODA to more accurately calculate its time estimation of the task. In the second round (semi-final) YODA performed perfectly, having no points deducted. YODA was ready for the final round.

During the final round, actor Alan Alda was filming for a PBS program, which made us even more nervous. Yet, 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 use of different voices. You could never guess what voice YODA would use next. The rest of story is history. YODA performed the task perfectly, and ended in the director room stating, "The meeting will start in one minute." Despite the fact that YODA was the biggest, heaviest, oldest and slowest 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 was truly a surprising feat and an extremely exciting experience.

Acknowledgments

We would like to thank Dr. 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

- [1] Denning Robot Manual. *Denning MRV-3 Product Manual*. Denning Mobile Robotics Inc. 1989.
- [2] Ronald C. Arkin. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*. 1987.