AI Magazine Volume 16 Number 2 (1995) (© AAAI)

ERRATIC Competes with the Big Boys

Kurt Konolige

■ This article discusses the development of the robot ERRATIC, the second-place winner of the 1994 AAAI Robot Competition and Exhibition. I detail the robot's history and describe the perceptual and control architecture. The success of the robot is highlighted in a description of the robot's performance during the competition.

I enjoyed competing in the first two AAAI Robot Competition and Exhibitions: lots of energy, great exchange of ideas, and interesting solutions to the problems of indoor navigation. However, competing meant bringing the big robot, FLAKEY, with its nightmarish logistics: crating a 300-pound metal monster with delicate protrusions and several SPARCSTATIONS and their consoles, not to mention all the tools and spare parts for the ever-present hardware spastics. This year, I thought I'd try something different.

ERRATIC Lives!

In spring 1994, I taught a course at Stanford University in which we built three small robots modeled after FLAKEY but smaller and much cheaper. When I say *built*, I mean just that: The students were given all the electronic and mechanical components and, during the course, assembled and tested the circuit board (several smoky disasters here); built the chassis from bent aluminum sheet metal; mounted motors, encoders, and wheels; and soldered a gazzillion cables to connect everything. It was the most work and the most fun I've had teaching a class.

Figure 1 is a picture of one of the critters. As you can see, the result wasn't pretty, but it was wonderfully robust and good natured compared to other robot hardware I've worked on—in part as a result of simplicity. Thanks to Fred Martin and Randy Sargeant of the 6.270 robot-building class at the Massachusetts Institute of Technology (MIT), there is a small, low-powered microprocessor board that's great for robotics projects. The board is based on Motorola's 68HP11 UP, a highly integrated device that can hook up with sensors and effectors without too much additional hardware "glue." I made the decision to use the 6.270 board as the only on-board processor and communicate to a host computer by a radio modem link for more computationally intensive tasks. With a packet protocol for commands and data, the robot base becomes a server implementing a few basic functions: servo control of the motors, position integration for dead-reckoning movement, and sonar control (figure 2).

One of the unique features of the robot server is a front-pointing sonar mounted on a servomotor, the kind used in model airplanes. The front sonar can be aimed anywhere in the front half-circle of the robot. This design is cheap and easy to implement: It's easier to control 1 sonar on a servo than the 8 to 10 fixed sonars that would be needed to cover the same area. An early prototype I built used just the front-mounted sonar (actually twin sonars at a slight angle to each other), but this model proved unworkable because the servo couldn't turn fast enough to gather enough data (the sonars can't be used while the servo is turning because they won't produce reliable echoes). As a compromise, we used five sonars in the pattern shown in figure 3. The pivoting front sonar detects forward obstacles, and the fixed-side sonars look for walls, doorways, and oblique obstacles.

The whole robot weighs about 6 pounds without the battery, but the battery is a bruiser, a lead-acid 12-volt cell weighing 9 pounds. Perhaps this battery is overdesign, but the benefit is that the robot can run six to eight hours without having to recharge. In fact, I've never had to worry about one of the most onerous duties of mobile robotics—trying to keep a fresh set of batteries charged and switching them in. However, all the weight

Copyright © 1995, AAAI. All rights reserved. 0738-4602-1994 / $2.00

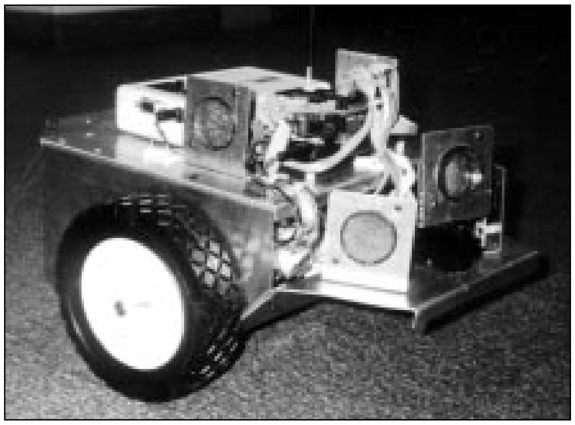

Figure 1. The ERRATIC *Mobile Robot Base.*

The whole platform is 6 inches high; 12 inches wide, including wheels; and 14 inches long. The 68HC11 processor is in the middle, flanked by two side-pointing sonars. The rest of the sonars are in the front of the robot. The radio modem, with its antenna, is also on top. The housing contains a large battery and the motors.

slows ERRATIC down a little: Top speed is about 20 centimeters a second, a slow walk. Still, it's light enough (15 pounds) that I was able to carry it in a cardboard box on board the plane to Seattle, Washington. One of the students from the class, Birdy (Bharadwaj S. Amrutur), took a SPARCSTATION, which was all we needed (unfortunately, the console had to be shipped, but next year, I'll use a portable MACINTOSH or PC).

Oh yes, the name: Because the robot mimics the motor and sonar portions of FLAKEY, a similar name was appropriate. ERRATIC had the right tone; it comes from the Latin root *err* (which means to wander, at least in this case). With its noisy, oscillating front sonar and slightly unstable motor control, ERRATIC does live up to its name.

Perceptual and Control Architecture

ERRATIC itself is just a server providing basic mobile robot capabilities. The main idea behind the client-server architecture is that the details of the server don't matter to the client, which communicates through a specified interface. Because FLAKEY was designed this way from the beginning, it was easy to use the control, interpretation, and planning software we developed for FLAKEY on the new ERRATIC platform—just a matter of changing some parameters to accommodate the differences in size, speed, and sonar placement. FLAKEY's control architecture has been written up in several places, so I just explain the basics here. At the center is the *local perceptual space* (LPS), an internal, egocentric representation of the local environment, where sensor readings are interpreted and registered (figure 4). The LPS gives the robot an awareness of its immediate environment and is critical in the tasks of combining sensor information, planning local movement, and integrating map information. The perceptual and control architecture makes constant reference to the LPS.

In Brooks's terms, the organization is partly vertical and partly horizontal. The vertical organization occurs in both perception (left side) and action (right side). Various perceptual routines are responsible for both adding sensor information to the LPS and processing it to produce surface information that can be used by object recognition and navigation routines. On the action side, the lowest-level behaviors look mostly at occupancy information to do obstacle avoidance. The basic building blocks of behavior are fuzzy rules, which give the robot the ability to react gracefully to the environment by grading the strength of the reaction (for example, turn left) according to the strength of the stimulus (for example, distance of an obstacle on the right).

More complex behaviors that move to desired locations are used to guide the reactive behaviors and use surface information and artifacts; they can also add artifacts to the LPS as control points for motion. At this level, fuzzy rules blend possibly conflicting aims into one smooth action sequence. Finally, at the task level, complex behaviors are sequenced, and their progress is monitored through events in the LPS. The horizontal organization comes about because behaviors can choose appropriate information from the LPS. Behaviors that are time critical, such as obstacle avoidance, rely more on simple processing of the sensors because it is available quickly. However, these routines can also make use of other information when it is available, for example, prior information about expected obstacles that comes from the map.

Controlling Executive: PRS-LITE

Behaviors are coordinated by a decision system called PRS-LITE, a real-time version of SRI's procedural reasoning system. The basic component of PRS-LITE is an *intention schema*, a finite-state machine whose arcs are labeled with conditions to check or goals to achieve. Each schema embodies a *strategy*, a sequence

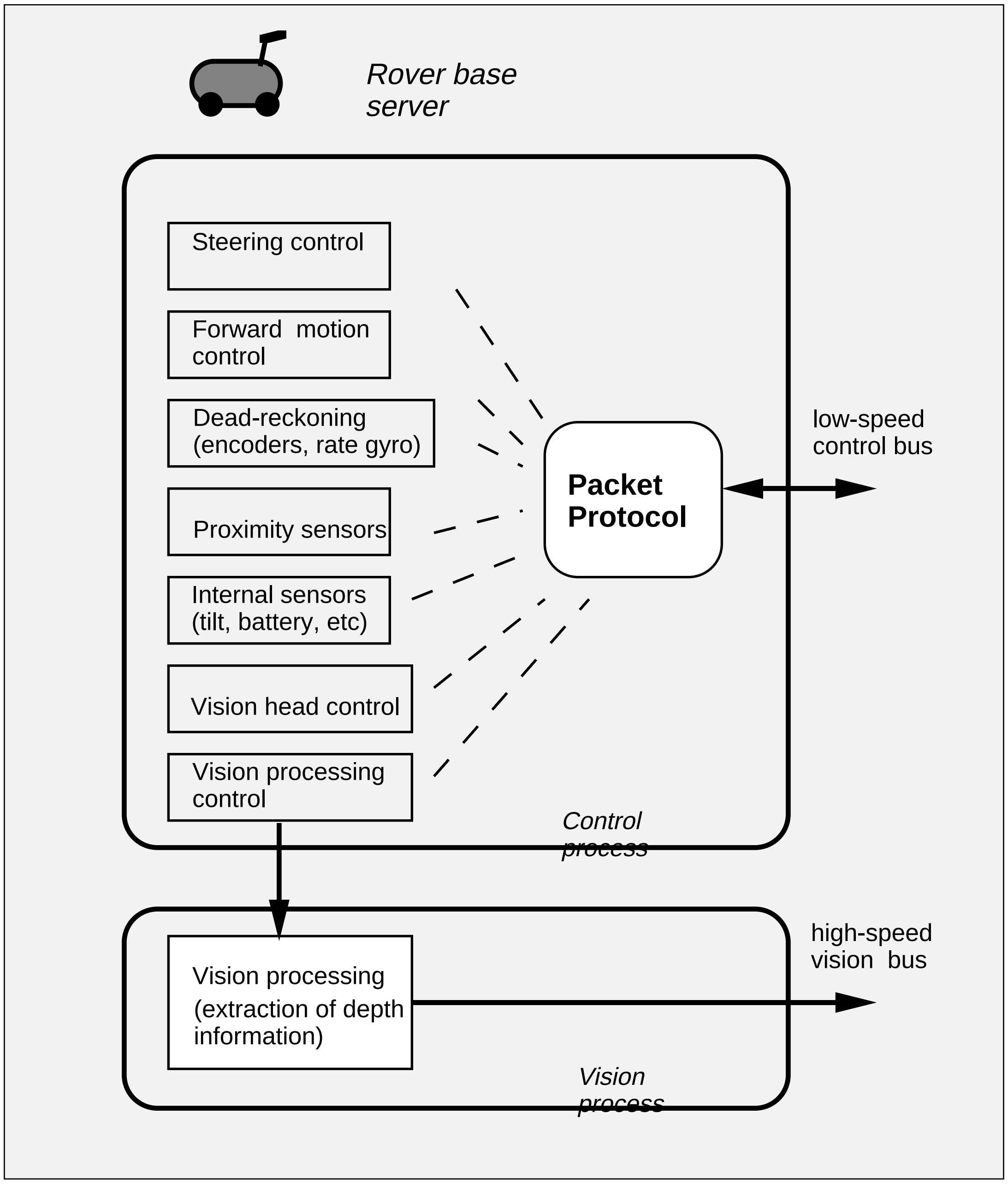

Figure 2. ERRATIC *Mobile Robot Base Functions.*

The base acts like a server, providing a set of low-level movement and sensor functions to clients. The interface is a packet protocol through an RS232 link.

of perceptual checks and behavior instantiations that accomplishes a goal. Some schemas are fairly simple. For example, to detect closed doors, a monitoring schema is fired up every time ERRATIC attempts to go through a doorway; if no progress is made after a fixed amount of time or if the sensors detect that the doorway is closed, the schema halts the current door-crossing behavior, updates a global map with the new information (more on this later), and signals the plan executive that the current plan has failed.

PRS-LITE is interesting because it incorporates a small amount of deliberation in its schemas but is still able to react to contingencies in real time. It throws away some of the more costly (and often useful) features of full PRS, especially the database of facts and goals

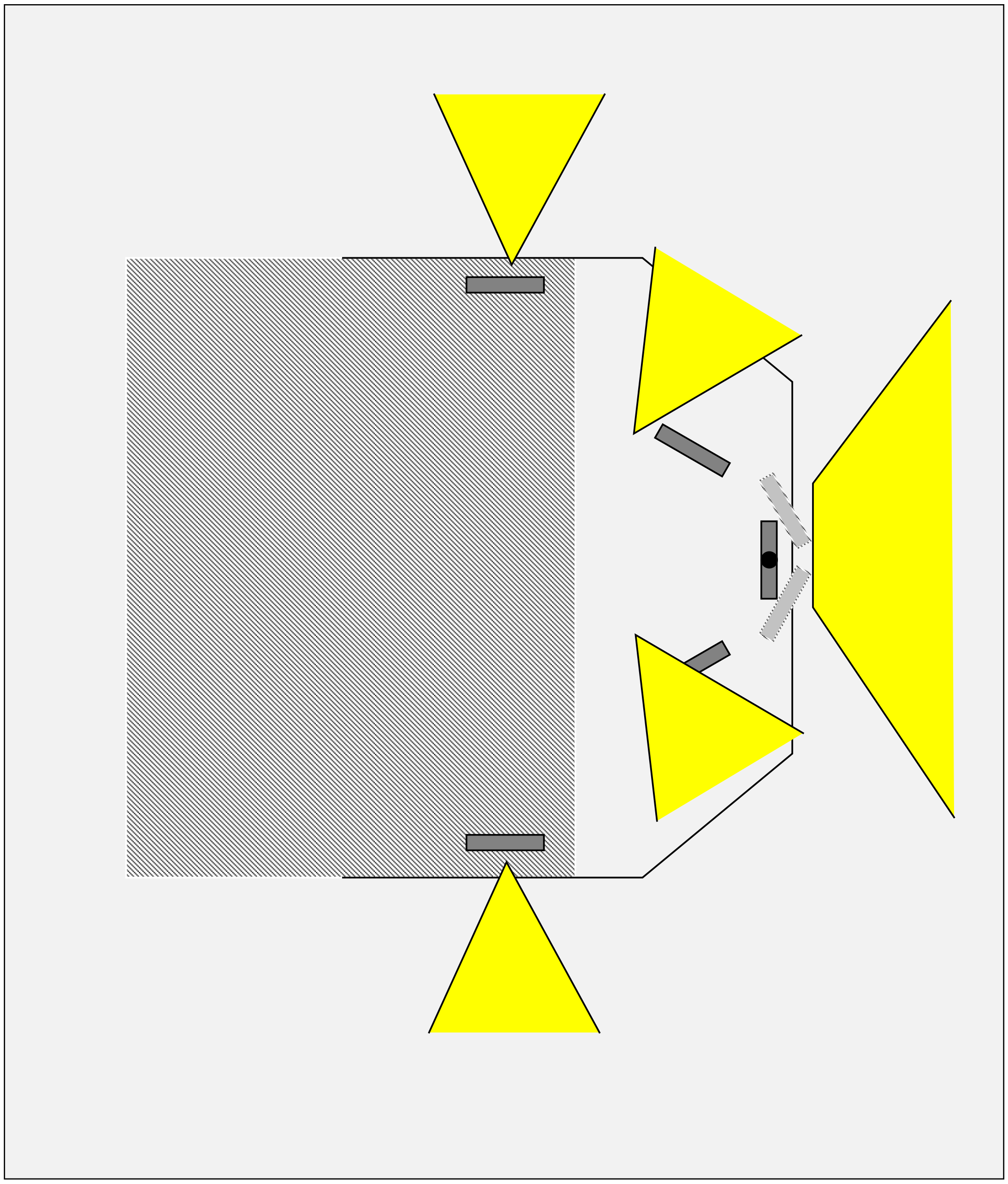

Figure 3. ERRATIC *Sonar Placement.*

The two side-pointing sonars in back are used for the extraction of wall segments and for obstacle avoidance. The other sonars cover the front hemisphere of the robot to detect obstacles. The single front sonar is mounted on a servomotor to swivel through 90 degrees.

that is used to trigger new schemas. Instead, every schema must be triggered explicitly by another schema, which gives a much less flexible system. However, the advantage is that PRS-LITE is fast: Written in C, it has a cycle time of 100 milliseconds, during which every executing schema is updated. This cycle time is very fast, considering that at any given moment, 10 or 15 schemas can be operating, monitoring various conditions and coordinating behaviors.

Interestingly, the ERRATIC team has not found a need for a larger planning capability to control ERRATIC. PRS-LITE is the main controlling agent and issues calls to higher-level functions, such as a navigation planner or a registration mechanism, when it requires their services.

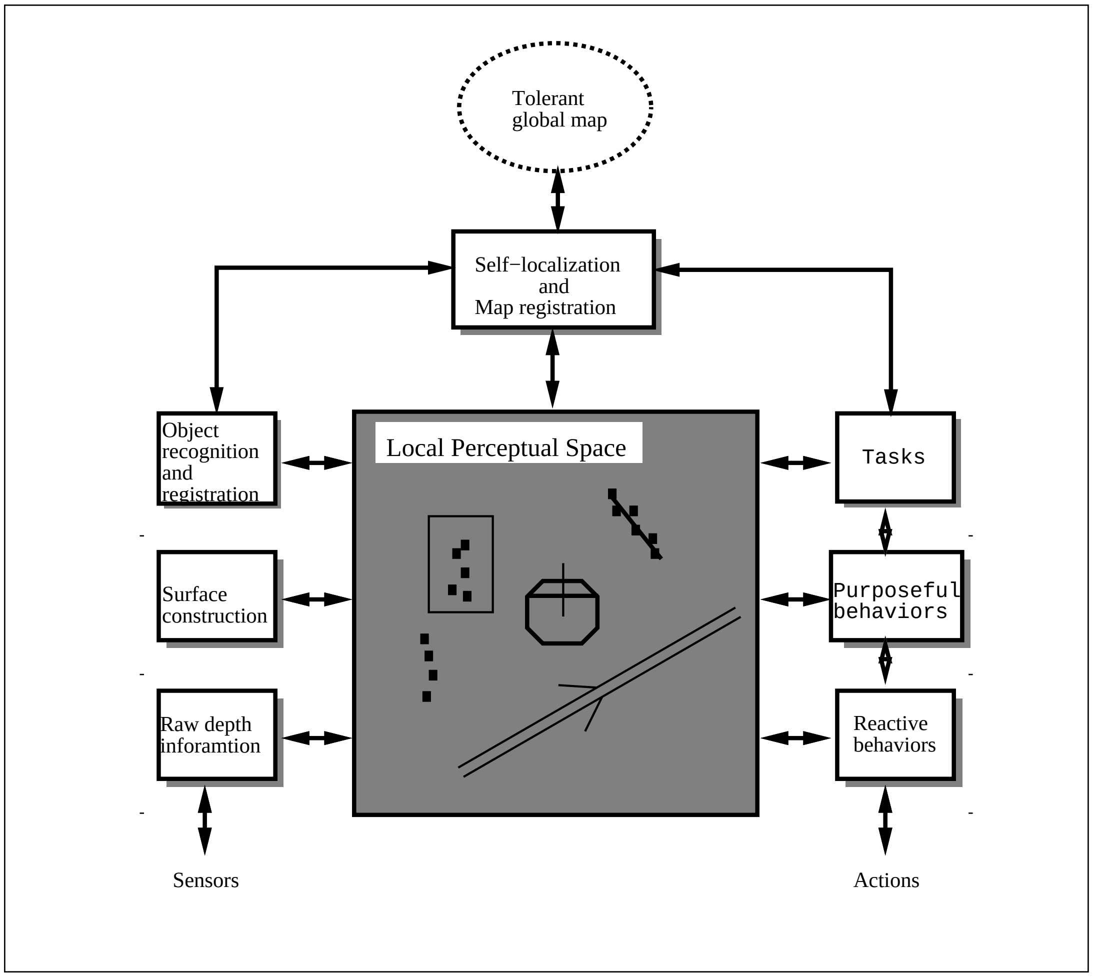

Figure 4. *FLAKEY'S System Architecture.*

Perceptual routines are on the left, action routines on the right. The vertical dimension gives an indication of the cognitive level of processing, with high-level behaviors and perceptual routines at the top. A map location module continuously matches local perceptual information to a stored global map, updating FLAKEY's global position. All modules operate independently in a distributed fashion.

Maps: Navigation and Registration

To navigate through extended regions, ERRATIC uses a global map that contains prior, imprecise spatial knowledge of objects in the domain, especially walls, doorways, and junctions of corridors. Using a map depends on the reliable extraction of object information from perceptual clues, and I (as well as others) have spent many frustrating years trying to produce object interpretations from highly uncertain sonar signatures. The best method I have found is to use extended aperture sonar readings. As ERRATIC moves along, readings from the side sonars are accumulated as a series of points representing possible surfaces on the side of the robot. This gives some of the resolution of a sensor with a large aperture along the direction of motion. By running a robust linear-feature algorithm over the data, ERRATIC can find wall segments and doorways with some degree of confidence. False-positive rejection, the bane of sonar sensors, is excellent for wall surfaces and reasonable for doorways, although we do get some false doorways when the robot rotates quickly. Extracting wall and doorway features makes it easy to build a global map by running ERRATIC around an area, but one problem must be overcome: By relying solely on ERRATIC's dead-reckoning ability, the map will be skewed terribly. For example, if I run ERRATIC 10 meters down the hall here at SRI and then turn and come back, the map will contain two hallways, and they won't be close to each other except at the point where ERRATIC turned around. ERRATIC's dead-reckoning ability is particularly poor, an error rate on the order of 10 percent for rotations and translations (to be contrasted with some of the very precise robots at the competition whose errors were closer to 0.1 percent). To say it is a deliberate design decision is partly correct: It's much easier to make imprecise robots. However, it also helps to wean students from "dead-reckoning dependency" and have them cope with the vagaries of real-world navigation. Interestingly enough, the robot that arguably had the best dead-reckoning (RHINO from the University of Bonn) relied on it too much and lost its orientation in a large room during the preliminary trials.

What to do? The usual answer is to register the robot based on its sensing. Take the case of ERRATIC: It goes up and down the same corridor. If it can make the assumption that it is seeing the same walls on its way back, then it can realign its internal idea of where it is to correspond with the walls it has already seen. The beauty of this idea is that the original map the robot makes does not have to be precise (and it can't be if dead reckoning is poor). The robot doesn't have to survey the area, just make a map that has approximately the correct dimensions. The registration process will keep the robot's position updated with respect to the objects in the environment, assuming, of course, that the robot can find and match objects of the right sort. For example, going down a long corridor with no features, the robot will stay correctly registered in the center of the hallway, but the error in its longitudinal position will grow. Because ERRATIC's registration process only used doorways—and not corridor junctions—we had some problems with localization during the finals of the competition, as I explain later.

Now ERRATIC can make a reasonably complete map of its office environment just wandering around. It helps to turn on a few exploration strategies, for example, following a corridor until it ends and then looking for likely openings. However, for the competition, we just ran ERRATIC by hand and saved the resultant map (deleting a few of the false doorways that were found). From this labeled metric map, a simple algorithm extracts a topological map of corridors, rooms, junctions, and doorways that is suitable for use by a simple topological planner. The planner takes the robot's current position (room, corridor) and a goal and produces a sequence of intermediate goal points at junctions and doorways. A plan executive schema in PRS-LITE takes these plans and fires up appropriate behaviors in sequence, monitoring their performance and updating the map to reflect closed doors and blocked corridors.

The Competition

As in previous contests, there is always a bit of confusion about the rules, especially as teams start trying out their robots in the new environment and discover problems. Our team's biggest concern was getting into the finals so we could show off our little monster. We decided on the most certain way of completing the navigation task—use an initial metric map of the arena (made as described earlier). Using this map gave ERRATIC a big penalty hit, but it had some pleasing results too. Most importantly, it meant that ERRATIC had a good idea of where doorways were, so it could detect closed doors fairly easily. Because the topological map was distributed free of charge, most other teams elected to use it without any metric information; as a consequence, they couldn't easily distinguish a closed door from a wall and would spend a lot of time trying to decide where they were in the topological map. As slow as ERRATIC was physically, it could beat the larger robots because it moved purposefully and continually.

In retrospect, our team could have had the best of both worlds because it was possible to accumulate metric information in the preliminary rounds and use it in the finals. We could have had ERRATIC start with just the topological map and build up its metric information as it moved toward the goal room, but we wanted to make sure we qualified, and using the metric map was the best way to achieve that goal.

ERRATIC did have several problems in the preliminaries. First, it would only poke its head into the goal room. I hadn't read the rules carefully enough; they stated that the whole body had to be inside the room. It took a while to fix the offending behavior, so we squandered a goodly amount of time in the preliminaries, finishing fifth. However, we did make it.

The second problem was one of interpretation. ERRATIC's topological planner didn't make use of metric information; so, it would pick a path to the goal based on the number of nodes visited. To test recovery from unexpected contingencies, the judges blocked a door to the goal room in the second preliminary round. They blocked the closest door to the robot, but the other door was just as close on the topological map, and ERRATIC chose that route. The judges had to wait until ERRATIC made a choice before moving to block the door.

ERRATIC was in the finals! I couldn't stay because of vacation plans, so the responsibility for getting ERRATIC ready and running was left to Birdy. I worried that something in the control architecture would break, that the hardware would fail, and so on.

For the finals, our team put a balloon on ERRATIC so that the audience could see it wandering the halls. Birdy says it was a cool sight to see the balloon bobbing along, marking the progress of the robot. The judges had decided to make the finals particularly difficult: The closest door of the goal room (in both a metric and a topological sense) was blocked, and a key connecting corridor was also blocked. ERRATIC was running fifth and last, and all but Stanford University's DERVISH had failed to make the goal; most were stymied by the blocked corridor.

ERRATIC started in fine style, balloon trailing behind, and the audience cheered when it recognized the closed door and started on an alternate path. Now, the serendipity of the topological planner surfaced: ERRATIC chose a longer path that completely avoided the blocked corridor. It looked like ERRATIC might win. Even with the high time penalty for using metric information, it was much faster than DERVISH, and if it made it into the goal room, victory was ours—but the robot gods giveth, and they taketh away. The final long corridor had only one door before the goal, and the failure to use corridor junctions for registration caught us: ERRATIC mistook a corridor entrance just in front of the goal door for the door itself; so, it didn't quite make it to the goal and first place. However, because none of the other robots besides DERVISH made it past the blocked corridor, ERRATIC was declared second-place winner.

Conclusions

The competitions are useful because of the lessons learned comparing different robot strategies aimed at the same task—indoor navigation. For the navigation task in the 1994 competition, no team had any significant advantages in hardware design: Everyone used sonars for navigation, and the computational and mechanical equipment was comparable (arguably, we had the worst dead reckoning of the group). Thus, the performance reflects the ability of control- and sensor-interpretation algorithms as well as the planning strategy of the teams. Here's some of what I learned:

First, don't depend on dead reckoning, even very good dead reckoning. I knew it before, and this contest confirmed it.

Second, make use of metric information. It is powerful if used in the right way. It lets the robot move quicker and recover from errors sooner.

Third, make sure you can use imprecise metric information. Just as with dead reckoning, it is never good to rely on the accuracy of a map. Good registration with respect to sensed real-world objects is essential. We learned it the hard way in the finals.

Fourth, it helps to have a random component in your rationality, just to keep the judges on their toes. If your robot is predictable, it's much easier for the judges to frustrate it. However, some randomness tends to frustrate the judges.

Let me end by making a shameless plug for ERRATIC. It's a cheap, reliable indoor robot platform; there's lots of good software to back it up; I'm developing course materials based on my classes at Stanford; and I'm looking into getting a robot maker to distribute the hardware. For more information on the robot and pointers to articles on the issues discussed here, check out http://www.ai.sri.com/people/erratic.

Kurt Konolige is a senior computer scientist at the Artificial Intelligence Center of SRI International and a consulting professor at Stanford University. He received his Ph.D. in computer science from Stanford University in 1984; his thesis, "A Deduction Model of Belief and Its Logics," developed a model of belief based on the resource-bounded inferential capabilities of agents. His research interests are broadly based on issues of commonsense reasoning, including introspective reasoning; defeasible reasoning; and reasoning about cognitive state, especially in the context of multiagent systems. More recently, he has conducted research in fuzzy control for reactive systems, constraint-based planning and inference systems, and reasoning about perceptual information.