AI Magazine Volume 14 Number 4 (1993) (© AAAI)

Robot-Building Lab and Contest at the 1993 National AI Conference

Carl Kadie

■ A robot-building lab and contest was held at the Eleventh National Conference on Artificial Intelligence. Teams of three worked day and night for 72 hours to build tabletop autonomous robots of legos, a small microcontroller board, and sensors. The robots then competed head to head in two events. I was one of the developers of JACK, the second-place finisher in the Coffeepot event. This article contains my personal recollections of the lab and contest.

The robot-building lab and contest was held at the Eleventh National Conference on Artificial Intelligence (AAAI-93). The contest was a chance to learn about building machines that operate in the real world. I was one of the developers of JACK, the second-place finisher in the Coffeepot event. This article contains my personal recollections of the lab and contest.

On Monday, the day before the conference proper began, I attended a tutorial on autonomous mobile robots. The tutorial filled a large room. It lasted two hours and was high level. The tutorial mentioned *emergent behavior*, which means the robot behaves in an interesting but unexpected way. Of course, machines in the real world can also behave in uninteresting, unexpected ways, in which case the term *buggy behavior* would be more accurate.

After the tutorial, 22 teams of about 3 persons each went to the robot-building lab. The lab was in a roped-off area of the main exhibition area. When the exhibition area was open to general conference attendees, they could come up to the rope and talk to the teams.

Some people preregistered as teams (both winning teams did). Others, such as me, were assigned to teams. My teammates were Karl Altenburg from North Dakota State University and Jalal Maleki from Linkoping University in Sweden. We had never met before the conference, but we did exchange e-mail when we learned we were on the same team. We made a good team, developing a good working style quickly.

JACK zoomed forward; the crowd liked the speed. JACK stopped; the crowd was worried. JACK turned slightly and then zoomed again, turned, zoomed, and entered the target area. The crowd liked the fast win.

The competition was to be run on Thursday at noon, so we had about 72 hours to build a robot that would operate by itself. If we chose to, we could also use some of the 72 hours to sleep, eat, and go to conference talks. (I made it to Herb Simon's excellent talk and most of the session on computational learning theory.)

Each team was given the use of a laptop computer and a box of parts. The robots were to be made of legos. The box seemed to contain every type of lego building piece in the universe.

We were also given a 6.270 board, a microcontroller board designed at the Massachusetts Institute of Technology for computer science class 6.270. It is based on the Motorola 6811 chip. It has 32K of memory: 16K for the operating system (multitasking C in 16K!) and 16K for user programs. We eventually had much code but had no trouble with memory. The board can control six motor output, has eight digital input, about a dozen analog input, several unswitched output, several light-emitting diode (LED) output, a servo motor, and modulated infrared (IR) input-output. The only ports we came close to running out of were the digital input (however, we could have used analog input instead) and the LED output (however, we used some of the motor output instead).

We were given some prebuilt sensors and the use of a soldering table to make more. The sensors included simple switches, photocells, IR transmitter-receiver pairs (good as proximity sensors), and a modulated IR receiver for receiving a signal from a target beacon (or a signal from the other robot). Sadly, the bend sensors (from the Mattel powerglove) did not arrive, so we could not use them.

We were given three motors: two DC motors and one servo motor that we could make turn, pretty reliably, to any angle from 0 to 180 degrees.

The Events

There were two events. Each was run two robots at a time. The contest was double elimination; so, to be eliminated from an event, a robot had to lose twice.[1]

Both events involved moving from a starting location across the playing

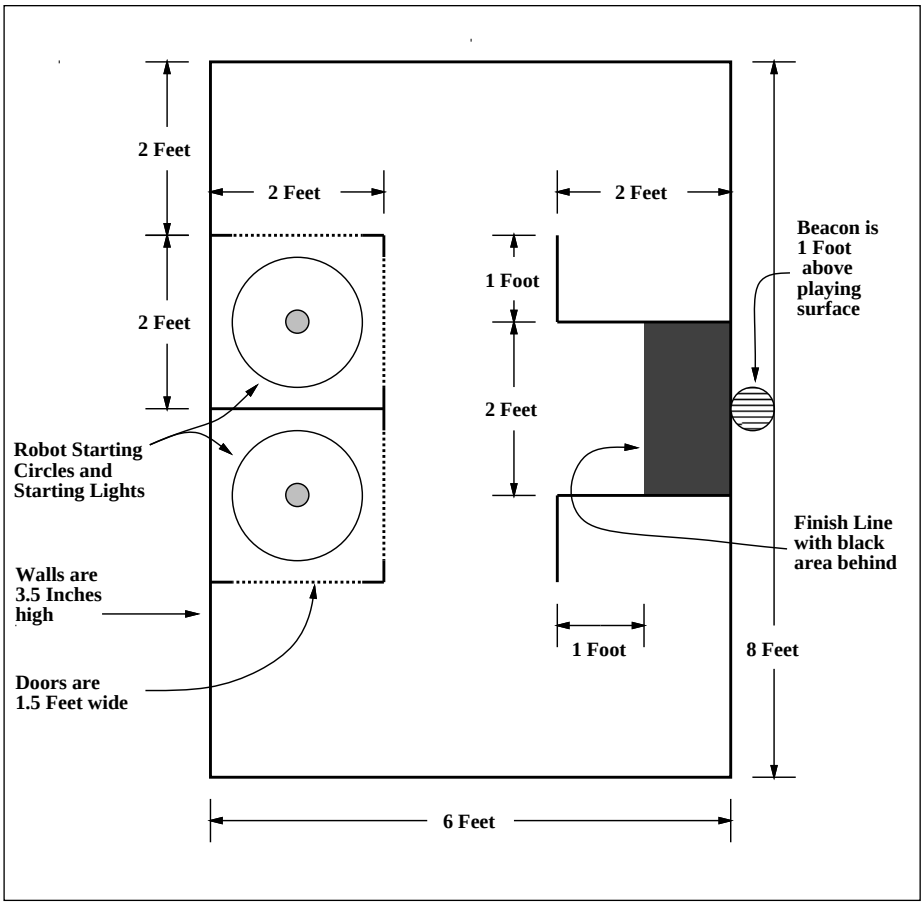

Figure 1. The Playing Field for the Escape from the Office Event.

field to a target beacon that was emitting an identifying, modulated IR signal. The playing fields were 6 feet by 12 feet, with 3-1/2–inch walls around the outside. In both events, the robots started side by side, separated by a 3-1/2–inch wall. A robot could be told if it was on the right or left side by having a tiny dual in-line package switch flipped. Robots were arbitrarily assigned to the left or right side and were started either facing the target, facing each other, facing away from each other, or facing back. The direction was decided with a roll of the dice. The robots were not told which way they were facing.

Figure 1 shows the field for Escape from the Office, the first event. The field and walls in the first event were painted white. The target area was black. The starting points had lights under them. When the lights turned on, the robots were to start; 60 seconds later, they were required to turn themselves off.

There were two movable wall-doors (one for each start location) covered with shiny metal. They could either be installed between each robot and the target beacon (meaning that the robot would have to go out the side) or installed on the side (meaning that the robot would have a straight path to the target area). Their installation location was determined by a roll of the dice.

The winner was the first robot to reach the target area in less than 60 seconds or the closest robot if neither reached it in 60 seconds.

The second event was called Coffeepot (figure 2). It involved object finding and retrieval. The scoring was as follows: 1 point for being the first to touch the coffeepot, 1 point for being the last to initiate a touch to the coffeepot, and 1 point for moving the coffeepot into the coffee-cup picture. The winner was the robot with more points. If both robots tied at 0, both lost. If they tied at 1, both won.

Design of JACK

We named our robot JACK, an abbreviation made from our initials. Our goal was speed. Most of the competitors copied a prototype lego robot. It had good power but poor speed. We (especially Altenburg) thought a faster robot would give us a competitive advantage. In the end, we had a robot that was probably the first or the second fastest.

The prototype robot had two drive wheels and one caster wheel. In contrast, JACK had four drive wheels. The left wheels were powered by one motor, the right wheels by another. JACK moved and turned like a tank, and like a tank, it scuffed (slid sideways on its tires) when turning, making turns to a preset angle difficult.

To sense the target beacon, we had one modulated IR sensor facing the front of the robot. It had side blinders on it so that it would only see straight ahead. We also had the same type of sensors mounted facing left and right. These sensors had no blinders because we wanted the robot to have peripheral vision, telling it which way to turn to face the target beacon.

Many of the other robots mounted a similar vision system on the servo motor, giving them an eye that moved independently of the robot's body. Given our development time constraints and the speed at which JACK could spin, we decided it was easier just to spin the whole body.

I did most of JACK's programming, with my teammates figuring out what high-level strategy the robot should follow.[2] The programming environment for the 6.270 board is wonderful. It uses IC (interactive C), a multitasking C. The code is written on a relativity big computer such as a MACINTOSH POWERBOOK or a Microsoft WINDOWS laptop. The big computer compiles the code into p-code that is downloaded to the 6.270 board. The 16K operating system on the board interprets the code quickly, even while multitasking.

The first code we wrote defined low-level functions for controlling actions with the two motors (forward, backward, spin). From the beginning, all the code was symmetric. A program written for the robot sitting on

the left side of the field also worked for the robot sitting on the right side of the field. I achieved symmetry by having a series of constants that gave a motor or sensor's true port number and a series of variables that gave its virtual location.

Deciding how to code high-level functions was more difficult. After several false starts, we ended up with what we called abstract sensors and behaviors. This scheme was not perfect, but given the few days in which we had to work, I think it was pretty good.

We created two abstract sensors. The first sensor told if the robot was bumping into anything in front (the only direction we ended up caring about). It was accessed with a global variable. The global variable was set by a background process that looked at seven real sensors (three mechanical switches and four IR proximity sensors). Each IR proximity sensor was made up of an IR LED and an IR receiver. The background process worked by turning LEDs off, waiting a moment, taking a reading from the receivers, turning LEDs on, waiting a moment, and then taking another reading. It then compared the difference between the before and the after readings. If the difference was over a threshold or if one of the mechanical switches was closed, then the global variable was set to true.

The other abstract sensor told if the target beacon was left, right, ahead, or behind. It was a background process. It compared the number of consecutive times the target beacon was seen from each of the three modulated IR sensors.

For behavior, we tried to get wall following to work, but without the bend sensors, we couldn't. I wasted the better part of a day working on wall following before my teammates told me to give it up.

We were more successful with *lockin,* a behavior in which the robot would spin for as long as two seconds, trying to get the target beacon in front of it. The time limit was important; in the second event, the robot could be too far from the target beacon to see it. Some of the other robots spent their whole turn spinning and

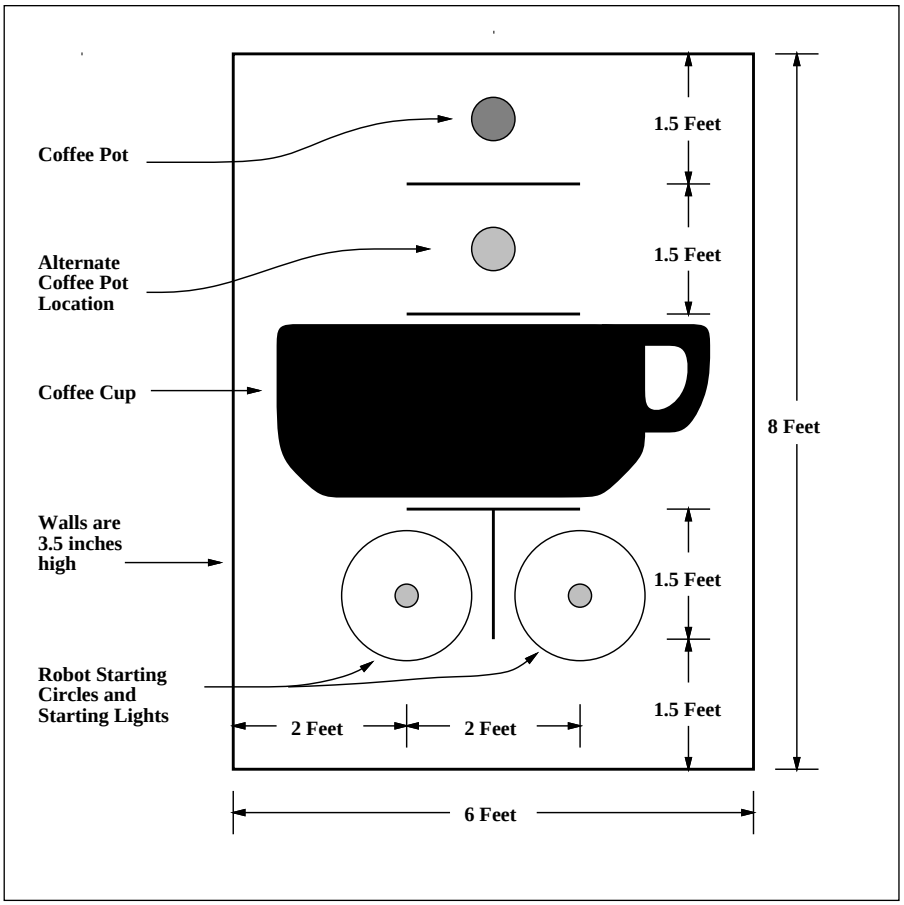

Figure 2. The Playing Field for the Coffeepot Event.

spinning in a vain attempt to detect the target beacon.

The other interesting behavior was *track*. I thought we could move toward the target beacon while the robot corrected its course with forward turns. It ended up that our robot could only turn reliably by spinning (having the left and right wheels going in opposite directions). Thus, *track* came to mean go forward until (1) a bump was met, (2) some time limit expired, or (3) the target beacon was no longer ahead.

By combining *lockin* and *track* in a loop, JACK was able to find and move to the target beacon if the target was not blocked by a wall or the other robot. To help deal with blockage, we had JACK back up and turn a bit whenever a track behavior was stopped by a bump.

Our team, along with many of the other teams, stayed up the entire night before the contest. At about 6 AM (the contest started at noon), we had the lockin and track steps working well. It was fun to play bullfight with JACK by moving the coffeepot away from it just as it was about to reach the pot and then watch it spin around and go for the pot again.

For the Escape from the Office event, we decided to have JACK follow a fixed pattern (mostly based on timed moves) and then go into a lockin-and-track loop. For the Coffeepot event, because of the lack of time, we decided to run the same program. We were still tuning the fixed pattern about 30 minutes before the contest started.

It was at this point that we knew we had to put in the organizer-supplied code for turning on the robot with the start light (perceived with a photocell) and turning it off in 60 seconds (easily done with a background process). We installed the code but found to our horror that the

PPCP94 Call For Papers

Second Workshop on Principles and Practice of Constraint Programming

May 2-4, 1994 Orcas Island, Washington, USA

The workshop will be held at Rosario Resort on Orcas Island. Orcas Island is one of the San Juan Islands, in northwestern Washington State, and is a place of great natural beauty. Rosario can be reached via various modes of transportation, including seaplane from Seattle direct to the Rosario dock.

Authors are invited to submit (by hardcopy or e-mail postscript file) five copies of a short paper, not exceeding 2000 words, by Wednesday January 12, 1994 to the program chair:

Alan Borning
Department of Computer Science & Engineering, FR-35
University of Washington Seattle, Washington 98915 USA
(for express mail, add the building name: Sieg Hall Room 114)
e-mail: borning@cs.washington.edu
tel: +1-206-543-6678 fax: +1-206-543-2969

IMPORTANT DATES
Deadline for submission: January 12, 1994
Notification of acceptance or rejection: February 25, 1994
Final paper due: March 25, 1994
Workshop dates: May 2-4, 1994

Proceedings will be available in technical report form at the workshop and, including feedback from the workshop, will be published in book format.

liquid crystal display on the 6.270 board started dimming. I thought it was probably a bug in the supplied code, but Altenburg suspected that he had wired in the photocell wrong. With 15 minutes to go, he went over to the soldering table and made and installed another sensor. It worked fine.

While waiting to compete, Altenburg suggested that for the Coffeepot event, we do more than run the Escape from the Office program. He suggested that we modify it so that every so often, the robot locked in and tracked on the left or right. This modification would mean sometimes moving sideways to the target beacon rather than always straight for it. I coded this change on the laptop computer but couldn't even compile it because the robot and its board were competing in the first event. After the first round of Escape from the Office, we ran with the robot to our worktable and downloaded. We then ran back with the robot to the first round of the Coffeepot event.

The Contest

Here, I describe the competition, detailing the individual rounds for each of the two events.

Escape from the Office, Round 1

On the playing field, the front door on the office was open—the easiest situation for us. JACK zoomed forward; the crowd liked the speed. JACK stopped; the crowd was worried. JACK turned slightly and then zoomed again, turned, zoomed, and entered the target area. The crowd liked the fast win. The other robot had not moved far from its starting position.

The target area was marked in black, and we had a white-black floor sensor in the robot. However, we didn't see the point in stopping JACK just because it thought it was in the target area; so, JACK kept hitting the back wall of the target area, backing up and turning a bit, then turning back to the target beacon, and slamming into the wall again. The crowd thought its performance was funny.

Coffeepot, Round 1

We had no idea what JACK would do. We were hoping for emergent behavior but feared buggy behavior. Here is what it did do. It zoomed, hit, locked, zoomed, hit, and so on. Sometimes, it even seemed stuck behind walls, but with its bump, backup, and track sideways behaviors, it always got out. It reached the coffeepot first (1 point). Then, it backed up, and it hit again and again and again. Backing up and hitting again seemed the best way to get the second point (for last to initiate contact).

The other robot eventually made it to the coffeepot. In the confusion that followed, JACK somehow got between the coffeepot and the other robot. Both robots were stuck in a corner when the 60 seconds were up, so we got the second point.

We were now 1 to 0 after one round in each event. We had visions of a double win.

Escape from the Office, Round 2

The open office door was on the side—the difficult configuration. I was so nervous that I wasn't even in the arena; I was 30 feet away, behind the crowd, watching on a television monitor. A teammate started JACK but had forgotten to flip the switch to tell JACK which side of the field it was on. JACK executed the wrong fixed pattern and didn't even make it out of the start area.

We were 1 to 1 in Escape from the Office.

Escape from the Office, Round 3

We all double-checked everything this time (even if we were nervous). Something still went wrong during the execution of the fixed pattern, and JACK got stuck in a corner and could not get out.

We were 1 to 2 in Escape from the Office, so we were eliminated.

The eventual winner of Escape from the Office was DEATH STAR 2000, built by a team from AT&T. This team had a secret weapon: It found that DEATH STAR 2000 could detect the shiny metal of the closed door from the start position (with a red LED detector). The robot could also make precise turns using its servo motor to turn the beacon detector at any angle. The robot moved at the same speed as most of the others (that is, rather slow), but it never made a mistake.

Our morale and hopes were low at this point.

Coffeepot, Round 2

Using its tracking, JACK found the coffeepot first (1 point). Several times the crowd thought that JACK was stuck, but sideways tracking always got it out. After it found the coffeepot, it kept backing up and hitting the coffeepot. The other robot wasn't close, so JACK had a good chance of getting the second point.

JACK's design didn't give it much mechanical power, but with speed and a running start, it had good momentum. Every time it hit the coffeepot, the pot moved. Miraculously and accidentally, JACK hit the coffeepot into the painted area on the playing field (3 points). We were happy. I think most of the crowd thought JACK hit the pot into the painted area on purpose. It did not. It had a happy accident.

Coffeepot, Round 3

The other robot got to the coffeepot first (1 point). JACK eventually got there and started its tapping. Unfortunately, the other robot was also tapping and got the last touch (2 points). This loss was JACK's first in the Coffeepot event.

Coffeepot, Round 4

Because of double losses in this event for many of the other robots (neither robot getting to the coffeepot), only three robots were left, each with one loss. It was decided that these robots would compete round robin; the robot with the most points would win.

NOT YET and JACK both went against the third robot, and both won 2 to 0.

Coffeepot, Finals Rounds

Now it was NOT YET versus JACK.

NOT YET was created by Thomas Pendleton, Peter Bonasso, Linda Williams, and Robert Ambrose of The MITRE Corporation. NOT YET was good at the Coffeepot event. It would lock in on the target beacon, carefully turn 45 degrees (by time), move until it hit the side wall, and follow the wall (using two wheels on the wall and a distance sensor) until it detected the coffeepot on the side. It would then turn toward the pot and grab the pot with a wonderful two-fingered magnetic hand. Finally, it would reverse and try to drag the pot to the painting of the cup. Its only loss was in the first round when it tried to follow the wall without the side distance sensor; it hit the wall at too steep an angle and got stuck.

Just as I thought it would, NOT YET got to the pot first because it went more directly (1 point). JACK got there before too long and started tapping. Because NOT YET never let go, JACK got the point for being the last to start touching. Time ran out. It was a tie.

The next round was just the same except that at the coffeepot, the robots were positioned so that every time JACK pulled back to tap, NOT YET pulled the coffeepot forward a bit toward the painted cup. With time running out, NOT YET narrowly missed the painting, but finally, it got the pot to the painting; NOT YET won a second point. First place went to NOT YET. JACK was the runner-up.

I believe that if NOT YET and JACK had competed 100 times, NOT YET would always get the first point and JACK the second point. However, JACK could never expect to get the third point. Thus, it was almost inevitable that NOT YET would have a round in which it would get the pot to the painted area and win; so, I am satisfied with the results.

Conclusion

After the contest at about 6 PM, I got back to the dormitory tired. Over 4 days, I had gotten a total of 13 hours sleep. At the dormitory, I found a two-day-old message to call my wife Nanci. When I couldn't reach her, I decided to set the alarm, sleep for an hour, and then call again. I fell asleep easily. When I woke up, it was dark, and the clock said 9:40 PM. I figured that the alarm must not have gone off. I called home and talked for about 30 minutes. As we were saying good-bye, I said, "See you tomorrow." Nanci said, "Don't you mean later today?" I said, "Isn't it about 9 o'clock in Illinois?" She said, "Carl, it's 3:30 in the morning here." I had apparently left my alarm clock in set-time mode rather than alarm mode; it apparently keeps time more slowly in set-time mode. I fell back asleep trying to decide if the clock's behavior was emergent or just buggy.

Acknowledgments

The contest was stressful but wonderful. Thanks to the organizers—Lynn Andrea Stein and David Miller—and to the staff—Matt Domsch, Karsten Ulland, Carol Lee, Mike Wessler, and Anne Wright. Thanks to my teammates, Jalal Maleki and Karl Altenburg.

Notes

1. The manual and rules for the events can be obtained by anonymous ftp from aeneas.mit.edu:/pub/ACS/6.270/AAAI.

2. The code can be obtained by anonymous ftp from a.cs.uiuc.edu:pub/JACK.

Carl Kadie is currently finishing his Ph.D. at the University of Illinois, where he is supported by a Hertz fellowship. His research interests include machine learning, statistics, online media, and robotics.