GEMINI: First of a New Breed

FEBRUARY 1985 VOL. 7 NO. 2

$3.00 USA/$3.50 CANADA

THE JOURNAL OF INTELLIGENT MACHINES

ROBOTICS AGE™

FAREWELL TO STEP COUNTERS

BY RAYMOND GA COTE

This month's cover story (see p. 12) has a picture of the Gemini robot from Arctec, the first of a new generation of robots. What makes this new generation different? Granted it has more processing power than previous commercially available robots (each subfunction has its own processor) and it also has more sensory information available than anything else yet to arrive. However, this is not what rates the next generation label.

Gemini is the first commercially available machine to actually move from place to place and go through doors without counting steps. Up to now, telling a robot to move from one room to another required the robot to remember how many steps it was between the two rooms. This could drive the control software absolutely crazy when it found an object blocking its path.

At the recent personal robotics session at the Robots-West show, Gene Oldfield spoke about *goal-driven* robot systems. His belief is that robot programmers should spend less time worrying about *avoiding objects* and spend more time trying to *attain goals*. Gene's ideal navigational goal is a doorway. He observed that people moving about a house always aim at doors. A room without visible doors is very disturbing and can produce an uneasy feeling in many people—a feeling of being trapped. Thus, Gene theorizes, a robot should be designed to seek out doors and move through them just like a person.

This is what the Gemini designers have developed, a door-seeking robot. The current design requires an active infrared room beacon in each room and a passive infrared reflector at each door. These are used to present convenient room markers. People also work with room markers. Anyone who has walked through a hall of mirrors at an amusement park knows the disorientation that occurs when all the rooms look identical and the doors are difficult to find. At this stage of technology, a robot operating without some sort of beacons and door markers is akin to a person wending their way through a hall of mirrors.

Yes, someday the robots will be sophisticated enough to determine what room they are in just by looking around. Until then, we've at least seen the next step in robot evolution where they don't have to keep track of every step they have made. I'm glad the days of the step-counting robots are numbered.

Circle 21

PHOTO ESSAY: A FIRST GLIMPSE AT GEMINI

BY CARL HELMERS

We recently had the opportunity to visit Jack Lewis and his crew in Columbia, Maryland to view the first of a new line of autonomous robots. Given the name "Gemini," this class of wheeled canister robots from Arctec Systems, Inc. is a significant step forward in the practical design of autonomous vehicles for research, practical personal use and experimentation. The Gemini-class machines will be of great interest to those who want to learn about and experiment with the use of today's concept of the self-feeding, self-powered autonomous platform robot.

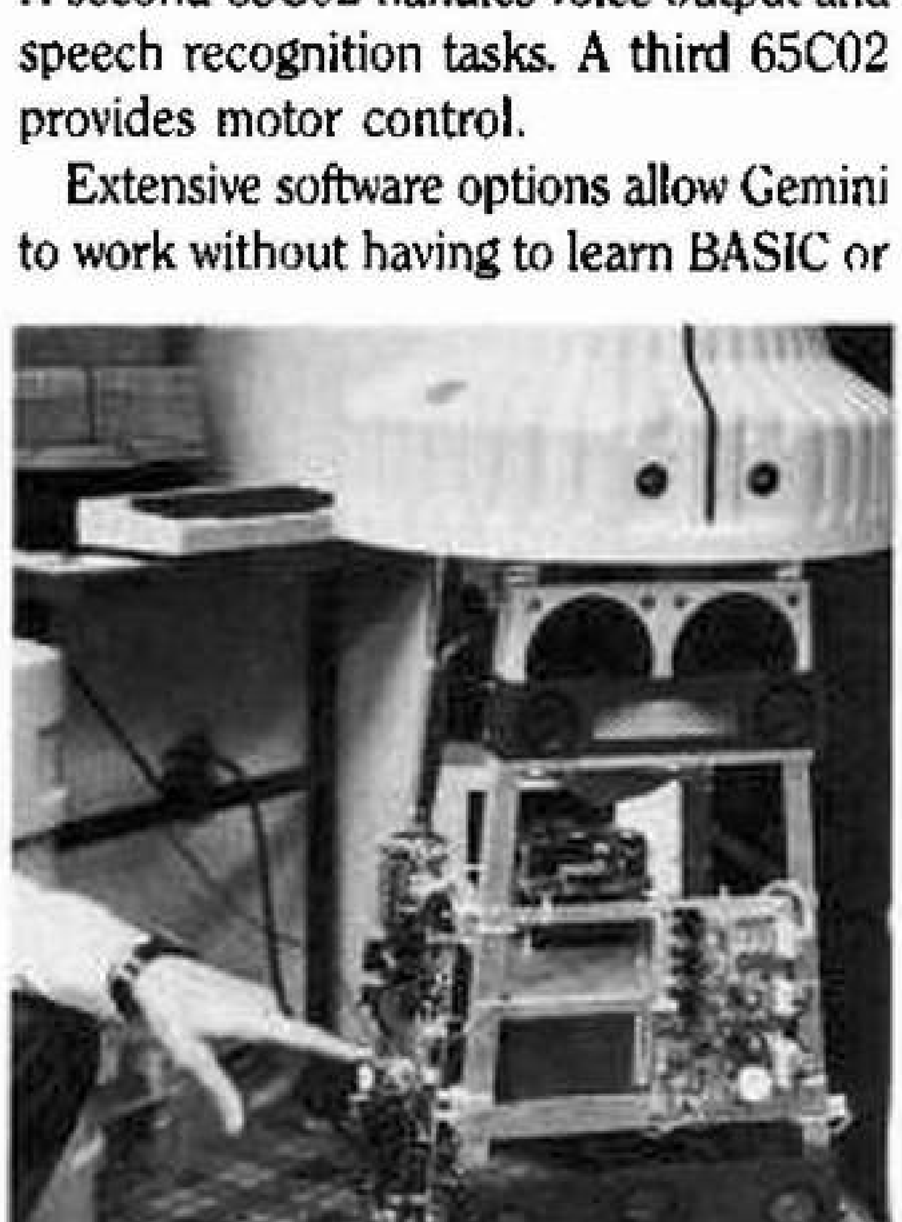
We observed Gemini in operation in both prototype and final forms, exhibiting complicated and reliable motion behaviors, including transit from one room to another and return to the starting point. This robot uses principles of free vehicle navigation derived from marine and aviation practice rather than attempts to memorize an open-loop trajectory on uncontrolled floor surfaces. Using a beacon system together with door markers, Gemini can measure positions, triangulate and determine an attempted path to reach a selected goal. Bump, collision and object-presence sensing allows it to seek paths around obstacles using feedback rather than a memorized trajectory. Gemini may be the first practical self-feeding robot as it returns to a battery charger station always placed under a specific room marker beacon.

An optical system is used to sense beacons. An active reflective infrared optical system is used to locate door markers. When given the task of going from room N to room M, Gemini uses a stored connectivity map to count door markers relative to room N's beacon, then positions itself approximately near the selected door, sidles up to the vicinity of the door to fix the precise physical opening with its sonar, does a final shuffle to make sure it is in the right place, and then cruises through the doorway. This maneuver—all with active sensors—takes several minutes to accomplish. The time is used mainly to accomplish mechanical scanning by the rotating turret sensor. (Clearly, an all-solid-state sensing approach would improve the speed by eliminating the mechanical scanning requirement. According to Jack, the CMOS version of a 6502 used in the navigation problem is using only a few percent of its computational bandwidth. In radar system engineering, the most advanced radars use solid-state scanning methods for exactly that reason—at a much higher cost.)

The system concept for Gemini is best illustrated by photographs. Photo 1a shows the pre-production prototype of Gemini with its clothes on. Photo 1b shows Gemini with its clothes off, with Jack pointing to one of the electronics boards. The machine's computational design uses a multiprocessor, all CMOS approach to conserve battery power. There are only a limited number of watt-hours that can be stored in the three batteries that Gemini uses. One 65C02 computer is the main computer orchestrating all operations. A backplane bus similar to the Apple II bus allows one to plug in extra peripherals. (The "similar" nature of the bus means that care must be taken to pay attention to power requirements and interaction with a slightly different address-space allocation.) This computer interacts via an 8-line by 40-character LCD display and keyboard, uses a mass storage peripheral (3¼ in. floppy or endless-loop "wafer" tape) and generally serves as a BASIC oriented personal computer with voice I/O capability. A second 65C02 handles voice output and speech recognition tasks. A third 65C02 provides motor control.

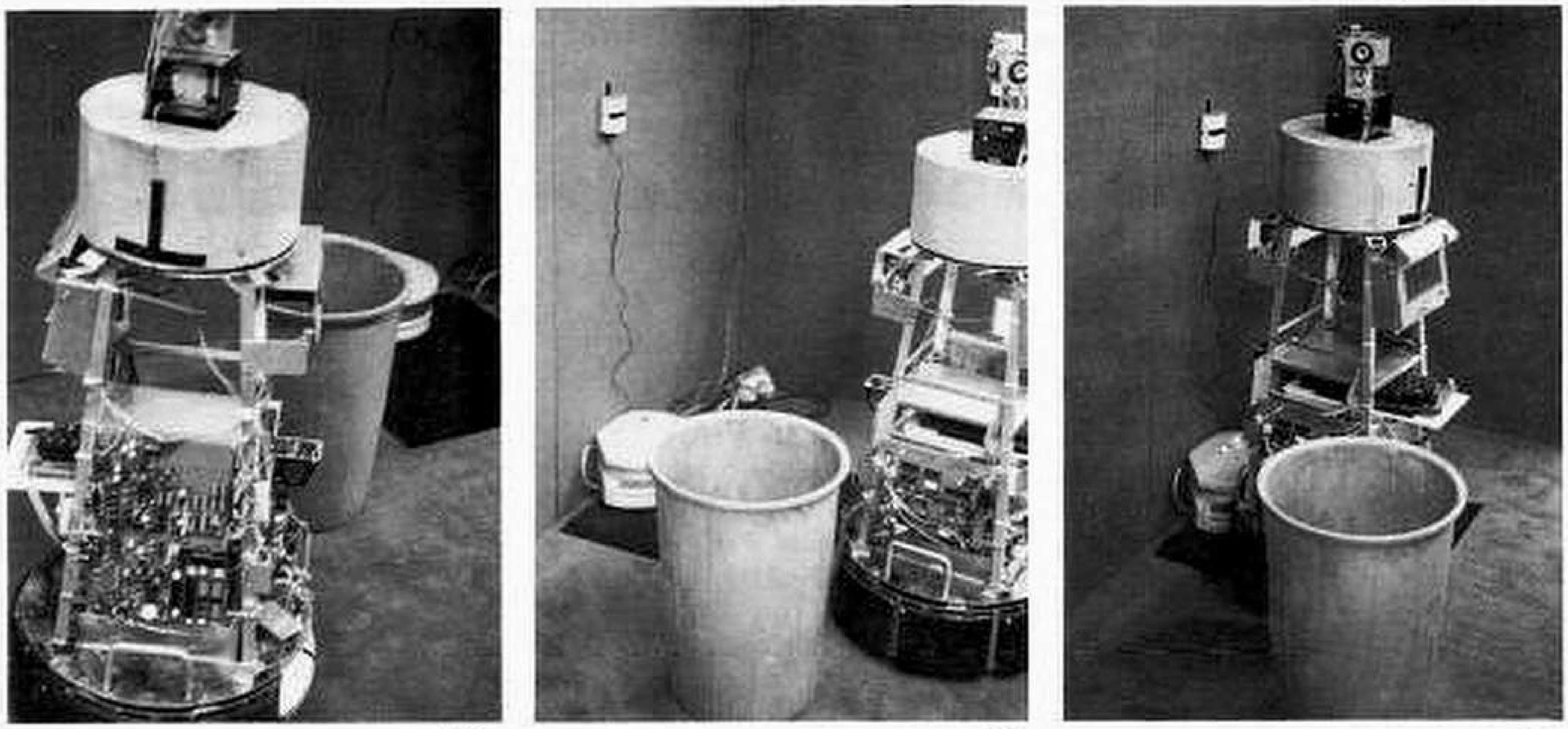
Extensive software options allow Gemini to work without having to learn BASIC or

1a



1b

Photo 1: Two views of the Gemini autonomous robot, with clothes on (a) and naked (b). In 1a the robot is shown nestled up against a charging station located on a black anti-static pad below a wall-mounted room identification beacon. In 1b we see Gemini's innards. Note the convex battery charger connection at the bottom of the robot, designed to mate with the corresponding pair of plates on the charging station partially seen in 1a.



2a 2b 2c

Photo 2: A sequence of photos taken while an earlier development model of Gemini seeks its charging station in a demonstration room. We start at (a) where the robot sees the charger station wall beacon but senses an obstacle in the way of a direct path. In our demonstration, the robot tested and probed for a while. After a couple of false starts, it finally moved to a position where it found a direct path, as shown in (b). The end result of the demonstration was to move forward into its charger, activating the charging process as signified by the two lights turned on.

assembly language programming. This software includes a Scheduler package to orchestrate periodic and random behaviors of the beast using its real-time clock. This is probably the single most practical application feature of Gemini—a robot that remembers and reminds. But for those of us with programmming experimentation proclivities, Gemini adds a new dimension to the possible enjoyment of this craft: real time and motion control.

Motion control and navigation software of Gemini allowed us to watch a naked prototype seek out its charging station in the presence of a wastebasket obstacle. The station (Photos 2a to 2c) is the object on the floor of the demonstration room under the "Room #1" beacon transmitter which we see mounted on the wall.) In Photo 2a, Gemini can see the beacon and begins approaching it to sip the electric juice of its life. As it cruises forward Gemini senses the wastebasket obstacle. This causes it to ponder about alternative methods of reaching its beacon goal. As we see in Photo 2b, a decision has been made to go around to the right, where a direct path to the beacon and charger is possible. Executing the direct path, the proto-Gemini is seen happily imbibing at its electronic food trough in Photo 2c. The behavior is incredible to watch.

Our observations of Gemini were presented in the environment of the research laboratories in which it was designed. Based on what we observed of Gemini's substance, engineering and behavior, we can hardly wait to get our hands on a production version. Then we'll do the ultimate test—is Gemini's behavior reproducible in the real world away from its birthplace? More on this later. ■

GEMINI SPECIFICATIONS

Each of Gemini's main functions is controlled by a separate Rockwell 65C02 CMOS microprocessor. The peripheral computers communicate with the central processor over serial communications lines. The central processor can save power by determining when the peripheral computers are turned on and off.

The central computer has 64 Kbytes of read-only memory and 56 Kbytes of programmable memory plus communications capabilities to all other processors.

The voice input/output and sound computer has 24 Kbytes of read-only memory and 16 Kbytes of programmable memory. The voice and sound generators are separated so Gemini can speak and make music at the same time. This board also handles the voice recognition routines for storing commands from up to three users.

The propulsion control computer controls all movements and coordinates the ultrasonic position sensing arrays. An optional fourth, remote computer is available that allows Gemini to control BSR controllers, modems, and other external devices over a radio control line.

Standard sensors include light, sound, temperature and battery voltage. Sensors for barometric pressure and smoke are optional. Gemini also comes with nine ultrasonic collison avoidance and navigation sensors that help it move from room to room. Other standard hardware includes a real-time clock, 16 channel, eight-bit CMOS analog to digital converter, an RS-232 serial communications port, a Centronics printer port, a hardware random number generator, a joystick controller, and a charger. Gemini's hardware can be increased through the use of the four peripheral expansion ports based on the Apple II/II+ expansion bus.

Gemini is equipped with its own cordless, detachable keyboard and a 40-character by 8-line alphanumeric liquid crystal display.

The software includes a 90 Kbyte "artificial intelligence" operating system, automatic self-checking and diagnostics on power up, extensive demo programs, a task scheduler program, and a voice synthesizer with text-to-speech software.

Optional software includes a three-user, 256-word voice recognition system with adaptive learning ability, an integrated voice command language (VOCOL), a user-configurable security guard program, and a floating-point BASIC with robot control commands.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

71	81	91
Excellent	Good	Fair

THE SIMULATION

The robot we are simulating is very simple. It has only one bumper and can detect when it hits a wall but cannot tell the direction of incidence. One of the first programs someone writes for a robot is to have it run around on the floor, bumping into things. We can have our robot do the same by drawing a room on the screen, placing our robot into the room, and letting it go. When it bumps into a wall, we can back it up and head it off in a new direction. This is done in Listing 1.

```
LISTING 1: LOGO LISTING OF SIM1

TO SQUARE :LEN
REPEAT 4 [ FORWARD :LEN RIGHT 90 ]
END

TO DRAW.ROOM
PENUP
SETPOS [ -100 -100 ]
PENDOWN
SQUARE 200
PENUP
SETPOS [ 0 0 ]
END

TO RECOVER
REPEAT 10 [ BACK 1 ]
SETH ( 22.5 * RANDOM 16 )
END

TO SIM1
CS
FS
DRAW.ROOM
WHEN 0 [ RECOVER ]
SETSP 20
END
```

ROUTINE SQUARE. This simply draws a square with each side :LEN units long.

ROUTINE DRAW.ROOM. This sets up our room. We first tell the turtle to lift up its pen so it does not make too many lines. We place it in a lower left corner of the screen (SETPOS [−100 −100]), put its pen down, and tell it to draw a square 200 units on each side. This is a rather simple room, but we could draw a more complicated one if we wished to. We raise the pen again and place the turtle in the center of the room.

ROUTINE RECOVER. This routine is called whenever the turtle hits a wall. We tell it first to back up ten steps, and then to select a new direction. In this routine we use the RANDOM 16 instruction, which will return a value between 0 and 15. We take this number and multiply it by 22.5 so the turtle will turn between 0 and 337.5 degrees. When the routine is finished, the turtle heads off in a new direction. This is the most powerful routine, the one in which we can do the most work. The way we recover from collisions and select a new direction begins to enter the realm of artificial intelligence.

ROUTINE SIM1. This is the main routine, with which we tell the turtle to clear the screen (CS), use the full screen (FS), and draw the room. SIM1 sets up the demons (WHEN 0 [RECOVER]), tells the turtle to begin traveling, and sets its speed (SETSP 20).

As the simulation runs, we might see the turtle bang its head several times before choosing a direction that gets it away from the wall. This is a function of the RECOVER routine and we can, if we wish, begin developing a more intelligent recovery routine for when the turtle hits a wall.

THE SMARTER MACHINE

Perhaps the easiest recovery routine to implement is the LAST-RIGHT method. This is accomplished as follows: When we hit a wall we back up. We know the direction we are traveling, so we look up in a list the *response we used last time to get away from the wall. If the number is less than 0, we have never before encountered a wall while traveling in this direction. So we pick a random direction and travel until we hit something. We then see how far we traveled. If we traveled more than, say, 20 units, then it is a valid direction and we place that value in the list, so we can "remember it" if we hit a wall while we are traveling in that direction again. In time, the turtle will get a good response table built up so that when it hits something it can get away without running into many walls.

I ran statistics on this method, comparing it to that of random choice. With random choice, the turtle picked the proper direction to get away from the walls about 50 percent of the time. With the LAST-RIGHT method, accuracy increased to about 70 percent. With the LAST-RIGHT method, the turtle spends more time exploring and less time banging into walls.

To implement the LAST-RIGHT method, we need to give the robot a way to measure distance. On mobile robot platforms this can be done by a number of methods. With a LOGO simulation we can find out the turtle's location by using the POS command, which returns a list containing the current position. We save this