AI Magazine Volume 18 Number 1 (1997) (© AAAI)

To Know or Not to Know

On the Utility of Models in Mobile Robotics

Sebastian Thrun

■ This article describes JEEVES, one of the winning entries in the 1996 Annual AAAI Mobile Robot Competition and Exhibition, held as part of the Thirteenth National Conference on Artificial Intelligence. JEEVES tied for first place in the finals of the competition after it won both preliminary trials. A key aspect in JEEVES's software design was the ability to acquire a model of the environment. The model, a geometric map constructed from sensory data gathered while the robot performed its task, enabled JEEVES to sweep the arena efficiently. It facilitated the retrieval of balls and their delivery at the gate, and it helped to avoid unintended collisions with obstacles. This article argues that JEEVES's success depended crucially on the existence of the model. It also argues that models are generally useful in mobile robotics—even in tasks as simple as the one faced in this competition.

JEEVES was the Carnegie Mellon University entry in the Clean Up a Tennis Court event at the 1996 Annual AAAI Mobile Robot Competition and Exhibition, held as part of the Thirteenth National Conference on Artificial Intelligence (AAAI-96). Robots competing in this event were given 15 minutes to collect 10 randomly scattered tennis balls, along with two self-propelled squiggle balls, in an arena 9 by 5 meters (m) and deliver them to a custom-built gate.

Figure 1 shows a picture of JEEVES. JEEVES was originally built as a service robot with an eye toward commercialization, independent of the competition (c.f. figure 4). The lightweight robot is equipped with a large rotating brush, capable of capturing as many as eight balls at a time. Apart from the brush mechanisms, however, JEEVES's hardware was not much better than that of most of its competitors. In fact, JEEVES's visual range was extremely limited, and its on-board controller imposed severe limitations on the maximum command rate.

What made JEEVES as successful as it was? A key aspect of JEEVES's success was that its software integrated reactive and model-based control. While JEEVES performed its task, it gradually constructed a geometric map of its environment, which modeled the following aspects: (1) the location of the walls, (2) the location of the gate, (3) the location of the tennis balls, (4) the location and motion direction of the squiggle balls, (5) its own location with respect to the model, (6) its previous location, and (7) the parts of the arena that were unexplored.

Armed with the model, it was simple to determine a suitable search pattern. It was also straightforward to determine appropriate pickup strategies, the location of the gate, and the appropriate time for moving there to unload the balls. The model was constructed on the fly based on sensory data and did not require any additional time or maneuvers. Not only did my team find the model-based approach to control to be extremely robust, it also found it easy to program. Undoubtedly, the existence of a model played the key role in JEEVES's success.

This article outlines the major ideas in the JEEVES software. It also argues more generally for the utility of models in mobile robotics. Using JEEVES and the AAAI-96 mobile robot competition as an example, it discusses the role of models in scalable mobile robot architectures.

Hardware

JEEVES's hardware was designed and built by German design student Hans Nopper, in col-

Figure 1. JEEVES, an Entry from Carnegie Mellon University.

A large rotating brush lifts tennis balls into a ramp inside the robot's shell. To unload balls, JEEVES can reverse the direction of the brush. JEEVES moves approximately 60 centimeters/second. When equipped with a basket, JEEVES can hold approximately 100 balls—sufficiently many for a real tennis court.

laboration with Real World Interfaces, Inc., a leading mobile robot manufacturer. The robot moves at an approximate maximum speed of 60 centimeters/second (cm/sec). It is equipped with seven ultrasonic proximity sensors (only five were used in the competition), a wide-angle color camera, and a high-speed color-based vision system manufactured by Newton Research Labs.

Prior to the competition, the vision system was trained to recognize yellow tennis balls, pink squiggle balls, and cyan markers that marked the gate. The vision system proved extremely reliable during the competition, benefiting from clear color cues provided by the objects. However, the visual range of the camera was below 1.2 m, making JEEVES one of the most myopic robots on the stage. To pick up balls, JEEVES used a rotating brush capable of lifting balls into the interior of the robot. The gate to which balls had to be delivered consisted of a small ramp (shielded by a curtain) just high enough to keep the squiggle balls inside. To unload balls, JEEVES reversed the direction of its brush.

JEEVES's control software was run off board on a remote Sun SPARC 5, with which JEEVES communicated through a 9600-baud radio link. The major computational load was the result of the graphic control interface; the remaining load was well below 20 percent of the available computational resources. The remote software received status updates from the robot and its sensors at a frequency of 10 hertz (Hz). Unfortunately, the maximum command rate for changing the motion direction was only 2 Hz (or less) because of limitations of the on-board controller. When designing the software, special attention was also paid to the fact that the radio link was unreliable. Often, the remote computer did not receive status reports for durations of several seconds, and motion commands were frequently lost and had to be issued multiple times. As a result, the team avoided open-loop control wherever possible.

JEEVES's most significant hardware advantage was its brush, which proved surprisingly capable and robust in picking up balls. JEEVES's most crucial handicaps were (1) the speed at which its on-board motion controller was willing to accept commands and (2) the extremely limited visual range. Because of these limitations, chasing squiggle balls around was not even an option.

Models

JEEVES's success cannot be attributed to hardware alone. Various other teams used robots that were capable of capturing multiple balls at a time, and some of them had a much more responsive hardware. Instead, an important factor in JEEVES's success was its software, which strongly relied on the geometric model (map) of the environment that was built on the fly.

This section describes the software components involved in controlling JEEVES. With data recorded at the competition finals as an example, figure 2 highlights the major components of JEEVES.

Filtering Sonar Measurements

JEEVES's sonars exhibited the typical characteristics of sonar sensors: They seldomly measured the distance of the nearest object within their main cone; instead, they often returned values that were significantly smaller or larger than the correct proximity.

Contrary to a popular myth, sonar sensors are not particularly noisy. They just do not measure proximity. Instead, they measure the time elapsed between emitting and receiving a focused sound impulse. For smooth objects, the chances of receiving a sonar echo depend on the angle between the main sonar cone and the reflecting object. Sound waves that hit a wall frontally are likely to be reflected back in the direction of the sensor, whereas sound that hits a wall in a steep angle is likely to be reflected away, to a direction where it

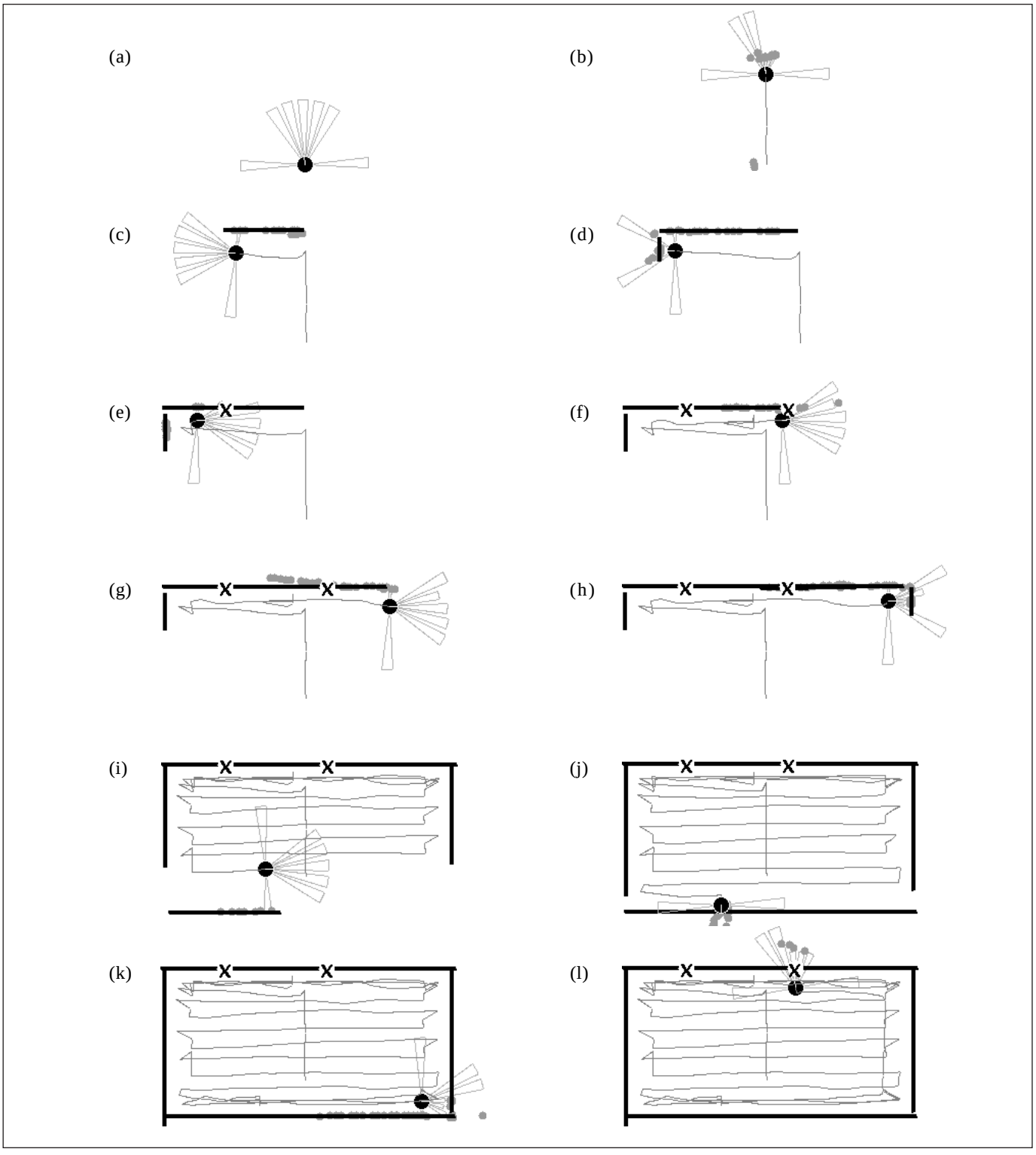

Figure 2. This Figure Summarizes JEEVES's Performance in the Competition Finals.

A. The robot starts without a model. B. It approaches the first wall, as indicated by the projected sonars' measurements. C. After turning left, JEEVES detects the first wall. D. While extending the first wall, JEEVES finds a second one. E. JEEVES turns and falsely believes to observe a gate marker. F. After successfully capturing a tennis ball (see path), JEEVES observes the correct gate marker. G. The software continuously corrects errors in JEEVES's odometry, here noticeable as the angular deviation between the wall and the sonar measurements. H. The third wall is found. I. The fourth wall is found. J. JEEVES picks up a second ball next to a wall. K. The sweep is complete, and JEEVES returns to the correct gate. L. All balls, including the squiggle balls, are shuffled into the gate.

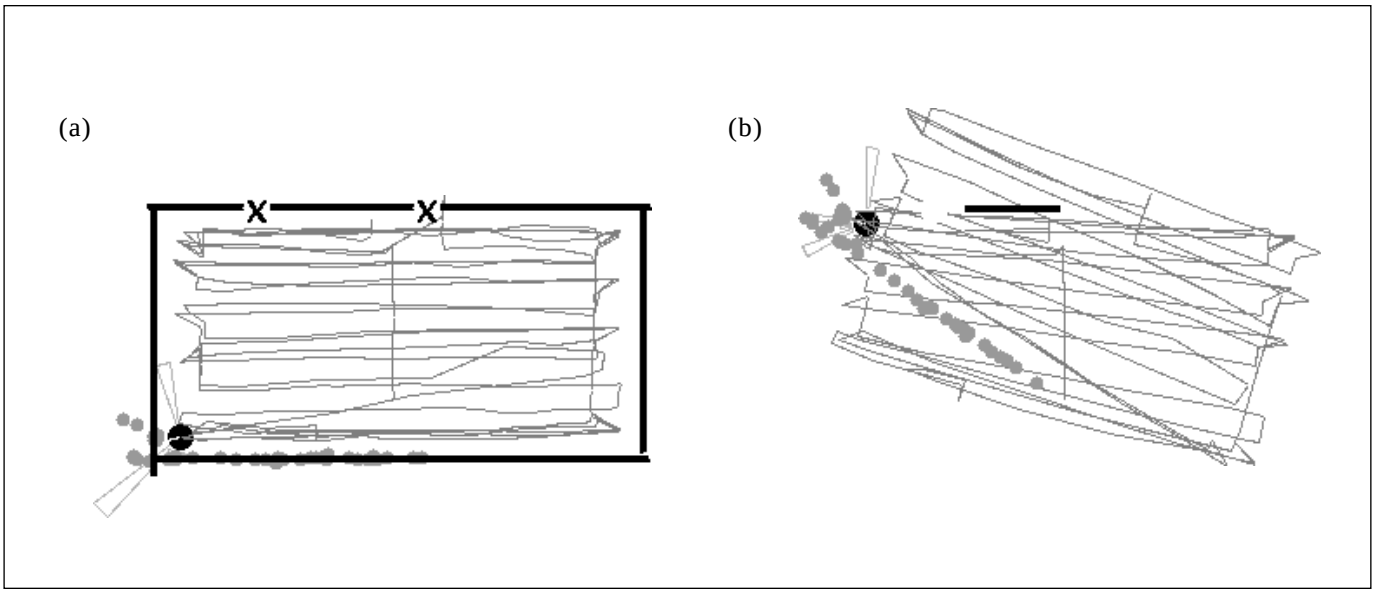

Figure 3. To Build and Use a Geometric Model, It Is Important to Know Where the Robot Is Relative to the Model.

A. This figure shows the path followed by JEEVES during the competition finals with the error-correction mechanism. B. Compare this path with that followed without the error-correction mechanism. Without it, the final dead-reckoning error is approximately 3.9 meters and 34°—clearly too much for any practical application.

cannot be detected. The latter effect is usually referred to as *total reflection*. As a result, only some of the sonar measurements reflect proximity.

Fortunately, trustworthy sonar measurements were easy to identify because of the structured nature of the competition ring. As part of its geometric model of the environment, the robot continuously estimated its relative orientation to the surrounding walls. By comparing its own orientation with the pointing direction of each individual sensor and the orientation of the walls, JEEVES identified which sensor was orthogonal to a wall and, thus, was likely to be correct.

In addition, sonar measurements were also corrupted whenever the motors drew too much power, as noticed previously. Power consumption could be deduced from the robot's status report by reconstructing its acceleration-deceleration when a measurement was taken. Only those sensor values that passed both of these filters—angle to wall and total power consumption—were used for the attributes that are described in the following subsections.

Finding Wall Segments

Once sensor values are filtered appropriately, the vast majority of them correspond to proximity, and detecting walls is a straightforward exercise. On JEEVES, the sonar sensors are divided into three sets: (1) the left sensor, (2) the right sensor, and (3) the three frontmost sensors (where each sensor angle differs from its neighbors by 15°). Within each set of sensor measurements, JEEVES checked whether the last 4 to 20 measurements corresponded to a straight line. If this was the case, the corresponding line segment was considered a piece of a wall. As such, it was used for two purposes: (1) augmenting the map and (2) identifying errors in the robot's odometry.

Building Maps

Admittedly, building a map in an unpopulated arena where all walls are either parallel or orthogonal simplifies the matter. In our previous work on indoor robot navigation, the team developed a more sophisticated probabilistic approach for building integrated metric-topological maps (Thrun et al. 1997), but the restrictive nature of the competition ring enabled us to use the following nonprobabilistic approach to map building:

When the robot found its first wall segment, it was considered to be part of a wall. The first wall segment was special in that it determined the principle wall orientation of the environment; all walls could only be par-

allel or orthogonal to the principal wall orientation. When a new wall segment was observed, the robot checked if this segment was part of an existing wall; in which case, it extended the existing wall. New walls were only added when they were parallel or perpendicular to the principal wall orientation. Thus, wall segments incrementally increased the environment model. The model finally could contain an arbitrary number of walls, as long as they were all parallel or orthogonal to each other.

Position Control

To use a map, JEEVES had to know its relative location therein. Unfortunately, drift and slippage introduce noise into dead reckoning, which, if not compensated for, can lead to a dramatic mismatch between the robot's internal belief and reality. Figure 3 illustrates the mismatch, using data from the competition finals as an example. After traversing the competition ring twice (approximately 15 minutes of autonomous robot operation), the cumulated dead-reckoning error amounted to approximately 3.9 m and 34°—clearly too much for the map to be of any practical use. This result illustrates the importance of position control in map-based mobile robotics.

JEEVES used wall segments to correct for dead-reckoning errors. Whenever it observed a wall segment, it checked if its orientation (modulo 90°) was reasonably close to the global wall orientation; in which case, the difference was used to correct its internal orientation. It also checked if the wall segment was close to a wall in the model; in which case, the spatial deviation, if any, was used to correct the robot's internal x-y location. Both updates were in proportion to the observed deviation. Experimentally, the team found that JEEVES could repeatedly operate in our university hallways for durations of several hours without losing its location. The position-control mechanisms were crucial for using maps in the competition.

Target-Point Navigation

Knowing the walls and the current location facilitates the navigation to arbitrary target locations. Whenever possible, JEEVES approached a target point on a straight line. JEEVES also obeyed a 30-cm safety distance to the walls. When moving to a target point that was within the 30-cm safety zone, it first moved to the nearest point outside the safety zone, then turned and moved toward the target point so that it directly faced the adjacent wall. This two-step procedure ensured that JEEVES did not come unnecessarily close to walls, but if it had to, its brush would be aligned totally with the wall—a necessary prerequisite for picking up balls next to a wall and dumping the balls into the gate.

Moving Parallel to Walls

JEEVES's systematic sweeping pattern required that the robot move parallel to a wall. It was repeatedly required that the robot move with as little as 5-cm side clearance parallel to a wall at a velocity of 60 cm/sec. When moving with that small a side clearance at full speed, accurate localization becomes a critical issue, particularly because the robot's hardware prohibits changes of the motion directions at a ratio of more than 2 Hz.

JEEVES's wall-following routines were based on target-point navigation. To move parallel to a wall, a target point was generated periodically 5 m ahead of the robot, whose distance to the wall was approximately the desired wall distance. This strategy was successful: JEEVES never touched a wall unintendedly in any single run during the entire competition. It is difficult to imagine that a purely reactive approach, that is, an approach that bases its decision only on its most recent sensor input, could have achieved the same result with the same precision.

Systematic Exploration

A key advantage to maps is that they enable robots to plan. For a task as simple as the one in the competition, however, deliberative planning was not even necessary because the exploration pattern could be predetermined entirely. As soon as JEEVES identified the first wall segment, it began its systematic exploration by moving parallel to it. The parallel motion was usually terminated by a frontal obstacle (part of a different wall), which prompted the robot to turn around and repeat the same pattern at an increased distance. As soon as the robot reached a wall opposite to the one it discovered first, it knew the arena had been swept systematically, and exploration was finished.

Capturing Tennis Balls

JEEVES used two different strategies for capturing a ball: (1) moving toward it and (2) not moving toward it. JEEVES basically ignored any tennis ball in the interior of the arena (beyond the 30-cm safety zone) because its sweeping pattern systematically covered the entire arena. It also ignored balls in a corner because it lacked a reliable pickup strategy. Other balls within the 30-cm safety zone

JEEVES's most significant hardware advantage was its brush, which proved surprisingly capable and robust in picking up balls.

Figure 4. In the United States, Professional Tennis Trainers Spend an Estimated Annual Average of $6500 of Their Customers' Money Letting Them Pick Up Balls—Certainly Not the Most Beloved Aspect of the Sport.

were treated differently: JEEVES moved toward them with its brush carefully aligned with the wall until it finally touched the wall. After the ball was picked up, JEEVES returned to the location where it first saw the wall to check if the pickup was successful. If not, the same pattern was repeated. This strategy proved extremely reliable in exhaustively picking up all tennis balls in the arena (figure 4).

It is interesting to note that in one of the preliminary competition runs, the team temporarily modified the pickup strategy so that the robot did not ignore balls in the interior of the competition ring. Here, the robot picked up interior balls directly, whenever a ball was observed. Because actively picking up a ball makes the robot deviate from its preplanned sweeping path, it had to return to the point where it first saw the ball and continue from there—quite a time-consuming maneuver. As a result, the time required for sweeping the arena was approximately doubled, and it took the entire competition time to sweep the arena only once.

Capturing Squiggle Balls

Squiggle balls were much harder to capture because physical limitations prohibited JEEVES from chasing them around. JEEVES visually tracked the squiggle balls and made every attempt to chase them. When a squiggle ball was visible, JEEVES extrapolated the motion direction from current and past observations and moved toward the anticipated next location of the squiggle ball. Because of the slow command rate (2 Hz), however, squiggle balls usually disappeared from the perceptual field before even the second or third turning command could be issued. The reader might notice that squiggle balls were the only aspect of the environment that was not fully modeled. JEEVES was able to detect them in a 1.2-m range, but it forgot about them as soon as they left its visual field.

The modeling and chasing limitations did not impair the robot's ability to successfully catch both squiggle balls. For an arena as small as the competition ring, my team quickly learned that it was extremely likely that both squiggle balls were captured just by chance, within the allotted time (15 minutes [min]). In fact, in every single run—testing, preliminaries, and finals included—both squiggle balls were captured within the first 10 min.

Returning to the Gate

After capturing all the balls, JEEVES was required to move to the gate and unload its balls. The gate was marked by two cyan markers that were taped to the ground in front of the gate. JEEVES was able to model multiple hypotheses for the location of the gate. Whenever it saw a cyan marker, it determined whether this marker had been seen before. If the marker had not been seen before, it was entered into the map as a new hypothesis for the location of the gate. Markers that had been seen before were used to better estimate the exact coordinates of the marker, using a weighted average algorithm.

Once JEEVES reached the other side of the arena, it terminated its systematic sweeping pattern and moved back to the gate, where it reversed its brush direction to unload the balls. If multiple hypotheses existed (as is the case in figure 2, where the vision system accidentally mistook a reflection on the ground for a marker), it chose the one for which it had the most sensor evidence (total number of pixels). JEEVES maintained multiple hypotheses about the location of the gate because I could not train the vision system to avoid false-positive measurements. However,

over time, the gate usually provided orders of magnitude more evidence than false-positive readings.

Velocity Control

The faster a robot moves, the faster it completes the task. This simple rule led me to make the robot almost always move with its maximum velocity. However, sometimes it is wise to move slower. JEEVES's velocity was controlled by the dynamic window approach described in Fox, Burgard, and Thrun (1995). In essence, the dynamic window approach sets the velocity in accordance to the proximity of obstacles, assuming the robot stays on its current trajectory. As a result, JEEVES traveled at its maximum speed until it approached an obstacle; in which case, it gracefully decelerated and finally halted.

The Case for Models: Scaling Up

JEEVES's control strategy was based on a centralized geometric model. As described in a previous subsection, JEEVES memorized the location of walls, balls, and gates—basically everything there was to be known for the task of picking up balls. JEEVES's control strategies benefited from the existence of this model. For example, I quickly learned that to pick up a ball next to a wall, the exact angle between the robot's brush and the wall mattered. The model made it easy to accurately control this angle. The model also facilitated various other things, such as following walls at a 5-cm distance at full speed, determining a strategy and time for picking up balls, moving backward without bumping into obstacles, finding the gate, and determining when to return to the gate. Obviously, the internal model was crucial for JEEVES's success at the competition because many of these capabilities would have been difficult to achieve without an internal model.

Recently, there has been a more general discussion about the nature and the utility of models in robotics. It has been argued that the environment is its own best model (Brooks 1991), an argument that has often been interpreted in favor of reactive approaches that maintain a minimum of internal state. To investigate the validity of such a claim, one has to be careful to specify what purpose the model is supposed to serve.

Undoubtedly, the environment is its most accurate model. How can any other model be more accurate than reality itself? Accuracy alone, however, is not sufficient for robot control. To be of practical use, a model must also be accessible, and unfortunately, the environment is often not its own most accessible model. In mobile robotics, the accessibility of the environment depends, among other things, on the perceptual ratio of the robot to its environment, that is, the ratio of the perceptual range of the robot relative to the size of its environment. The perceptual ratio is of practical importance because to gain knowledge about the environment beyond the perceptual range, a robot has to actually move there. The accessibility of the environment decreases—and, thus, the utility of internal models increases—as the perceptual ratio decreases; therefore, it seems plausible that robots that acquire and maintain internal models scale better to more complex environments than those that do not.[1]

Let us investigate scalability more concretely using JEEVES and the AAAI-96 mobile robot competition as an example. To contrast JEEVES's model-based approach, let us also consider a purely *reactive robot*, that is, a robot that makes decisions based on a short history of perceptual input with a minimum of internal state. A typical reactive robot would move around somewhat randomly while it possibly followed a wall, until a ball appears in its visual field. A typical reactive approach might then chase this ball and, after a successful capture, continue its random walk until it comes across another ball or the gate, at which point it would either capture the ball or deposit previously captured balls into the gate. In fact, I suspect that variants of this reactive, model-free approach were used by several other teams at the competition—with remarkable success, as the impressive entry by Newton Research Labs illustrates (see article by Sargent et al., also in this issue).

Although a purely reactive robot might perform well in an environment as small as the competition ring, it is difficult to imagine that such a robot would scale up to more complex environments. For example, consider the following three situations:

First, consider an arena 10 times as large. The larger the arena, the smaller the perceptual ratio, and the more the robot has to search. Because a purely reactive robot would run danger to search the same part of the arena over and over again, its chances of exhaustively covering the arena within a given time are smaller than that of a model-based robot. The advantage of systematic search increases with the size of the environment. Although reactive (history-free) search might work reasonably well for finding balls where the number of total balls is large, theoretical and em-

To be of practical use, a model must also be accessible, and unfortunately, the environment is often not its own most accessible model.

pirical research on the complexity of search has shown that history-free search strategies tend to be inefficient for searching large environments exhaustively (Koenig and Simmons 1993).

Second, consider an arena with 10 times as many balls. More balls would force a robot to return to the gate more frequently. A purely reactive approach would require that the robot search for the gate even if it has been there before. If the gate is hidden in an unaccessible corner unlikely to be found by random motion, a purely reactive robot could easily waste a lot of time searching for the gate over and over again, whereas a model-based approach that remembered the location of the gate could move there directly—just like JEEVES.

Third, consider the same task with one-tenth the time. Efficiency becomes even more important as time becomes a limiting factor. It is important to notice that models, if used the right way, do not slow robots down. In fact, the opposite is true. JEEVES's performance illustrates that models can enable a robot to make more rational action choices in real time, yielding more efficient control. Because learned models integrate multiple sensor measurements, model-based robots also tend to be more robust to noise in perception than purely reactive robots.

Although it should not be dismissed that JEEVES's control software—in its current version—also faces some scaling limitations because of its inability to handle nonorthogonal walls or huge open spaces, these scaling limitations are not caused by the fact that JEEVES used a model; instead, they carefully exploit the restrictive nature of the task and the environment. For example, a simple one-line change in JEEVES's software would have enabled it to model arbitrary, nonorthogonal walls. In previous research, the team successfully demonstrated robust localization even in large-scale environments with huge open spaces, using probabilistic approaches based on models (Thrun et al. 1997; Burgard et al. 1996).

To summarize, I believe that model-based approaches scale better to more complex environments and complex tasks that people (outside the scientific community) really care about. Although it might be tempting to program robots by connecting sensors directly to actuators, such approaches are unlikely to scale up and provide the level of sophistication required in all but the most simple mobile robotics applications.

Acknowledgments

I want to thank Hans Nopper and Grinnell More from Real World Interfaces, Inc., for designing and building the hardware. JEEVES would not exist without the enthusiastic initiative of Hans Nopper. I would also like to thank Randy Sargent, Anne Wright, and Carl Witty from Newton Research Labs for their help setting up the vision system and Shyjan Mahamud for his initial help training it. The name JEEVES was suggested by Greg Armstrong.

Note

1. The obvious exception to this argument is robots that perform tasks that require exclusively local sensor information; however, such tasks are often trivial and rarely of interest in robotics.

References

Brooks, R. 1991. Intelligence without Reason. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, 569–595. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Burgard, W.; Fox, D.; Hennig, D.; and Schmidt, T. 1996. Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 896–901. Menlo Park, Calif.: American Association for Artificial Intelligence.

Fox, D.; Burgard, W.; and Thrun, S. 1995. The Dynamic Window Approach to Collision Avoidance, Technical Report, IAI-TR-95-13, University of Bonn.

Koenig, S., and Simmons, R. G. 1993. Complexity Analysis of Real-Time Reinforcement Learning. In Proceedings of the Eleventh National Conference on Artificial Intelligence, 99–105. Menlo Park, Calif.: American Association for Artificial Intelligence.

Thrun, S.; Bücken, A.; Burgard, W.; Fox, D.; Fröhlinghaus, T.; Hennig, D.; Hofmann, T.; Krell, M.; and Schimdt, T. 1997. Map Learning and High-Speed Navigation in RHINO. In AI-Based Mobile Robots: Case Studies of Successful Robot Systems, eds. D. Kortenkamp, R. Bonasso, and R. Murphy. Cambridge, Mass.: MIT Press.

Sebastian Thrun is a member of the research faculty in Carnegie Mellon University's Computer Science Department. He earned his Ph.D. from the University of Bonn, Germany, in 1995. Thrun's research interests lie in machine learning and robotics.