To Know or Not To Know: On the Utility of Models in Mobile Robotics

Sebastian Thrun
Department of Computer Science
Carnegie Mellon University
http://www.cs.cmu.edu/∼thrun

Abstract—**This article describes Jeeves, one of the winning entries in the 1996 AAAI mobile robot competition. 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 while the robot performs 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 paper 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.**

1 Introduction

Jeeves was the Carnegie Mellon entry in the event "clean up a tennis court" at the 1996 AAAI mobile robot competition. Robots competing in this event were given 15 minutes to collect ten randomly scattered tennis balls along with two self-propelled squiggle balls in an arena of size 9 by 5 meters, and to deliver them to a custom-built gate.

A picture of Jeeves is shown in Figure 1. Jeeves was originally built as a service robot with an eye towards commercialization, independently of the competition. The light-weight robot is equipped with a large rotating brush, capable of capturing up to 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.

So what made Jeeves as successful as it was? A key aspect of Jeeves's success was the fact 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:

- the location of the walls
- the location of the gate
- the location of the tennis balls
- the location and motion direction of the squiggle balls
- its own location with respect to the model,
- where it had been before, and what parts of the arena 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 in order 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 we found the model-based approach to control to be extremely robust; we 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 Jeeves software. It also argues more generally for the utility of models in mobile robotics. Using Jeeves and the AAAI mobile robot competition as an example, it discusses the role of models in scalable mobile robot architectures.

2 Hardware

Jeeves's hardware has been designed and built by German design student Hans Nopper, in collaboration with Real World Interface Inc., a leading mobile robot manufacturer. The robot moves with an approximate maximum speed of 60 cm per second. It is equipped with seven ultrasonic proximity sensors (of which 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 meters, making Jeeves one of the most myopic robots on the stage. To pick up balls, Jeeves used a rotating brush

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 cm/sec. When equipped with a basket, Jeeves can hold approximately 100 balls—sufficiently many for a real tennis court.

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' 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 due to the graphical control interface; not counting it, the remaining load was well below 20% of the available computational resources. The remote software received status updates from the robot and its sensors at a frequency of 10 Hz. Unfortunately, the maximum command rate for changing the motion direction was only 2 Hz (or less) due to limitations of the on-board controller. When designing the software, special attention was also payed 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, we 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.

3 Models

Jeeves's success cannot be attributed to hardware alone. Various other teams employed 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. Using data recorded at the competition finals as an example, Figure 2 highlights the major components of Jeeves. The reader not interested in the details might inspect this figure and Figure 3, and then directly move to the next section.

3.1 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. With a little bit of physics, it is easy to see that for smooth objects chances to receive a sonar echo depend on the angle between the main sonar cone and the reflecting object. Sound waves that hit a wall frontally are very 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 into a direction where it cannot be detected. The latter effect is usually referred to as *total reflection*. As a result, only some of the sonar measurements reflect proximity, whereas others do not.

Fortunately, trustworthy sonar measurements were easy to identify, due to 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, thus was likely to be "correct."

In addition, sonar measurement were also corrupted whenever the motors drew too much power, as noticed above. 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 what is described in the remainder of this section.

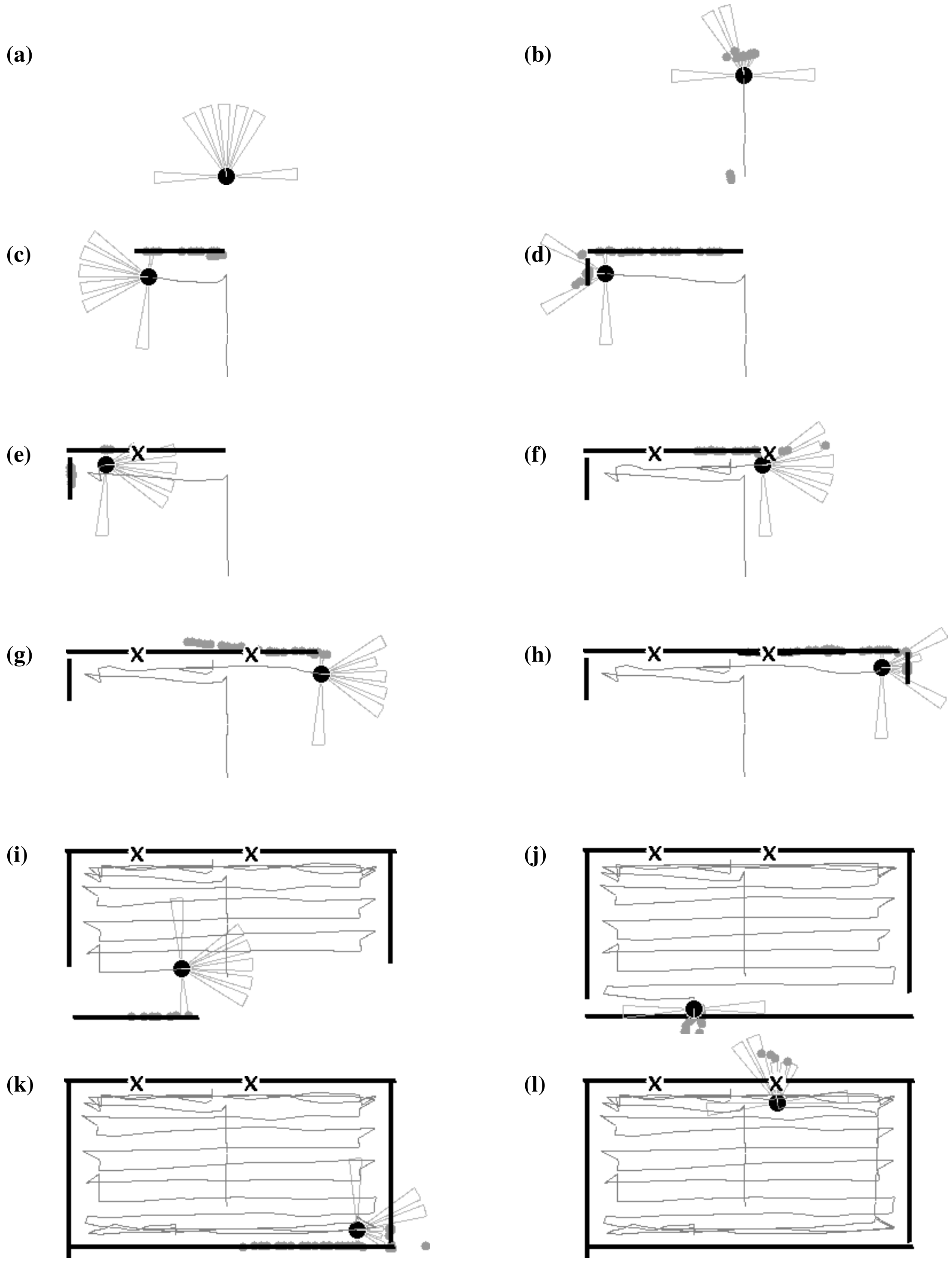

Figure 2: This figure summarizes Jeeves's performance in the competition finals: (a) The robot starts without a model. (b) It approaches the first wall, as indicated by the projected sonars measurements. (c) After turning left, Jeeves detects the first wall. (d) While extending the first wall, Jeeves 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 and (i) the fourth walls are 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.

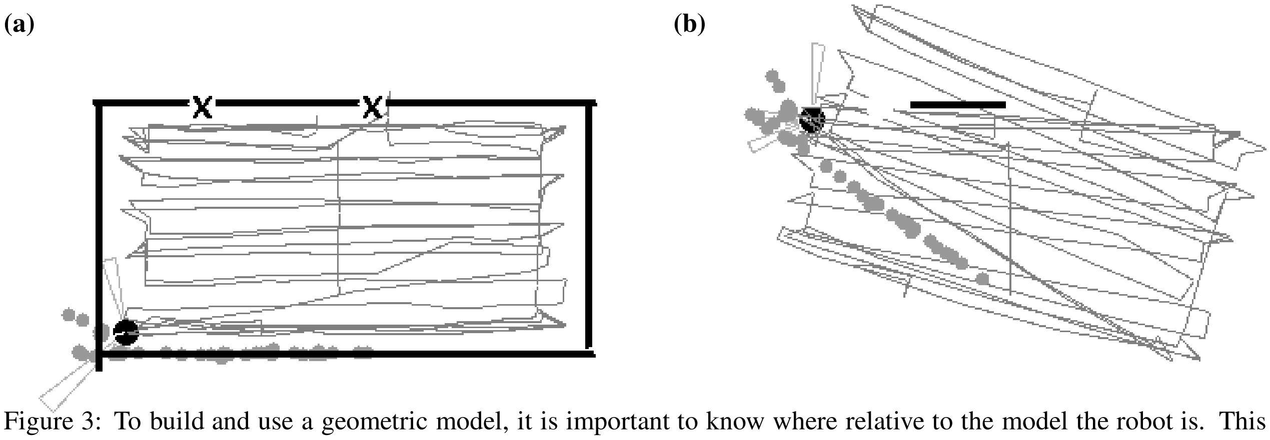

Figure 3: To build and use a geometric model, it is important to know where relative to the model the robot is. This figure compares the path followed by Jeeves during the competition finals (a) with and (b) without the error correction mechanism described in the paper. Without it, the final dead-reckoning error is approximately 3.9 meter and 34 degrees—clearly too much for any practical application.

3.2 Finding Wall Segments

Once sensor values are filtered appropriately, the vast majority of them does correspond to proximity, and detecting walls is a straightforward exercise. On Jeeves, the sonar sensors were divided into three sets: the left sensor, the right sensor, and three frontmost sensors (where each sensor angle differs from its neighbors by 15°). Within each set of sensor measurements, Jeeves checked whether or not 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: augmenting the map, and identifying errors in the robot's odometry.

3.3 Building Maps

Admittedly, building a map in an unpopulated arena where all walls are either parallel or orthogonal simplifies the matter. Whereas in our previous work on indoor robot navigation, we developed a more sophisticated probabilistic approach for building integrated metric-topological maps [5], the restrictive nature of the competition ring enabled us to use the following non-probabilistic 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 were only allowed to be parallel 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 arbitrarily many walls, as long as they were all parallel or orthogonal to each other.

3.4 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, 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 (approx. 15 minutes of autonomous robot operation), the cumulated dead-reckoning error amounts to approximately 3.9 meter and 34 degrees—clearly too much for the map to be of any practical use. This illustrates the importance of position control in map-based mobile robotics.

Jeeves used wall segments to correct for dead-reckoning errors. Whenever if observed a wall segment, it checked if its orientation (modulo 90°) was reasonable close 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, we found that Jeeves could repeatedly operate in our University hallways for durations of several hours without loosing its location. The position control mechanisms was crucial for using maps in the competition.

3.5 *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 towards 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, yet if it had to, its brush would be totally aligned with the wall—a necessary prerequisite for picking up balls next to a wall and for dumping the balls into the gate.

3.6 *Moving Parallel to Walls*

Jeeves systematic sweeping pattern required that the robot moved parallel to a wall. It was repeatedly required that the robot moved 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, since the robot's hardware prohibited changes of the motion directions at a ratio of more than 2 Hz.

Jeeves wall-following routines were based on target point navigation. To move parallel to a wall, a target point was generated periodically 5 meter ahead of the robot, whose distance to the wall was basically the desired wall distance, plus a small term that counterbalanced deviations from the desired 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, *i.e.*, an approach which bases its decision on its most recent sensor input only, could have achieved the same result with the same precision.

3.7 *Systematic Exploration*

A key advantage of 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. Instead, the exploration pattern was entirely predetermined. As soon as Jeeves identified the first wall segment, it begun 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 to 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.

3.8 *Capturing Tennis Balls*

Jeeves employed two different strategies for capturing a ball: Moving towards it and not moving towards it. Jeeves basically ignored any tennis ball in the interior of the arena (beyond the 30 cm safety zone), based on the observation that its sweeping pattern systematically covered the entire arena. It also ignored balls in a corner, due to lack of a reliable pick-up strategy. Other balls within the 30 cm safety zone were treated differently: Jeeves moved towards 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 that 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.

It is interesting to notice that in one of the preliminary competition runs, we 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. Since actively picking up a ball makes the robot deviate from its pre-planned 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.

3.9 *Capturing Squiggle Balls*

Squiggle balls were much harder to capture, since physical limitations prohibited 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 towards the anticipated next location of the squiggle ball. Due to 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 may 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 meter 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, we quickly learned that it was extremely likely that both squiggle balls were captured just by chance, within the allotted time (15 minutes). In fact, in every single run—testing, preliminaries and finals included—, both squiggle balls were captured within the first 10 minutes. We suspect that even an immobile device with a brush would have been equally likely to capture all squiggle balls within the allotted time.

3.10 Returning to the Gate

After capturing all balls, the task required Jeeves to move to the gate and to unload its balls. The gate was marked by two cyan markers, 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 or not 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 concerning the location of the gate because we were unable to train the vision system so as to avoid false-positive measurements. However, over time the gate usually provided orders of magnitude more evidence than false-positive readings.

3.11 Velocity Control

The faster a robot moves, the faster it completes the task. This simple rule led us to make the robot almost always move with its maximum velocity. However, sometimes it is wise to move slower. Jeeves velocity was controlled by the dynamic window approach described in [3]. 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.

4 The Case for Models: Scaling Up

Jeeves control strategy was based on a centralized geometric model. As described in the previous section, Jeeves memorized the location of walls, balls, and gates—basically everything there was to be known for the task of picking up balls. Jeeves control strategies benefited from the existence of this model. For example, we 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 very 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 backwards 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, as 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* [1]—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 in specifying as to what purpose the model is supposed to serve: *best for what?*

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, *i.e.*, 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, and thus the utility of internal models, increases as the perceptual ratio decreases; therefore, it seems to be plausible that robots which 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 mobile robot competition as an example. To contrast Jeeves's model-based approach, let us also consider a purely *reactive* robot, *i.e.*, 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 possibly following a wall, until a ball appears in its visual field. A typical reactive approach might then chase this ball, and after a successfully 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, we suspect that variants of this reactive, model-free approach were employed by several other teams at the competition—with remarkable success, as the entry by Newton Research Labs illustrates (described in a different paper in this volume).

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

While 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

. . . an arena ten time as large. The larger the arena, the smaller the perceptual ratio, and the more the robot has to search. Since a purely reactive robot would run danger to search the same part of the arena over and over again, its chances to exhaustively cover 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. While reactive (history-free) search might work reasonably well for finding balls as long as the number of total balls is large, theoretical and empirical research on the complexity of search has shown that history-free search strategies tend to be very inefficient for searching large environments exhaustively [4].

. . . an arena with ten times as many balls. More balls would force a robot to return to the gate more frequently. A purely reactive approach would have to 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 enormous amounts 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.

. . . the same task with a tenth of 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 the case. Jeeves's performance illustrates that models can enable a robot to make more rational action choices in real-time, yielding more efficient control. Since models integrate multiple sensor measurements, model-based robots tend also to be more robust to noise in perception than purely reactive robots.

While it should not be dismissed that Jeeves's control software—in its current version—also faces some scaling limitations, due to its inability to handle non-orthogonal walls or huge open spaces, the scaling limitations of Jeeves are not caused by its 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 Jeeves to model arbitrary, non-orthogonal walls. In our previous research, we have successfully demonstrated robust localization even in large-scale environments with huge open spaces, using probabilistic approaches based on models [2, 5].

Figure 4: In the US, 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 that sport.

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

Acknowledgment

The author would like to thank Hans Nopper and Grinnell More from Real World Interface Inc. for designing and building the hardware. Jeeves would not exist without the enthusiastic initiative of Hans Nopper. The author 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.

REFERENCES

[1] Brooks, R. *Intelligence without reason*. in: **Proceedings of IJCAI-91**, IJCAI, Inc. 1991, pp. 569–595.

[2] Burgard, W., Fox, D., Hennig, D., and Schmidt, T. *Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids*. in: **Proceedings of the Thirteenth National Conference on Artificial Intelligence**, AAAI. AAAI Press/MIT Press, Menlo Park, 1996.

[3] Fox, D., Burgard, W., and Thrun, S. *The Dynamic Window Approach to Collision Avoidance*. **IEEE Robotics and Automation**, to appear. *(also appeared as Technical Report IAI-TR-95-13, University of Bonn, 1995)*.

[4] Koenig, S. and Simmons, R. G. *Complexity Analysis of Real-Time Reinforcement Learning*. in: **Proceeding of the Eleventh National Conference on Artificial Intelligence AAAI-93**, AAAI. AAAI Press/The MIT Press, Menlo Park, CA, 1993, pp. 99–105.

[5] Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M., and Schimdt, T. *Map Learning and High-Speed Navigation in RHINO*. in: **AI-based Mobile Robots: Case studies of successful robot systems**, edited by D. Kortenkamp, R. Bonasso, and R. Murphy. MIT Press, Cambridge, MA, to appear.

ABOUT THE AUTHOR

Sebastian Thrun is member of the research faculty at CMU'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.