Chapter 1

Introduction

DRAFT OF CHAPTER 1 (8 Jan 2013)
(likely to be updated soon)

Mobile Robotics: An Information Space Approach

Steven M. LaValle, University of Illinois

Mobile robots are fascinating because they bring our ideas to life. By connecting computers to motors and sensors, robots are able to move themselves around, learn about their environment, interact with people, and accomplish many useful tasks. They inspire children of all ages, engineering students, hobbyists, news media, and a large number of researchers. As the robotics industry continues to grow, we see them appearing all over, including delivering boxes in warehouses, cutting grass and vacuuming floors in homes, entertaining children, and driving themselves down the road.

1.1 Mobile Robots Over the Years

For over a century, people have designed and built mobile robots of all shapes, sizes, and purposes. As early as 1868, Zadock Dederick patented a steam-powered humanoid (Figure 1.1). An electronic light-seeking dog was made in 1912 by John Hammond, Jr. and Benjamin Miessner (Figure 1.2(a)). In the late 1940s, Grey Walter constructed several electronic tortoises (Figure 1.2(b)), that used sensors to move toward light and avoid obstacles. Edmund Berkeley in 1951 took it to the next level by making an electronic squirrel that could gather objects (Figure 1.2(c)). In 1952, Claude Shannon, the father of *information theory*, constructed an impressive, mechanical maze-exploring mouse (Figure 1.2(d)). The circuitry of these robots was usually simple, connecting motor responses directly to sensed stimuli (Figure 1.3). Piero Fiorito unveiled a gigantic humanoid robot called Cygan

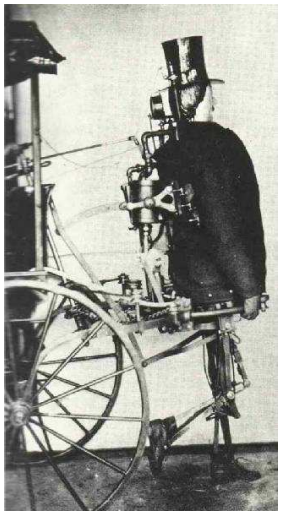
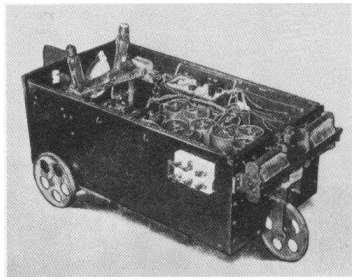


Figure 1.1: In 1868, a "Steam Man" was patented. Its torso was the boiler and its body was mounted upright onto a carriage. Using legs it could walk and steer while pulling a load in the carriage. It is rumored that US President Ulysses Grant once rode in the carriage, called the "Devil's Car".

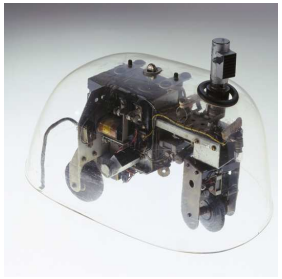
in 1957 (Figure 1.4(a)). It was able to walk, turn, and raise its arms. The Stanford Cart was one of the earliest autonomous vehicles (Figure 1.4(b)). Starting in 1960 to study possibilities of lunar exploration, versions of it were developed for two decades.

In 1966, the Shakey mobile robot system, developed by Nils Nilsson at Stanford, inspired a generation of research efforts on constructing algorithms that *plan* a collision-free route for a mobile robot through an obstacle course (Figures 1.4(c) and 1.4(d)). More advanced versions were built in labs around the world over the following decades. For example in 1977, at LAAS/CNRS in Toulouse, France, the Hilare robot was developed in and used to study the problems caused by autonomously steering the wheels. By the 1990s, academic research labs could purchase well-designed mobile robot platforms that included motor controllers, sensors, and on-board computers (Figures 1.4(e) and 1.4(f)). As sensing technology, computational power, mathematical models, and algorithmic techniques improved, robots could move reliably over longer periods of time. A prime example is the RHINO museum guide robot (Figure 1.5(a)) from the University of Bonn, which could navigate through unpredictable crowds of people, avoid difficult-to-detect exhibits, and smoothly interact with tourists.

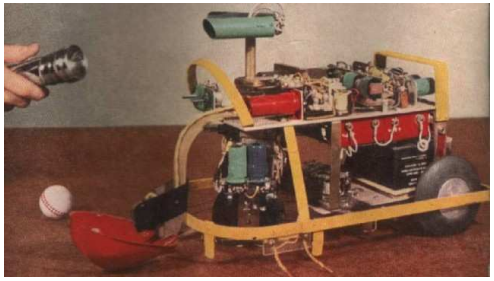
Modern mobile robots are astounding. Self-driving cars come a long way since the Stanford Cart. They have successfully navigated urban obstacle courses and driven for over 150,000 kilometers in everyday traffic (Figures 1.5(b) and 1.5(c)).



(a)



(b)



(c)

(d)

Figure 1.2: Early "animal-like" mobile robots: (a) An electronic light-seeking dog (1912). (b) A tortoise (1948). (c) A "nut" gathering squirrel (1951). (d) Claude Shannon and his maze-searching mouse (1952).

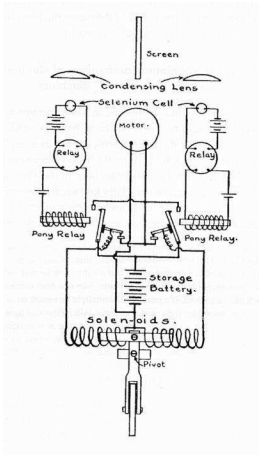
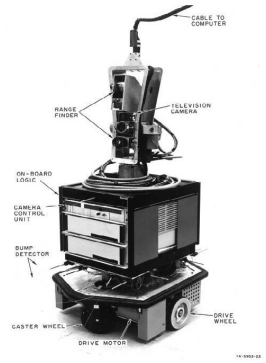


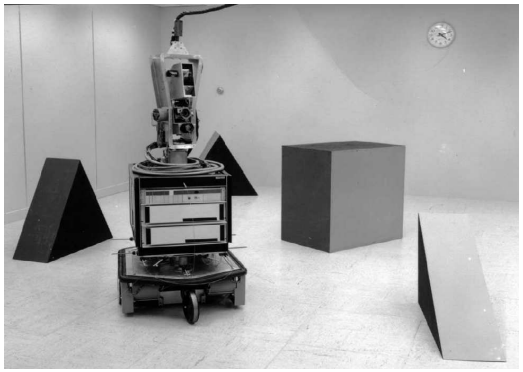
Figure 1.3: Circuit diagram of Hammond and Miessner's 1912 electric dog.

(a)

(b)



(c)



(d)

(e)

(f)

Figure 1.4: (a) Cygan the humanoid robot (1957) (b) The Stanford cart (1960s). (c) and (d) Shakey mobile robot (1966). (e) The Pioneer 3-DX8 mobile robot. (d) The Nomad XR4000 mobile robot with a Puma industrial arm mounted on top.

(a) (b) (c) (d) (e) (f)

Figure 1.5: (a) RHINO museum guide robot (1995) (b) The CMU Boss vehicle, which won the DARPA Urban Challenge. (c) The Google self-driving car, which is a modified Toyota Prius. (d) A rendering of the NASA Curiosity rover, which started exploring Mars in 2012. (e) and (f) The BigDog robot from Boston Dynamics.

Figure 1.6: The HRP-4 humanoid robots produced in Japan by National Institute of Advanced Industrial Science and Technology (AIST) and Kawada Industries.

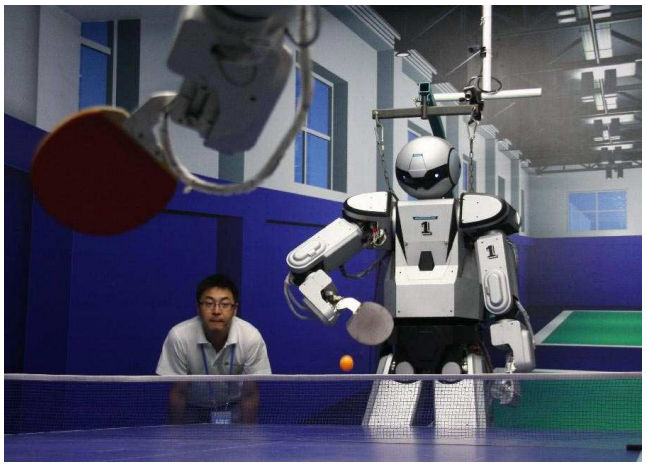


Figure 1.7: The BHR-4 robot from the Beijing Institute of Technology playing table tennis.

(a) (b)

Figure 1.8: Mobile manipulators: (a) The PR2 from Willow Garage. (b) The Kuka Youbot.

(a) (b) (c)

Figure 1.9: (a) The Aqua2 underwater swimming robot from McGill University. (b) A Hydronalix robotic boat. (c) The MQ9 Reaper Unmanned Aerial Vehicle (UAV).

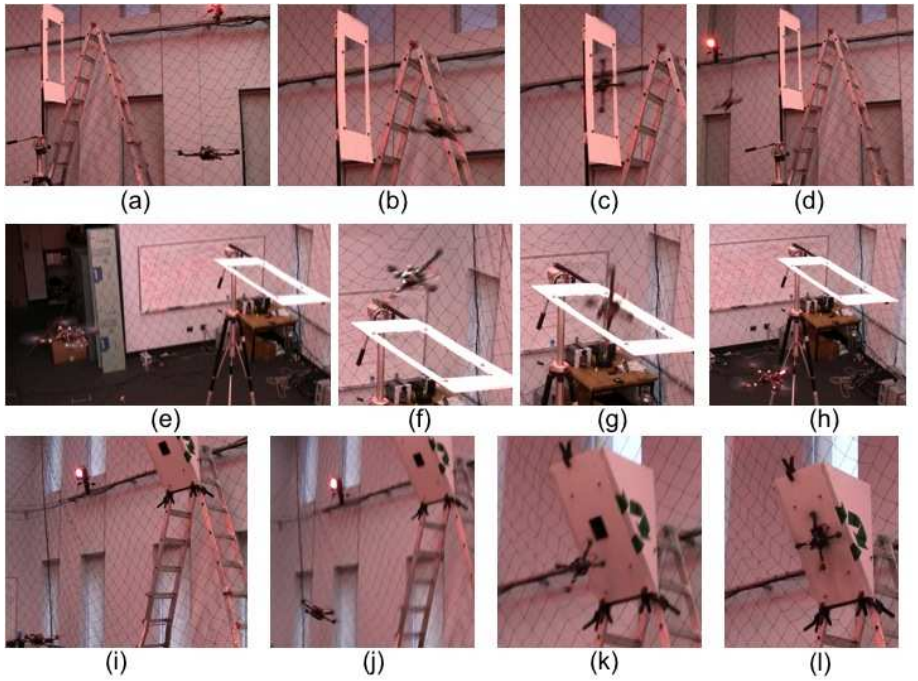


Figure 1.10: An acrobatic quad-rotor helicopter from the University of Pennsylvania.

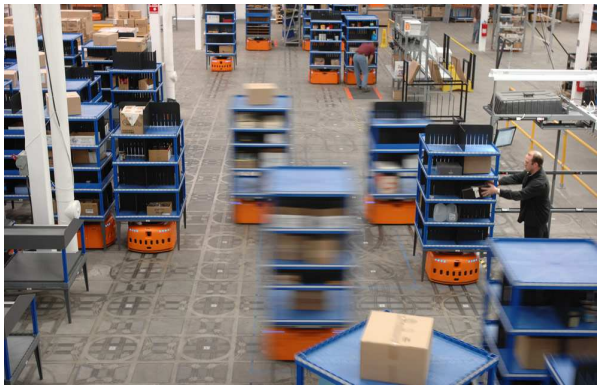


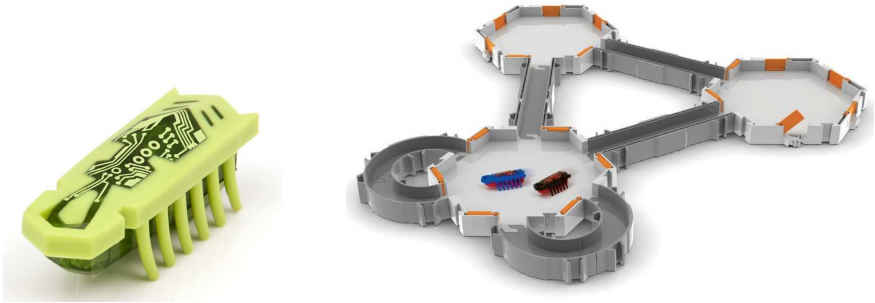
Figure 1.11: An army of Kiva mobile robots moving shelves of boxes in a warehouse.

(a) (b) (c)

Figure 1.12: (a) A Roomba vacuum cleaner by iRobot. (b) A lawn mowing robot by Husqvarna. (c) The RoboMop floor duster.

The NASA Curiosity rover landed on Mars in August, 2012 and is expected to drive autonomously for years while gathering imaging and chemical composition data (Figure 1.5(d)). Early turtles and mice have been supplanted by impressive animal-like robots, such as BigDog, by Boston Dynamics, which reliably runs over land at 6.4 kilometers per hour and can carry 150 kilograms (Figures 1.5(e) and 1.5(f)). Humanoid robots have come a long way since Cygan; some modern examples are shown in Figures 1.6 and 1.7. An alternative platform for interacting with humans is a mobile manipulator (Figure 1.8). Mobile robots also swim under water (Figure 1.9(a)), swim on top of water (Figure 1.9(b)), fly over long distances (Figure 1.9(c)), and fly through tricky obstacle courses (Figure 1.10).

Commercial successes of mobile robots have continued to grow, often with the introduction of simpler systems that are designed to be low-cost, robust, and reliable. For example, Kiva Systems developed a mobile robot system that efficiently arranges boxes in a large warehouse (Figure 1.11). The robots sort the boxes based on their frequency of use and bring the requested boxes directly to the user.



(a) (b)

Figure 1.13: (a) A Hexbug Nano. (b) A habitat in which they can navigate along corridors and through rooms. More recent versions include one-way doors and uphill ramps that serve as escalators.

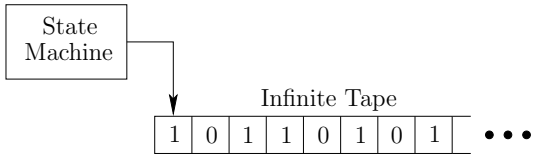
The company was recently acquired by the Amazon on-line purchasing corporation to improve its distribution efficiency. The well-known Roomba vacuum cleaning robot (Figure 1.12(a)) lives in millions of households and has stimulated a whole industry of competitors for vacuuming and related tasks such as cutting grass (Figure 1.12(b)) and mopping. One of the simplest and most humorous designs is the RoboMop, which is little more than a Weaselball enclosed in a dusting ring (Figure 1.12(c)). The Weaselball is a toy that sells for around $4 US and contains only a battery connected to a motor that oscillates around 2 Hz.

1.2 Playing a Game with Imperfect Information

During all this time, roboticists have struggled to understand what should be in the "brain" of a successful robot. There are numerous designs that we can all draw lessons from. For example, many robots use powerful sensors, such as laser scanners, cameras, and GPS units to precisely map their world and maintain their position on the map. This corresponds to a large robot brain that contains a detailed model of itself and its surroundings. At the other extreme, some robots accomplish their tasks quite well with simple sensors and little or no memory. Such simple robots are often inspired by nature, which provides an abundance of simple creatures, such as bees or ants, that have no trouble with navigation, finding food, avoiding predators, building homes, and so on. Even the widely available Hexbug Nano toys (Figure 1.13) appear to navigate in clever ways in spite of only being able to vibrate.

To understand the fundamental difficulty, it is helpful to first think about what differentiates robotics from ordinary computation. With computing systems and sensors embedded into virtually everything (phones, cars, keys, ...), it is becoming

(a)



(b)

Figure 1.14: Computation models far form mobile robotics: (a) The TRS "Trash" 80, a fine example of classical computing. (b) A Turing machine, which can be imagined as a finite state machine that can read and write bits to an infinitely long tape.

increasingly difficult to define "ordinary" computation. Therefore, suppose we travel back in time to the 1970s and want to run some software using the computer shown in Figure 1.14(a). All interactions with the outside world come for either the keyboard or a removable magnetic disk that stores programs and data. Even better, we could go back to the 1920s and imagine what might be possible if a computer could be built that implements the beautiful abstraction known as a Turing machine (Figure 1.14(b)). The key idea was to create a complete and perfectly predictable environment in which information could be stored and rapidly manipulated with no mistakes or loss of any kind. In other words: Keep the error-prone and incomprehensible physical world as far away as possible and instead build up discrete components that each functions as dictated by mathematical logic.

Once computers are embedded into robots and other devices that "read" from sensors, and "write" to control motors, the irony begins. Connections to the physical world have been directly reopened, thereby reintroducing its frustrating qualities of uncertainty, unpredictability, and unrepeatability. Of course, computers have been connected in this way for many decades, which has led to successful, well-principled approaches. One of the most important is the theory and practice of digital control systems. Common examples include the control of a chemical plant, cruise control in a car, and spacecraft attitude adjustment. The tendency is to develop sensors that measure every relevant variable, quickly feed the information into the computer, and generate a signal to motors (or other effectors) that control the system as desired.

As the devices become smaller, more autonomous, and are placed into complex environments, it becomes much harder to understand precisely what "variables" need to be measured and controlled. Modeling issues tend to dominate in robotics, more than in many other fields. To solve its task, such as navigation, exploration, or vacuuming, how much does a mobile robot need to know? What should it measure about the world around it? What information should it keep track of? What

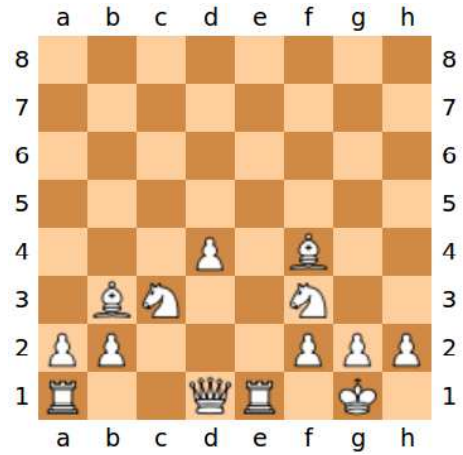


Figure 1.15: Kriegspiel: An example of a game with imperfect information. The white players knows the locations of its own pieces, but does not know where the black pieces are.

should the decision to change the signals to the motors be based upon? Loosely speaking, we will call the information in the robot's brain an *information state*, or *I-state* for short. This concept dates back to 1944, when John von Neumann (yes, that one!) and Oskar Morganstern formulated the information that players have for making decisions in a multi-stage game in which they know the complete status while playing. Recall common games such as Mastermind, Battleship, or Kriegspiel (a form of blindfold chess). In Kriegspiel, for example, each player moves its own pieces on the board, but is not aware of its opponent's moves and piece locations (Figure 1.15). In this context, the I-state might characterize *possible* board configurations, in a form that is useful for making the next decision, even though the full state of the game is not known.

Although counterintuitive, it is often useful to intentionally discard data while forming an I-state. The purpose is to make decision making simpler and more reliable. For example, consider trying to improve your odds while playing blackjack in a casino. Because the cards are not drawn with replacement, the probabilities change based on which previously played cards have been seen. Expert players develop *card counting* strategies. A simple score is given for each card that has been played, and the player increments or decrements a single number, the "count", accordingly. The decision in each card-playing round is then based on entirely on the currently visible cards and the count. A good card counting scheme can move the expected earnings advantage to the player, and the player does not have to memorize every card that has been played.

In this book, we imagine that a mobile robot is engaged in a game with its en-

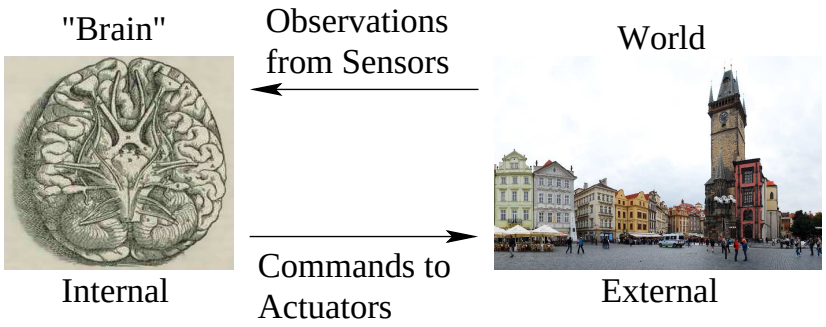


Figure 1.16: In the design of a mobile robot, we erect a boundary between internal and external realms. Information comes from the external, physical world via sensors that produce observations. Commands are transmitted to the external world that cause wheels to roll, legs to walk, and so on. The "brain" holds the information state (I-state), upon which the commands are based.

vironment. The game in this case is the task that it needs to solve, such as keeping the floor clean. Can the robot "win" its game with an effective strategy, without being able to reconstruct everything about its surrounding, physical environment? We should try to understand what information is necessary for the task and then reliably maintain that, and only that, in the robot's brain. Data regarding sensor observations and commands given to motors will be converted into an I-state that is sufficient for solving the task, but excludes unnecessary information. That is the key theme throughout this book.

1.3 The Robot's Brain

Continuing on our quest to understand what belongs in the robot's brain, consider Figure 1.16. The internal "brain" of the robot will keep track of the I-state and use it for deciding which commands to send. When the robot is switched on, the brain contains an initial I-state. As it receives new information from sensors, the I-state needs to be updated. Furthermore, the memory of which commands were previously issued may be relevant to deciding on new commands. Therefore, the I-state be also be updated to take this information into account. These concepts will be made more detailed and precise later in the book.

Before going down that path, consider an example to provide an intuitive understanding of I-states. To keep the discussion absurdly simple, suppose that as the robot moves in the physical world, a single parameter is sufficient for specifying its placement. On the right in Figure 1.17 is a robot that moves along a track. We would like the robot to travel back and forth forever along the full track length. Call this task *patrolling*. The only commands that can be issued by the brain to

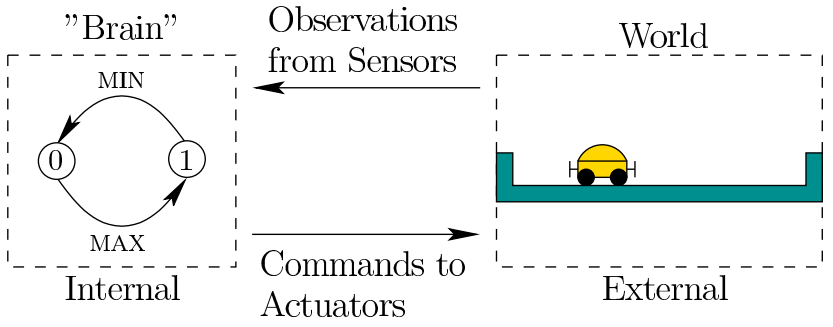


Figure 1.17: Compare to Figure 1.16. The external world is reduced to a robot that rolls along a track, capable of moving only from side to side. On each side is an independent mechanical bumper that senses whether the end of the track as been hit. A "bit brain" that solves the task of patrolling back and forth is merely a two-state machine (also called an automaton).

the motors are FORWARD, which causes motion from left to right, or REVERSE for motion in the opposite direction.

There is one sensor on each side of the robot. The right one provides the observation MAX when the robot hits the right wall (meaning the *maximum* position); otherwise, it observes FREE. The left one similarly provides MIN when contacting the left wall, or FREE, otherwise.

To solve the problem, the robot's brain will only have to maintain two possible I-states, called 0 and 1. We call it the "bit brain" robot! At any given time, the I-state must be either 0 or 1. Suppose that initially, the I-state is 0 and the robot is on the midpoint of the track. To make the robot move, the following strategy is implemented:

I-state	Command
0	FORWARD
1	REVERSE

This can be viewed as a list of IF-THEN statements. For example, IF the I-state is 0, THEN the robot is commanded to move forward.

The I-state remains the same over time, unless a particular sensor observation is received. If MAX is received while in I-state 0, then the I-state immediately becomes 1; otherwise, the robot would grind itself into the wall. This causes the command to switch from FORWARD to REVERSE. Likewise, if MIN is received, then the I-state becomes 0. These transitions continue indefinitely, achieving the desired patrolling behavior.

This example already contains enough complexity to illustrate four important spaces that will be defined and used throughout this book:

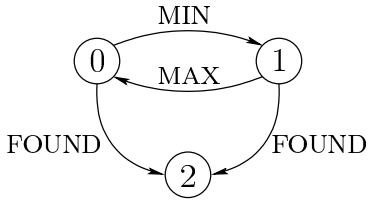


Figure 1.18: To solve a searching problem, a new I-state is added and a sensor that detects whether the spot has been FOUND.

Name		Example	Meaning
Information space	$\mathcal{I}$	{0,1}	Internal brain states
Observation space	$\mathcal{Y}$	{MAX, MIN, FREE}	Possible sensor outputs
Action space	$\mathcal{U}$	{FORWARD, REVERSE}	Possible commands
Physical state space	$\mathcal{P}$	$[0, 1]$	External parameter values

The *information space*, $\mathcal{I}$, is the set of all possible I-states. The *observation space*, $\mathcal{Y}$, is the set of all possible outputs of sensors. The *action space*, $\mathcal{U}$, is the set of all possible commands that the brain can issue to cause motions or induce other changes in the external, physical world. Using the notation, the strategy can be written as a function $\gamma : \mathcal{I} \rightarrow \mathcal{U}$. The particular action is chosen based entirely on the I-state.

Finally, the *physical state space*, $\mathcal{P}$, is the most complicated. Each physical state $p \in \mathcal{P}$ corresponds to a possible "state of the world". It is assumed that $\mathcal{P}$ is sufficiently large to encompass all relevant possibilities. For the problem in Figure 1.17 the external world can be unambiguously captured by specifying the precise position of the robot along the track. Suppose the leftmost and rightmost positions on the track are marked with 0 and 1, respectively. All positions in between are assigned a real value in the interval $[0, 1]$ in the obvious way, making the midpoint at $1/2$. In this case, $\mathcal{P} = [0, 1]$ and each $p \in \mathcal{P}$ corresponds to a robot position and is called a *physical state* or *P-state*. The relationship between the *internal* I-states and the *external* P-states is crucial!

Now we slightly extend the task from Figure 1.17. Instead of patrolling, we want to *search* for a spot on the floor. It is placed at some location $p_{spot} \in [0, 1]$ that is unknown to the robot. A new sensor is designed that observes FOUND only if the robot is directly over the spot; otherwise, it observes NOTHING. The action space is extended by including a STOP action, which halts the robot. The simple brain in Figure 1.18 generates a solution to the search problem with three I-states. From I-states 0 and 1, the actions remain as before. From I-state 2, the STOP command is issued, causing the robot to stop on the spot and remain there indefinitely. (Of course, in a real system, time delays would cause it to stop slightly past the spot!)

Note that the robot solves the search task without ever learning p_{spot}, the spot

location. If the location had been given to the robot in advance, how would the strategy change? This information could be simply ignored because an effective strategy has already been developed. However, if we introduce a more powerful sensor that continuously produces that robot's position p along the track, then we could command it to move directly to p_{spot}, which is more efficient. If $p < p_{spot}$, then the I-state is 0, which moves the robot right. Likewise, $p > p_{spot}$ yields I-state 1. Upon closer inspection, a sensor does not necessarily have to observe p. Sensing only the sign of $p_{spot} - p$ is sufficient for the task. Imagine, for example, that the spot can be seen by a camera and appears upside down if $p_{spot} - p < 0$ and rightside up if $p_{spot} - p > 0$. If the task were even simpler, by requiring that the robot be guaranteed to run over the spot without necessarily being able to detect it, then the original strategy from Figure 1.17 would already solve it. The patrolling motion automatically causes every position in $\mathcal{P}$ to be reached.

1.4 Big Model vs. Small Model

We have already seen that even for a simple, one-dimensional world, the problems of what to sense, what to maintain in the robot's brain, and what commands to issue are becoming interesting! The challenge of determining a model that is big enough to allow good predictions and analysis while being small enough to be learnable and usable from real data exists all over engineering and the sciences. Robotics is no exception. Perhaps the modeling difficulties in robotics are about as bad as anywhere else. For the robots of Section 1.3, it was clear that the P-state did not need to be measured by sensors. The strategy did not require it and the robot brain was incapable of even storing it. We were lucky that such a simple design sufficed. As tasks become more complicated, how many more I-states are needed? How much more information about the physical world needs to be sensed? The answers to these questions will depend on many factors, including the robot motion capabilities, the complexity of its surrounding world, the sensors available, and mostly importantly, the particular tasks to be solved.

Why not make the largest and most complete brain possible? The external, physical world can be imagined as a gigantic hard drive that holds all information that we could ever need to solve the task (and much more!). Imagine having an absurdly powerful *hypersensor*, which instantaneously measures everything and records it in the brain. The hypersensor takes measurements continuously, and the brain can store all of the data along with the time at which each measurement was taken. In this case, the I-state not only encodes everything about the present, but anything about the past can be recovered as well. This leads to three issues:

1. *Implementing the hypersensor:* Of course, no real sensor can measure *everything* in a complete and accurate way. However, sensors have recently become both powerful and cheap. For example, the widely available Kinect sensor provides 3D position data and color for around $100 US. Even cheaper sensors appear everywhere, especially on our phones. Furthermore, *cloud*

robotics allows robots to access observations from sensors (and more!) distributed all over the world. The availability of this much data about the real world seemed impossible only a decade ago. Even though the hypersensor is not realizable, modern sensing systems are closer than ever, especially in comparison to two decades ago.

2. *Storage and retrieval:* Information loss and time delays are unavoidable. If computers are used, then digitization must occur over space and time. If the amount of information is large, then compressed representations and special data structures may be required to allow efficient lookup and modification to the data. For example, gigabytes of raw coordinate data of walls and objects could be converted into a polyhedral mesh.

3. *Strategy design:* Recall from Section 1.3 that a strategy is like an IF-THEN table. Imagine how large and complex the strategy $\gamma : \mathcal{I} \rightarrow U$ might be if $\mathcal{I}$ is enormous. Having more information certainly enables more complicated strategies to be developed, but it might also lead to higher computational burdens. Furthermore, the strategies might depend critically on the completeness, accuracy, and timeliness of the information provided by the hypersensor.

Because of these difficulties, it is best to carefully determine what information from the physical world actually needs to be sensed and maintained in the robot's brain. Irrelevant information can be left on the "gigantic hard drive". This is in line with the famous 1990 quote of roboticist Rodney Brooks []:

> The world is its own best model.

From the 1960s to 1980s, the predominant approach was to maintain either logical or geometric models in a computer that attempts to crisply capture exactly what should be happening in the world. Due to limited sensing technology, the separation between the I-states and P-states became too large and unpredictable. Brooks argued moving to the other extreme: Design strategies that directly connect sensor observations to actions. He and others showed that complex robot behaviors could be obtained from feeding information directly from simple sensors to the motors, much like the old animal-like robots from Figure 1.2. However, there were limits on the tasks that could be solved in this way.

During the 1990s, sensing technology dramatically improved. The SICK laser scanner, costing around $8000 US, could produce highly accurate distance measurements, in 360 directions, 30 times a second. This led to a rise in techniques that extract geometric maps of the environment from sensor data. The Simultaneous Localizaton And Mapping (SLAM) problem was robustly solved through the combination of dense sensor data, probabilistic modeling, and sampling-based algorithms. Probabilistic modeling has proved useful for accounting for noise and model uncertainties that arise when using real sensors. Sampling-based algorithms, including *particle filters*, were critical in making Bayesian posterior computations efficient.

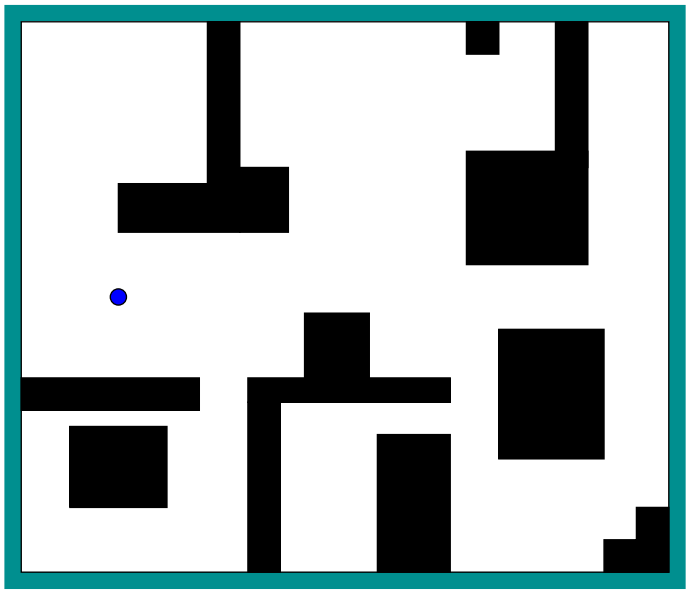


Figure 1.19: Consider solving tasks such as navigation, localization, mapping, and coverage in an indoor environment using a Roomba-like robot (recall the robot from Figure 1.12).

These days we have the option of developing mobile robots across the whole spectrum. We can employ powerful sensors and construct large models of the world. This leads to a big-brained robot that may require powerful computation systems. In this case, we imagine that the brain is a fully fledged computer with abundant memory. We can alternatively try to make the sensing as cheap and simple as possible, leading to a small-brained robot. The brain could feed sensor outputs directly to motors, or use some minimal amount of logic circuitry to maintain I-states. Many other possibilities exist in between. The brain could even represented without digitization as a continuous-time differential equation that transitions over a continuous space of I-states. The exciting challenge is to start from the task specification and design the right brain for the job.

Such "brain design" issues should become clearer while studying this book. Here is an example that is one step closer to reality than the one-dimensional robot from Section 1.3. Figure 1.19 depicts a mobile robot that rolls along the floor in a home or building. Here are some typical tasks that we might ask a mobile robot to perform:

- *Navigate:* Go to a prescribed location.

- *Localize:* Determine the robot's position and orientation (the direction it is

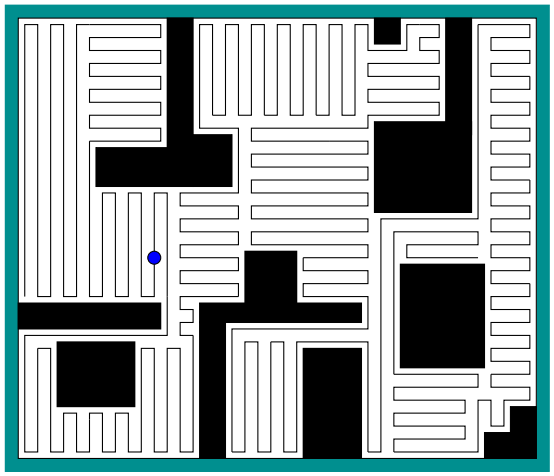


Figure 1.20: A big-brain approach to covering the whole floor area. Build a map. Determine the robot location and direction in the map. Compute a path that will complete the task. Command the robot to follow the path using powerful feedback from sensors.

facing.

- *Mapping:* Construct a map of the area accessible by the robot.
- *Patrolling:* Travel forever along a route that visits a prescribed set of places.
- *Searching:* Move around until a sensor detects a special spot.
- *Coverage:* Ensure that no part of the floor has been left untouched by the robot.

Each of these tasks has subtle variations that dramatically affect the sensing and information requirements. For navigation, how is the prescribed location specified? If specified as x-y coordinates, then it seems hard to avoid having to sense coordinates. For a robot with a camera, it might be specified as "move to a position where you can see the clock on the wall". In that case, it need not know its position in x-y coordinates to solve the ask. Similarly for localization, the robot might need to determine exact coordinates, or something weaker, such as determining which room it is in. For mapping, there are many ways to encode the map. A bitmap could indicate where the walls and floors are at a high resolution. A set of polygons could precisely encode the walls, making the representation closer to an architectural floor plan. Alternatively, a kind of *topological* map could be built, which indicates high-level information, such as the arrangement of rooms.

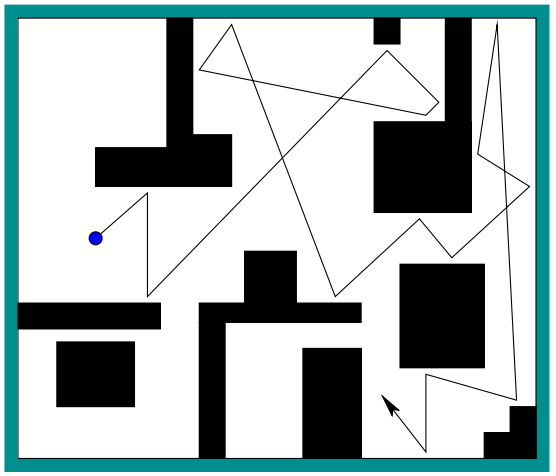


Figure 1.21: A little-brain approach implements a simple strategy of moving straight and then "bouncing" from the wall each time according to some prescribed law. One bit of memory is sufficient.

The patrolling and searching tasks were solved in Section 1.3 by the bit brain robot for the one-dimensional world. The coverage task is motivated by common chores such as mopping, vacuuming, and painting. Consider the following two robot designs:

1. *Big brain:* Provide a complete picture of what is happening in the physical world at all times. Design a systematic strategy for covering the whole floor. See Figure 1.20.

2. *Little brain:* Move straight until a wall is hit. Rotate a random amount and move straight again. Repeat forever. See Figure 1.21.

The big-brain robot stores a perfect map of its environment, giving precise coordinates of all wall edges and corners. How is this obtained? Either it is measured in advance and stored in memory, or it is learned from a mapping phase in which the robot explores all of its environment and pieces together its sensor observations. To accomplish this, powerful sensors are used that can measure the distance from the robot to the walls in all directions. At the same time, the robot localizes itself with respect to its map so that its position and the direction it is facing is always known. Once the mapping phase is complete, the brain then uses a *planning algorithm* to compute a detailed zigzagging route through the map. Following this computation, the robot is commanded to move along the path while using its sensors to correct for any deviations from the planned path during execution. At any given time during execution, the I-state corresponds

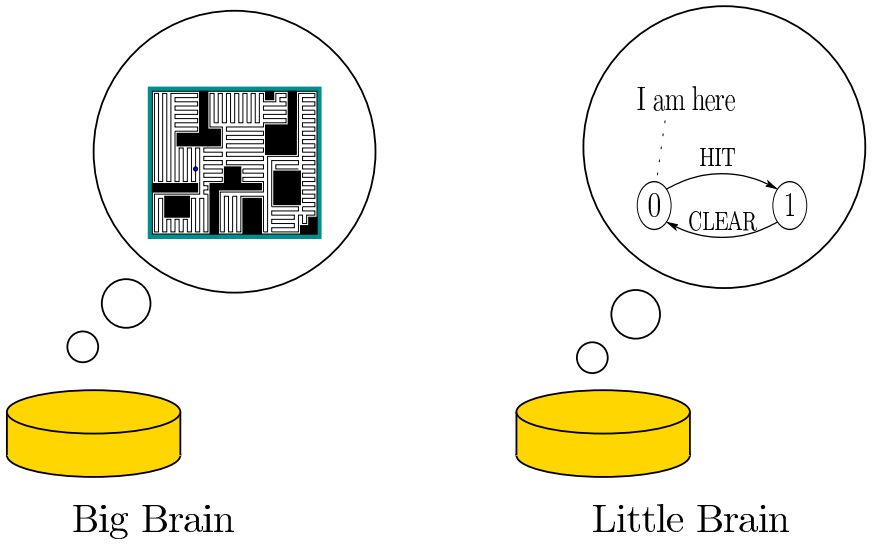


Figure 1.22: What robots are thinking. The big-brain robot maintains a full map of the environment and its own location in the map. The little-brain robot switches between only two I-states.

to the map and the robot's position and direction in the map. The space of all possible I-states, especially if the map is learned during execution, is enormous because it includes all possible maps, positions, and directions.

The small-brain robot is much simpler, operating with roughly the same level of awareness as the one-dimensional robot of Section 1.3. There are two I-states and the strategy is

I-state	Command
0	FORWARD
1	SPIN

The robot starts in I-state 0 and is commanded to move FORWARD. A bumper on the front of the robot holds a sensor that detects when the wall is encountered, producing the observation HIT. This causes a transition to 1 in which the SPIN command is given. The robot rotates at constant angular speed for a random amount of time, at which point a CLEAR observation is sensed. This causes a transition back to I-state 0, and the robot moves straight again. Some subtleties regarding SPIN and CLEAR have been neglected and will be addressed in more detail in the book. For now, imagine having a separate circuit that chooses a time t_{rand} from the interval $(0, t_{max})$, in which t_{max} is a long enough time to ensure that the robot could rotate 360 degrees in that time. Once time t_{rand} is reached (while the robot is rotating), the CLEAR signal is sent to the brain. See Figure 1.22.

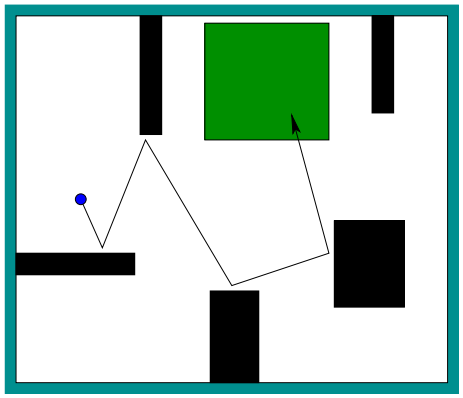


Figure 1.23: A simple strategy may be best for a hitting a large spot in an uncluttered environment.

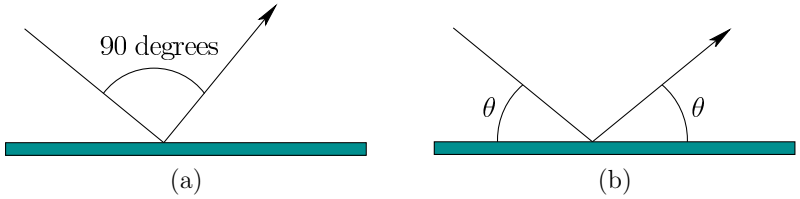


Figure 1.24: Simple, predictable rules for "bouncing" off of the wall: (a) Right angle bounce. (b) Specular reflection (as in light rays hitting a mirror).

The probability that the small-brain robot will reach every place in its environment actually converges to one, which is quite amazing for such a simple brain and simple sensors. However, it will clearly take much longer for full coverage in comparison to the big-brain robot. Furthermore, the small-brain robot cannot detect when the task is complete. The big-brain robot immediately knows it is done when the end of the planned path is reached. On the other hand, the small-brain robot could solve the searching problem in this environment and be able to detect task completion by introducing the sensor that observes FOUND. By a slight change in the task, the small-brain robot becomes more appealing. If the spot on the floor covers a large area and the environment is less cluttered, then the poor efficiency would no longer be a concern because the robot would be likely to hit it quickly (Figure 1.23). Furthermore, if you are bothered by the probabilistic aspects of the strategy, an effective solution can even be made by rebounding from the wall with a fixed bouncing rule (Figure 1.24). Examples include rebounding at 90 degrees from the incoming angle and rebounding using the laws of specular reflection (imagine a light ray hitting a mirror). In this case, absolute guarantees

can be made on solving the searching and coverage problems.

The remainder of this book is organized as follows. Chapter 2 explains how to make robots move by issuing commands, as we have talked about in this chapter. Some of the details include models of rolling, other locomotion methods, kinematics, and dynamics. At that point, the robot is totally blind, but models are used to predict where it goes when a command is issued. To enable robots to "see", Chapter 3 introduces sensing, from explaining many widely available sensors to developing mathematical models of the information provided by sensors. Putting Chapters 2 and 3 together leads naturally to Chapter 4, where motion commands are issued directly from the sensor observations. Interesting and useful robot behaviors emerge from this, such as wall following and collision avoidance.

It soon becomes clear that robots should have some memory, rather than basing their commands entirely on current sensor observations. This leads directly to Chapter 5, which explains how to transform the history of all previous sensor observations and commands that were issued into I-states that are useful for solving tasks. The result is a *filter*, which expresses precisely how I-states change each time new observations are received or commands are issued. Once filters are available, many tasks can be solved in terms of their I-states. Chapters 6 and 7 introduce the localization and mapping tasks, respectively. Simplified versions of these were mentioned in this section for the coverage task. Strategies are developed as functions (or IF-THEN statements) over the information space. For each topic, various versions exist along the spectrum from a smaller brain to a larger brain. Chapter 8 explains various forms of robot navigation in terms of information spaces. Finally, Chapter 9 considers other interesting tasks, such as coverage, tracking a target, playing hide and seek, and manipulation.