A THEORY FOR COMPARING ROBOT SYSTEMS

BY

JASON MATTHEW O'KANE

B.S., Taylor University, 2001
M.S., University of Illinois at Urbana-Champaign, 2005

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

To everyone who has loved and

supported me along the way.

Acknowledgment

Completing this thesis would never have been possible without generous assistance from many different people. I gratefully acknowledge the mentoring of Steve LaValle, who acted as a powerful catalyst for my development as a researcher and provided inestimably valuable advice along the way. I also offer sincere thanks to my colleagues Peng Cheng, Hamid Chitsaz, Steve Lindemann, Benjamín Tovar, and Anna Yershova for critiquing my ideas and pushing me to do my very best work. My committee members Mike Erdmann, Jeff Erickson, David Forsyth, and Rob Ghrist offered helpful guidance and unique perspectives on my work. Many other teachers and professors, including Leon Adkison, Felix Aguilar, Stefan Brandle, Tim Diller, George Lepsch, Richard Malay, Anne Marie Nucci, Frau Diana O'Donnell, Janet Robb, Wally Roth, Bill Toll, and Art White, have had immense influence on my development as computer scientist and as a person. Thanks for the unique contributions you each have made to my life.

More personally, I am thankful for the support I have received from my parents, other family, and my friends throughout my time in graduate school. Thanks to Greg Mackey, Naomi Caldwell, Abby Rigdon, Abe Millar, Steve and Lynn Rigdon, Karin Knapp, and Blake Johnson, who have each been friends to me in various ways and at various times.

Finally, it would be impossible to fully describe the steadfast patience and loving encouragement I've received from my wife, Heather. Thank you.

Financial support for this work was provided by DARPA under awards HR0011-05-1-0008 and HR0011-07-1-0002, by ONR under award N00014-02-1-0488, by NSF and CONACyT under award 0296126, and by a Roy J. Carver fellowship.

Abstract

As robots interact with the physical world, their usefulness depends directly on how effectively they can sense and move through their environments. Unfortunately, sensors provide only limited (and sometimes incorrect) information. In extreme cases, no sensors at all will be available. Therefore, for robots to be useful, they must act effectively in spite of uncertainty about the current state. This reality motivates a careful study of the *information requirements* of the problems we intend to solve. What sensing and actuation abilities are needed to complete a given task? Are some robot systems provably "more powerful," in terms of the tasks they can complete, than others? Can we find meaningful equivalence classes of robot systems? This thesis presents two related lines of research that make progress toward answering these questions.

First, we introduce a new technique for comparing the power of robot systems based on how they progress through their information spaces, which encapsulate the robots' state uncertainty. The goal is to understand the relative power of different sets of sensors and actuators and to determine which of these sets enable the robot to complete its task. This line of research is inspired by the theory of computation, which has produced similar results for abstract computing machines. The central idea is a dominance relation over robot systems, formalizing the idea that some robots are more powerful than others. This comparison induces a partial order over the set of robot systems. We prove some basic properties of this partial order, show that it is directly related to the robots' ability to complete tasks, and give several examples.

Second, we apply these ideas to the problem of active, global localization for mobile

robots. Sensor systems of varying ability have been proposed and successfully used to complete this task. We probe the lower limits of this range by describing three extremely simple robot models and addressing the active localization problem for each. The robot, whose state includes its position and orientation, moves in a fully known, simply connected polygonal environment. We pose the localization task as a planning problem in the robot's information space. We consider robots equipped with (1) angular and linear odometers, (2) a compass and contact sensor, and (3) an angular odometer and contact sensor. We present localization algorithms for models 1 and 2 and show that no algorithm exists for model 3. An implementation with simulation examples is presented. These three results, combined with the comparison results mention above, allow us to fully classify a set of 15 robot models according to their ability to complete the localization task.

Table of Contents

List of Figures

List of Tables

List of Symbols

Chapter 1

Introduction

These are exciting times in robotics. After decades of research, robot technology is poised to have profound near-term impact on many aspects of society. Evidence of this promise includes:

1. The growing popularity and mainstream commercial availability of household robots like iRobot's Roomba[1] (see Figure 1.1) or the RoboMower from FriendlyRobotics.[2] As prices decrease and functionality is enhanced, such household robots can be expected to integrate more completely into everyday life.

2. The DARPA Grand Challenge races, in which 5 different vehicles navigated autonomously across a 212km desert course [35, 110]. Such autonomous vehicles could drastically reduce the need for human presence in some dangerous military contexts. The upcoming Urban Challenge.[3] may result in similar progress for autonomous driving on congested urban settings and eventually lead to improved highway safety for civilians.

3. The increasing robustness and autonomy of humanoid robots such as Honda's Asimo.[4] Since their physical characteristics mimic those of humans, humanoid robots are well-suited to operation in human living spaces. Such robots may soon be used to provide assistance and companionship to the elderly.

[1] http://www.irobot.com
[2] http://www.friendlyrobotics.com
[3] http://www.darpa.mil/grandchallenge/
[4] http://asimo.honda.com/

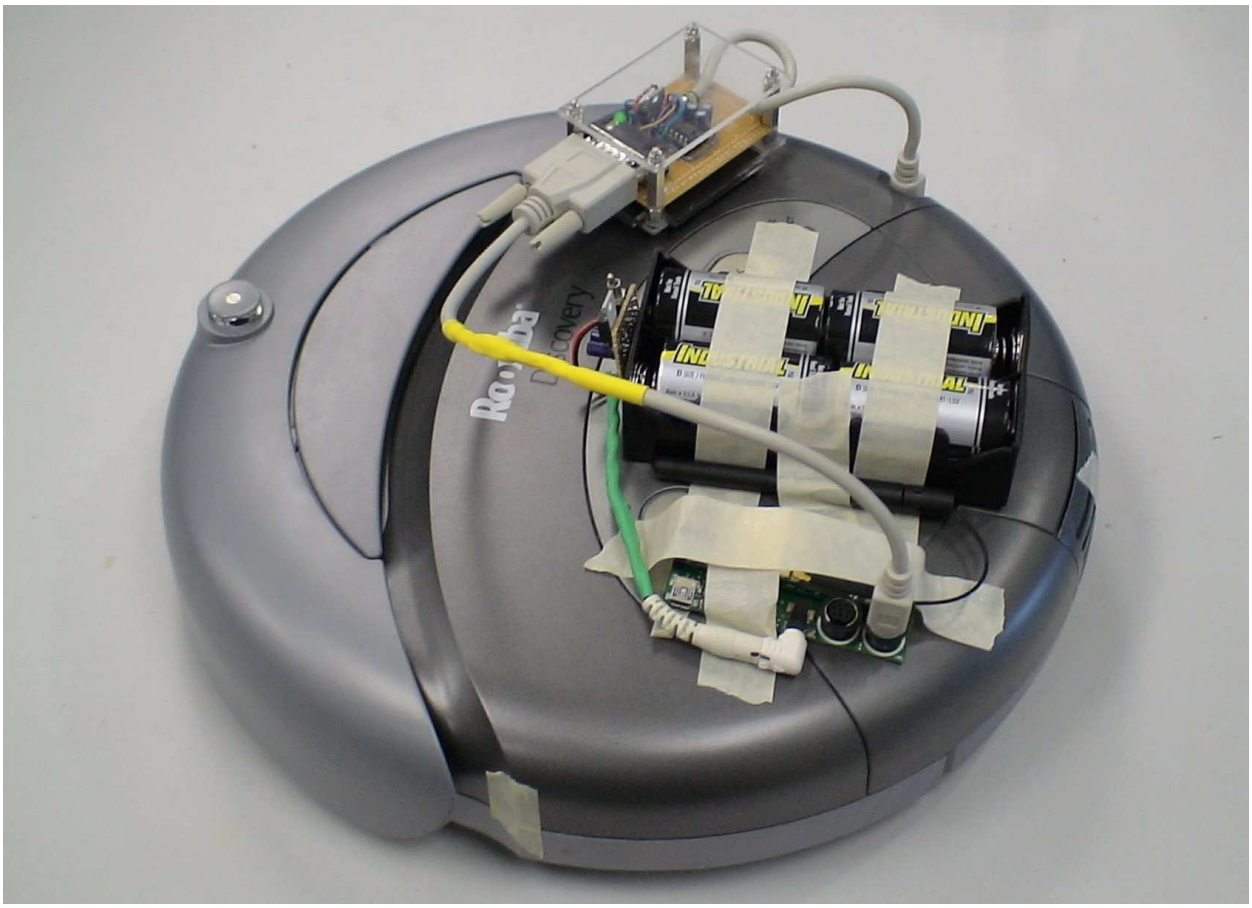


Figure 1.1: A Roomba robotic vacuum cleaner. Roombas inexpensive and commercially available, but their sensing capabilities are very weak.

These examples suggest that, for the first time, truly autonomous robots are becoming practical. In addition, general-purpose software tools such as the Microsoft Robotics Studio[5] and the Player/Stage framework[6] along with the maturing collection of prebuilt research platforms such as the Pioneer[7] and Khephera[8] suggest that experimental robotics research will accelerate even more. Planning techniques developed in robotics are even having increasing impact on manufacturing [13, 101], animation [100], and computational biology[41].

[5] http://msdn.microsoft.com/robotics/. See also [69].
[6] http://playerstage.sourceforge.net/
[7] http://www.activrobots.com/
[8] http://www.k-team.com/

1.1 The challenge of autonomy

In spite of this potential, autonomous robots are not yet widely deployed. What are the remaining roadblocks? Loosely defined, a robot is a device that couples computation of some kind with some type of direct, substantive interaction with the physical world. The distinguishing feature of robots, therefore, in comparison to other computing machines, is that is that robots interact purposefully with the physical world. *This reality presents a major challenge because the physical world is unstructured, unpredictable, and complex.* Consequently, we claim that finding effective ways to collect and act upon information about the external world will be crucial for robot technology to continue its advance.

Historically, most approaches to dealing with the challenge of interacting with the physical world fall into one of two general categories. The first approach attempts to skirt the issue by limiting the robot to operate only under closely controlled conditions. In some cases, the environment is modified (or even designed from scratch) specifically for a certain task. This approach has been especially successful in industrial settings, but its usefulness is ultimately limited to contexts where the environment is very well-modeled and very predictable. A more complete criticism of this approach appears in [161].

Other systems attempt to deal with a larger subset of the world's complexity using elaborate sensor systems. A typical example might include some combination of laser range sensors, sonars, cameras, GPS receivers, wheel and joint encoders, and other sensors. Although such sensors are capable, in principle, of providing a wealth of information about the environment, they also increase the cost and energy consumption of the resulting system. Dealing robustly with noise in these sensors also increases the modeling burden and can drastically increase the complexity of the system's software. Mason gives an insightful (and amusing) critique of this approach in [117]. On a more fundamental level, designing systems with an excess of sensors generally sheds little light on which of those sensors provides information that is necessary, rather than merely sufficient, to complete a given task.

In this thesis, we argue for a third approach characterized by robots that interact with their environments using a small set of sensors that provide only limited information. This approach is motivated by at least three potential benefits. First, it may lead directly to simpler robot designs and algorithms for some tasks. Second, we expect to gain a better understanding of robotic problems by understanding the information requirements of those problems – the conditions on sensing under which those problems can and cannot be solved. Third, knowledge of how to solve relevant problems in spite of sensing limitations can be useful in failure modes of more complex robots, particularly for cases, such as space robotics, in which the deployment cost is very high.

In the remainder of this chapter, we review the main ideas that guide the work (Section 1.2) before previewing the primary results and structure of the thesis (Section 1.3).

1.2 Core ideas

The arguments of this thesis are built on the philosophy that sensing and uncertainty are core, defining issues in robotics and that the key to understanding robotics problems is to understand how these problems can be solved when sensing is limited and uncertainty is great. This philosophy leads us to explore three interrelated themes in this work.

1.2.1 Information spaces

Many existing methods in robotics focus on the robot's progression through a space of states, assuming that the robot has full knowledge of the current state at all times during its execution. What happens when the robot has insufficient information to determine its state? One approach is to use *state estimation*, in which the robot uses the information available to it to make an "educated guess" about its state. The robot can treat this estimated state as its true state and ignore the uncertainty. In some extremely limited contexts, this is

provably optimal. (See, for example, Section 6.1 of [22].) For many tasks, however, accurate state estimation is impossible.

The central technical tool we use to account for this difficultly is the *information space*, which naturally encodes the robot's knowledge. Rather than attempting to estimate the current state, the robot instead relies on its history of actions and sensor readings. These histories, together called the robot's *information state*, are always fully known. The twofold challenge is to devise efficient ways to represent, update, and query this information, and to develop plans defined in terms of this information state, rather than the robot's unknown true state.

Information spaces are analogous to the configuration spaces that arise in mechanics and classical motion planning, in the sense that they provide a unified way of approaching many different problems. Therefore, we claim that a deep understanding of the information space is essential to effective methods for planning in the presence of uncertainty. Information spaces in various forms appear in game theory [16, 94], control theory [22, 95], artificial intelligence [124, 138, 167], and robotics [52, 53, 62, 73, 111]. Unfortunately, the information spaces themselves, particularly as they arise in robotics, are not yet well understood.

Note also that the raw sensor and action histories recorded by the robot's history I-state usually are not directly informative. This motivates our study of *information mappings* into *derived information spaces*. These mappings, in their most general form, were introduced by LaValle in [102]. A derived information space can be viewed as a "compression" or "interpretation" of the histories. Such a mapping could be based on probabilistic or nondeterministic models of the robot's sensors and environment. Other more drastic or problem-specific mappings are also possible. The choice of an information mapping determines the conclusions the robot can make about its state and indirectly determines the problems the robot can solve. Consequently, the choice of an information mapping and a derived information space are crucial modeling decisions.

1.2.2 Minimalism

Both sensors and actuators are subject to significant errors in precision and accuracy. Effective robots must be robust to these errors. Starting, perhaps, with Whitney's critique of mid-1980's robotics research [161], an approach has arisen in which these difficulties are dealt with by designing extremely simple robots that exploit the conformant properties of the system to complete their tasks. In industrial settings, complex tasks can be solved by sequences of these simple robots [37]. Other work has explored the more general question of the minimal sensing requirements to complete a given task [29, 52, 63]. This *minimalist* approach, which we take throughout this thesis, has been applied to several different kinds of robotics problems.

Many effective systems for manipulation have used a minimalist approach. Akella and Mason [6] give a complete planner for pushing objects on a planar surface while avoiding obstacles. A broader focus of research has been on part orienting systems, in which parts with unknown initial orientation are manipulated into some known final orientation. This has been accomplished with limited sensing using tilting trays [59, 60, 74, 123], parallel jaw grippers [72, 73], vibratory bowl feeders [4, 21, 31], linear pushes [7], and active [5, 7] or passive [36, 162] fences over conveyor belts. These methods are surveyed in [158]. For some of these cases, the problem of planning to orient parts can be reduced to that of finding a sequence that resets a finite state machine from an unknown initial state to a known final state [58]. Another approach is to consider carefully the effects of initial conditions, for example in the context of dropping parts onto a designed surface [121]. More generally, the preimage planning framework [111] has been used for manipulation planning under uncertainty [61, 63].

Others have considered certain navigation and exploration tasks for mobile robots with limited sensing. An analysis of the basic requirements for navigation in an unknown three dimensional environment appears in [96, 97]. Bug algorithms [87, 88, 114, 115] and related

methods [27, 51] are used for navigation by robots capable of moving toward their goals and discovering obstacles, usually by coming in contact with them, along the way. In [156, 157], the robot has an extremely crude range sensor that can only detect discontinuities in depth information. As the robot explores its environment, this information is used to construct a data structure that allows for optimal navigation between previously visited locations. More explicit maps based on metric measurements can be built with a range sensor by traversing the generalized Voronoi graph of the environment boundaries [2, 3, 43, 44, 125]. Another approach is to assume that the robot can move reliably only when it is near certain known landmarks. Lazanas and Latombe give a method for navigation under such constraints [104]. The robot model used in [164] is even simpler, capable only of following walls and "jumping" across the interior of the environment at reflex vertices of the environment boundary. Many of these results are surveyed in [135].

More abstractly, one may think of the universal traversal sequences that arise in graph theory [8, 9] as a minimalist approach to the coverage problem. Let g denote a d-regular graph. For each vertex of g, we may bijectively label the incident edges with the labels $\{1, \ldots, d\}$. Fixing a start vertex v, a string $s \in \{1, \ldots, d\}^*$ can be considered a path in g by following edges in the indicated order. If s visits every vertex in g, we call s a *traversal sequence* for g starting at v. Now consider the family $\mathcal{G}(n, d)$ of all connected n-vertex d-regular graphs. A sequence s is a (n, d)-*universal traversal sequence* if s, for each $g \in \mathcal{G}(n, d)$ and each $v \in g$, s is a traversal sequence for g starting at v. Universal traversal sequences can be considered as solutions to planning problems with uncertainty both in environment space (that is, the selection of g from $\mathcal{G}(n, d)$) and in state space (that is, the selection of a start vertex $v \in V(g)$). More concretely, observe that $\mathcal{G}(n, 4)$ contains all grid-like environments in the plane with n unoccupied cells, so that an $(n, 4)$-universal traversal sequence will visit every square of any n-element planar grid. Borodin et al. [33] give several lower bounds on the lengths of universal traversal sequences. In [10], the problem is addressed for complete graphs. Bounds for other special cases appear in [38, 85, 151].

This collection of work can be interpreted in at least two different ways. At the surface level, it can be construed as a collection constructive proofs that certain tasks can be completed with limited sensing. More deeply, these results can be understood as revealing something about the information required for the tasks they address.

1.2.3 Feasible feedback planning

Most classical formulations of planning assume that the desired output is a sequence of actions that achieves a goal while minimizing some cost criterion. We consider formulations that differ in two important ways. First, since the sensor data received cannot, in general, be predicted ahead of time, we expand the notion of a plan from a sequence of actions to allow *feedback*. Such a plan takes the form of a function mapping from a derived information space into the space of actions available to the robot. Second, we emphasize *feasibility* over optimality. That is, we ask "Can the robot complete its task?" rather than "How efficiently can the robot complete its task?".

One informal way to view this approach is to envision some space of robot systems ranging from very simple to very complex, arranged so that robots with similar abilities are near one another. For a given task, some robot systems are capable of completing the task whereas others are not. Our goal is to search the space of robot systems for the boundary between the "can solve" and "cannot solve" regions. This boundary gives an indication of necessary conditions on robot models for that task. By neglecting the possibility of tradeoffs between the robot's capabilities and the quality of solutions that can be achieved, we create a crisp boundary for this solvability region.

1.3 Thesis overview

We conclude this introductory chapter with a preview of the remainder of the thesis. Basic definitions appear in Chapter 2. Chapters 3 and 4 contain original contributions, for which

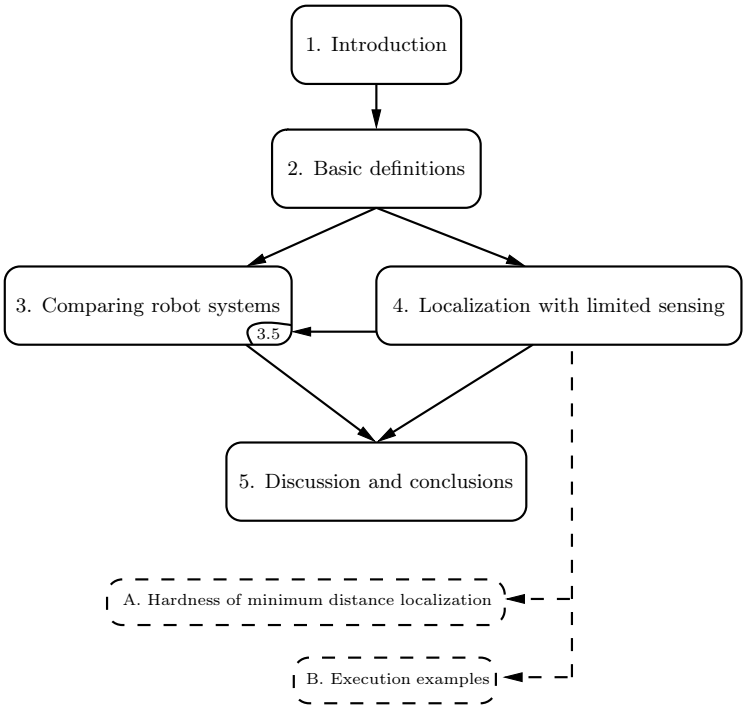


Figure 1.2: Organization of this thesis. Arrows indicate dependencies.

previews appear in Sections 1.3.1 and 1.3.2 respectively. We make some concluding remarks in Chapter 5. The structure and dependencies between chapters are shown in Figure 1.2.

1.3.1 Comparing the power of robot systems

The localization work described above identifies, in some informal sense, the part of boundary in a space of robot systems between regions containing robots that can complete the task from those that cannot. Can this notion of a space of robot models and a feasibility boundary within it be made more precise?

This line of inquiry is inspired by the theory of computation, which asks similar questions about its precisely defined, abstract models of computation. Can a given machine solve a given problem? If so, how efficiently? Are some machines strictly more powerful than others? Are there apparently dissimilar machines with provably equivalent computation power? In mainstream computer science, there are well-established formalisms (asymptotic

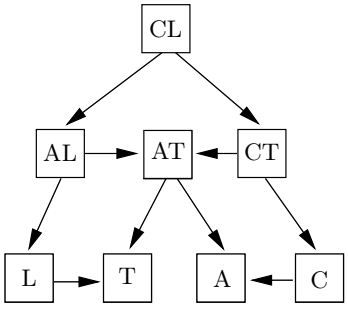


Figure 1.3: A dominance hierarchy for a collection of simple robot models. Arrow indicate that one robot model dominates the other, in the sense of Definition 3.3. Details are in Section 3.5.

analysis, complexity theory, formal models of computation, etc.) for measuring the difficulty of a problem and for assessing the effectiveness of solutions. Although these measures are sometimes at odds with contemporary practice – witness, for example, problems reduced to boolean satisfiability [91] in spite of the theoretical hardness of SAT [46, 109] – they are a universally accepted foundation for the study of algorithms. Unfortunately, standard models of computation are fundamentally ill-suited for robotics problems, in which sensing and uncertainty are unavoidable. As a result, current robotic science lacks a unified theory in which meaningful statements can be made about the complexity of robotic tasks and the robot systems we build to complete these tasks.

This thesis begins to address this weakness by presenting an analysis technique for comparing robot systems. The result is a formal definition of dominance of one robot system over another. This definition accounts for sensing and uncertainty by considering how the robots move through their information spaces. The intuition is that one robot dominates another if the former can "simulate" the actions of the latter in a certain way. This dominance relation is useful because we show that it is consistent with the ability of robots to complete tasks: If R_2 dominates R_1, then with certain technical conditions, R_2 can complete every task that R_1 can. Moreover, if the robots are defined as sets of independent robotic primitives, several results about combinations of these primitives follow that are reminiscent

of the axioms of rationality in decision theory. By combining these techniques with the localization results discussed below, we grouped a collection of 15 simple robot models into 8 equivalence classes, determined the dominances between these classes, and classified each with regard to its ability to complete the localization task. See Figure 1.3. This work is presented in Chapter 3.

1.3.2 Localization with limited sensing

Consider a mobile robot with access to map of its environment, but ignorant its position within that environment. The problem of global localization is to command the robot so that, at the end of its execution, the position uncertainty has been eliminated and the robot knows its location. Many different sensing modalities are available for mobile robots (Figure 1.4), but little is known about the necessary conditions for solving the problem. How simple a robot can complete this task? What are the simplest collections of sensors that enable the robot solve the problem?

This thesis presents results in an idealized setting with a point robot, a polygonal map, and perfect control. In this context, a robot with only a contact sensor and a compass is able to solve the global localization problem (Figure 1.5), but a similar robot with only angular odometry rather than a compass cannot. If the contact sensor is augmented with an odometer for measuring translations, the problem is once again solvable. In combination, these results give a rough indication of the "feasibility surface" that divides robots that can complete the task from those that cannot. We also describe adaptations of these techniques to an experimental setting using probabilistic reasoning. This work is presented in detail in Chapter 4.

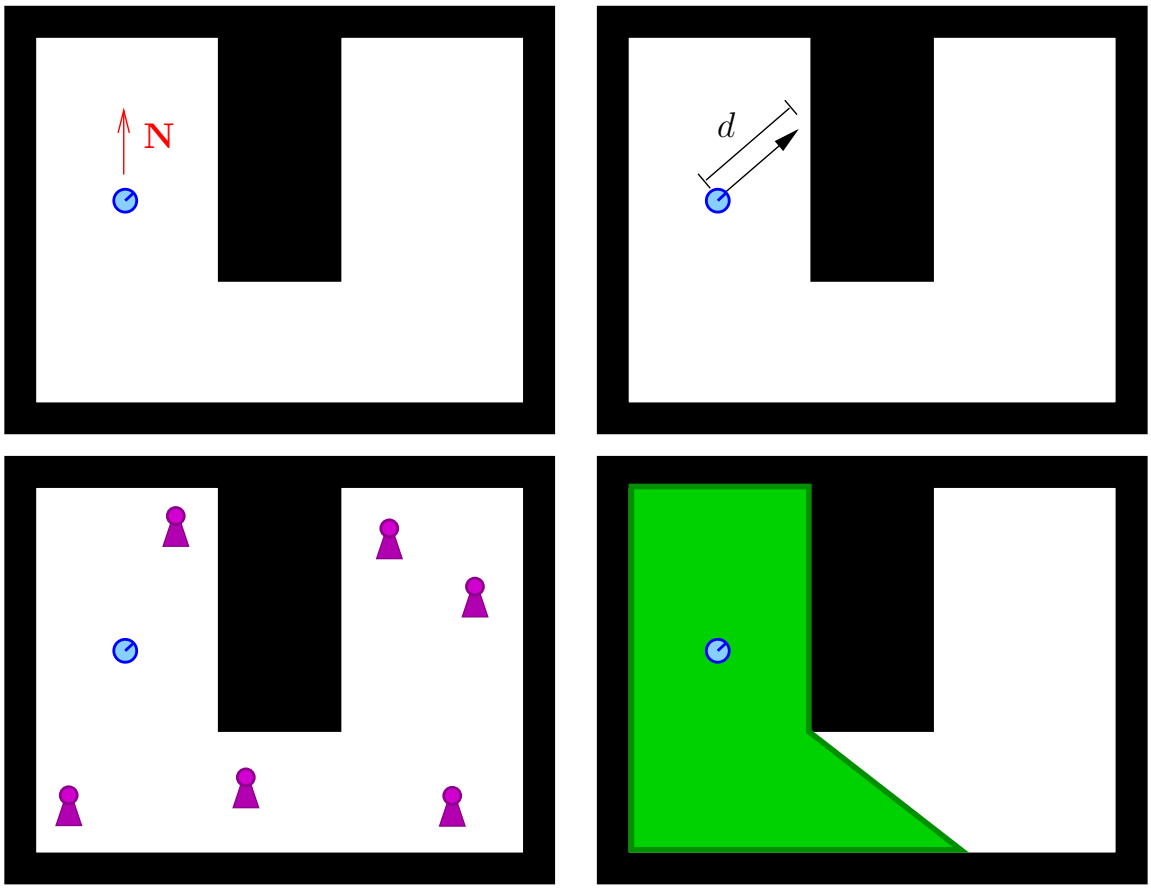


Figure 1.4: There are many options for mobile robot sensing. [top left] A compass reports the robot's orientation with respect to a global reference direction. [top right] A linear odometer measures the distance the robot travels. [bottom left] A landmark detector identifies relevant features of the environment. [bottom right] A range sensor measures the distance to nearby obstacles.

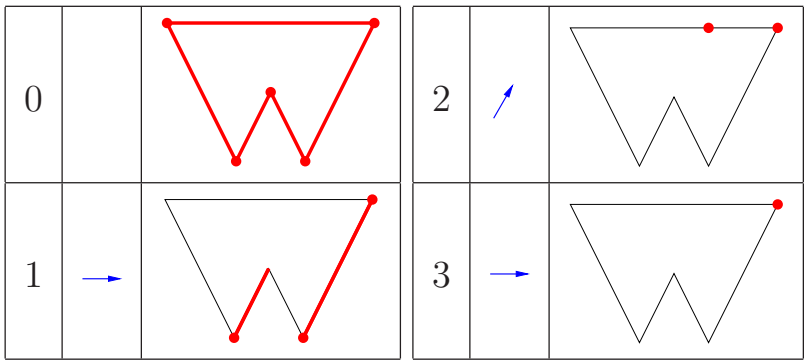


Figure 1.5: A localizing sequence generated by Alg. 4.2. A robot starting with no information about its position localizes itself to the top right corner of the environment. Possible states after each motion are shaded. Details of the robot model and the algorithm appear in Chapter 4.

Chapter 2

Basic definitions

All of the problems considered in this thesis can be classified as planning problems with uncertainty in the current state. In this chapter, we present a general formulation for such problems. These definitions give a common foundation to the remainder the thesis. We describe the most basic model in Section 2.1. In this model, which is depicted in Figure 2.1, a robot affects its environment by executing *actions*. In response to those actions, the robot receives *observations* that provide information about the current situation in the world. This basic structure is found throughout the literature in control theory [48], artificial intelligence [98], and elsewhere [113]. Our notation is largely borrowed from control theory. Next, we define the notion of an *information space*, a concept essential to explicitly managing state uncertainty. By planning in the information space, we sidestep the problem of state uncertainty. The tradeoff is that the information space is generally much larger and more complex than the underlying state space. Definitions and notation for information spaces appear in Section 2.2. Finally, we use these concepts to define general notions problems and solutions in Section 2.3.

2.1 Basic ingredients

We begin by defining a few important spaces and describing the relationships between them.

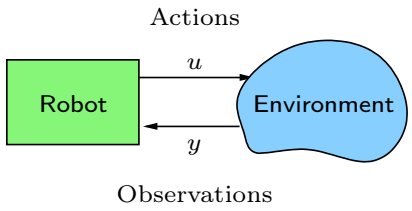


Figure 2.1: A robot interacts with its environment by executing actions and receiving observations.

2.1.1 The state space

The robot moves in a *state space* X. To maximize the generality of the model, we make no assumptions about X, except that a single state $x \in X$ describes the situation of the world in sufficient detail to model all problems of interest. One typical choice is the robot's configuration space [112].

Example 2.1 (Point robot in the plane) *Many problems we consider in this thesis will be for mobile robots with orientation in the plane. In this context, we use the configuration space* $X = E \times S^1$*, in which* $E \subset \mathbb{R}^2$ *is the robot's environment and* $S^1 = [0, 2\pi]/\sim$*, where* $\sim$ *is an equivalence relation identifying* 0 *and* 2π*, represents the robot's orientation. See Figure 2.2.* ◇

In other cases, a more complex state space is needed. For example, formulations like those of Examples 2.1 silently assume that the robot operates in a single environment whose geometric details are known *a priori*. What happens when the robot begins with limited or no knowledge about its environment, in the sense that positions and geometry of obstacles, map topology, navigability of terrain, and so on are unknown? Such imperfect knowledge about the environment is a more drastic instance of the general issue of state uncertainty. If the state is defined to include a description of the environment in addition to the robot's configuration, then uncertainty in the environment can be represented as an additional dimension of state uncertainty.

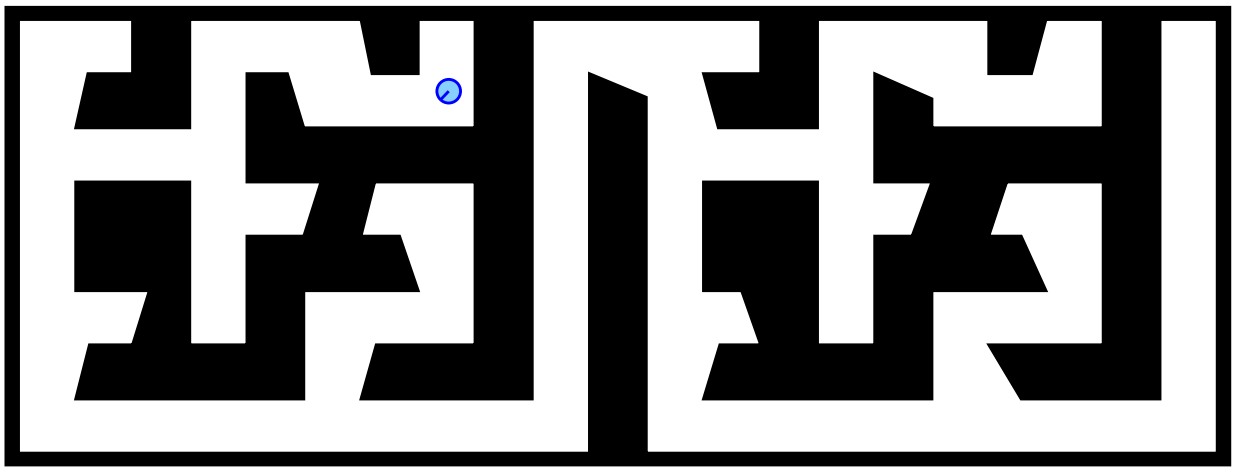


Figure 2.2: A robot in a planar environment E. Its state space is $X = E \times S^1$.

Example 2.2 (Environment space) *Choose an* environment space $\mathcal{E}$ *of which each element* $E \in \mathcal{E}$ *is a potential environment for the robot. The environment* E *is unknown, but remains fixed throughout the robot's execution. Possibilities for* $\mathcal{E}$ *with varying degrees of realism, interest, practicality, and amenability to analysis, include:*

1. *the set of bounded planar grids with occupancy maps,*
2. *the set of simple polygons in the plane*
3. *the set of compact regions in* $\mathbb{R}^2$ *or* $\mathbb{R}^3$ *with connected interiors and piecewise analytic boundaries, and*
4. *the set of terrain maps from* $\mathbb{R}^2$ *to* $\mathbb{R}$*, giving the elevation or navigability at each point in the plane.*

The state space is formed by combining the robot's configuration space $\mathcal{C}$ *with* $\mathcal{E}$*, so that* $X = \mathcal{C} \times \mathcal{E}$*. See Figure 2.3.* ◇

In defining the state space, it may be important to make a clear distinction between the state of the robot (its configuration along with other variables that describe its internal

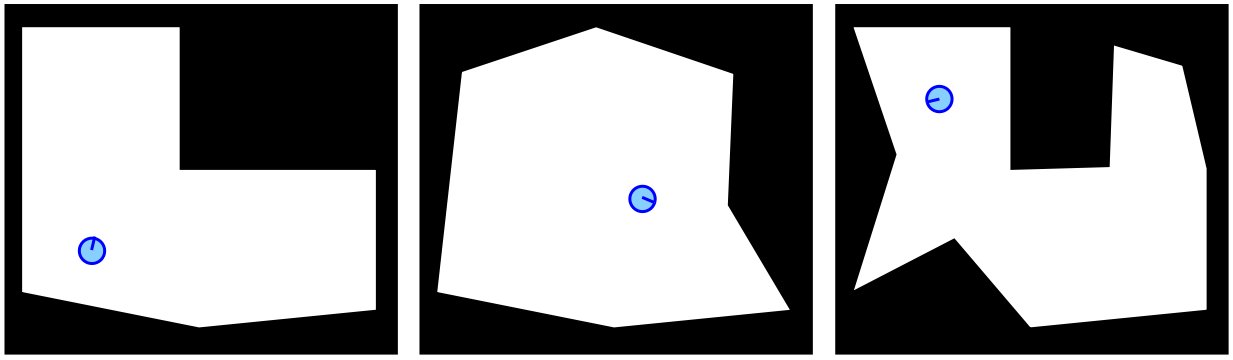


Figure 2.3: Three states of a system modeling a mobile robot in the plane with environment uncertainty. When the environment is uncertain, the identity of the environment becomes part of the state of the system.

state) and the external state of the world (for example, configurations of other agents or manipulable objects within the environment).

Example 2.3 (Factored state space) *Suppose a robot is charged with a delivery task, in which a collection of n manipulable objects must be moved about within the environment. Model this situation with a state space factored into the robot state $\mathcal{C}$ and the world state $\mathcal{M}$, and let $X = \mathcal{C} \times \mathcal{M}$. In this way, the state of the robot is kept distinct from the state of the complete system. Figure 2.4 shows an example in which $\mathcal{C} = E \times S^1$ and $\mathcal{M} = E^n$. We revisit this type of formulation in Example 3.13 in the context of analyzing the power of robots. This decomposition will allow the comparison to be made independent of configurations of the robots themselves.* ◇

These examples represent only a small sampling of the issues to be considered in selecting a state space. Other state spaces are also quite reasonable to consider, including the phase spaces that arise in the analysis of dynamical systems. Ultimately, the choice of a state space is a subjective process driven by concerns of modeling and abstraction.

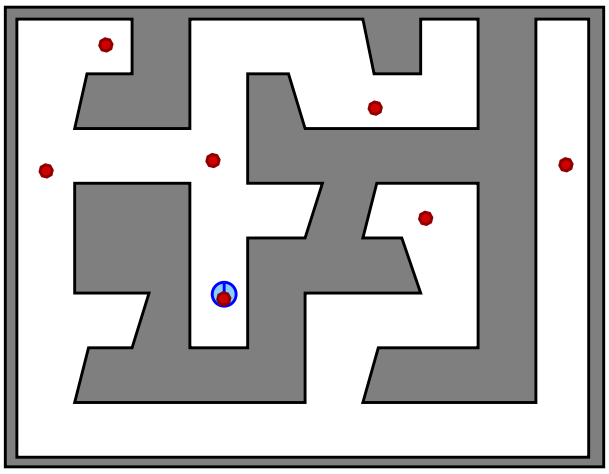


Figure 2.4: A robot in a planar environment E, along with a collection of 7 movable objects. The state space of the complete system is $X = (E \times S^1) \times E^7$.

2.1.2 Actions and transitions

The robot influences its state by selecting *actions*. The *action space* U represents the set of actions available to the robot. As with the state space, we minimize as much as possible the assumptions made on U. We do, however, assume that same actions are available at every state. Under another reasonable formulation, the actions at each state x are limited to a subset $U(x) \subseteq U$ of the complete action space. Such limitations make sense, for example, at the physical limits of the machine or when the robot is in contact with an obstacle. Ultimately, of course, the robot would need to use some form of sensing – either proprioceptive or exteroceptive – to determine that this is the case. Therefore, in this thesis we model this phenomenon directly through observations (defined in Section 2.1.3) and define state transitions accordingly.

Throughout most of this thesis, we divide time into discrete *stages*[1]. The sequence of stages is indexed by consecutive integers starting with 1. Each stage represents a time interval during which the robot makes only a single decision. As such, the stages need not

[1]Continuous models of time certainly have a more direct correspondence with reality than any discretization. We consider continuous-time models in Section 3.6.2.

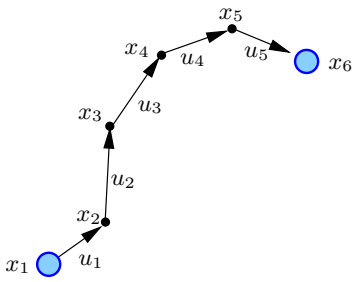


Figure 2.5: A simple system in which a robot's action directly specify the magnitude and direction of the robot's motion. A sample 5-stage execution is shown.

have equal physical duration. The robot's state at stage k is denoted x_k; its action at stage k is u_k. We describe the change made to the state by each action by way of a *state transition function*

$$f : X \times U \to X. \tag{2.1}$$

The relationship between states and actions as time progresses is given by

$$x_{k+1} = f(x_k, u_k). \tag{2.2}$$

An iterated version of f that applies several actions in succession will also be useful:

$$f(f(x, u_1, \ldots, u_k) = f(\cdots f(f(x, u_1), u_2) \cdots, u_{k-1}), u_k). \tag{2.3}$$

The next example is a simple realization of this kind of system.

Example 2.4 *Let $X = \mathbb{R}^2$, $U = \{u \in \mathbb{R}^2 \mid ||u|| < 1\}$, and $f(x, u) = x + u$. This system models a kinematic omnidirectional robot in a planar environment, where the actions specify a direction and magnitude of motion. See Figure 2.5. Note that to account for environment obstacles would require a more complex transition function.* ◇

We emphasize that f need not have a clean, closed-form representation. In Section 4.2, for example, we define several transition functions that depend directly on the geometry of

the environment. In an extreme case, one might imagine an f that is not even computable, although the implications of such a model are not immediately clear.

2.1.3 Observations

Next, we include in the model a notion of sensing by allowing the robot to receive *observations*. At the conclusion of each stage, the robot's sensors provide an observation y from an observation space Y, according to the observation function

$$h : X \times U \to Y. \tag{2.4}$$

Each observation can be viewed as providing a "hint" to the robot about its true state. The robot's observation at stage is denoted y_k and determined by x_k and u_k:

$$y_k = h(x_k, u_k). \tag{2.5}$$

Again we illustrate with a simple example.

Example 2.5 *Suppose a mobile robot in the plane has a sensor that detects the distance (but not the direction) to a landmark at a known, fixed position $p \in \mathbb{R}^2$. This situation can be modelled by setting $Y = \mathbb{R}$, with $h(x, u) = ||x - p||$. See Figure 2.6.*

Combined with the action space and transition function from Example 2.4, this forms a complete sensing-actuation system. ◇

An important special case of (2.5) occurs when the observation depends only on the current state, rather than on the action taken in the current stage. In this case, we can simplify the observation function to $h : X \to Y$, with $y_k = h(x_k)$. Models of this kind are of

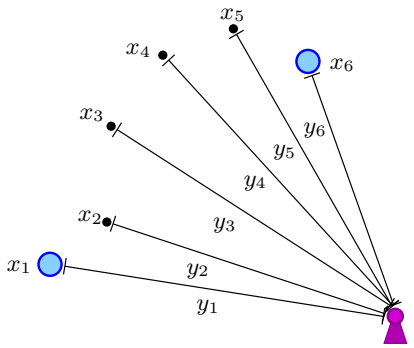


Figure 2.6: A robot that can sense the distance to a stationary landmark.

interest because the preimages

$$h^{-1}(y) = \{x \in X \mid y = h(x)\} \tag{2.6}$$

form a partition of X, indicating sets of states that are indistinguishable by a single observation. This is the view of sensing taken, for example, by [63] and elsewhere. We use a formulation in which the observations depend on actions to model faithfully situations where sensing must be *active*, that is, when the robot must explicitly query its sensors (perhaps at some cost), rather than passively reading from them as time passes. This subtlety is particularly important to the robotic primitives we introduce in Section 3.2.

Although we are assuming in this section that both state transitions and observations are deterministic, we acknowledge that in realistic contexts, managing unpredictability in motion and sensing is a crucial issue. We omit such uncertainty here because of the additional complications it would introduce. The extensions needed to relax this assumption are introduced in Section 3.6.1.

2.2 Information spaces

In the formulation presented in Section 2.1, the robot's true state is hidden, so the robot must make its decisions based on the limited information available to it. In this section, we

present the notion of *information spaces*, which are the most natural spaces for studying such systems. A much more complete treatment of information spaces and their use in robotics appears in Chapters 11 and 12 of [102].

2.2.1 The history information space

In the absence of perfect state information, what information is available to the robot? In our formulation, the robot has access only to the histories of actions it has selected and observations it has received. The space of such histories is the robot's *history information space* (history I-space), denoted $\mathcal{I}_{hist}$ and defined in terms of U and Y:

$$\mathcal{I}_{hist} = \bigcup_{i=0}^{\infty} (U \times Y)^i. \tag{2.7}$$

After the completion of stage k, the robot's history information state is a sequence of length $2k$ given by

$$\eta_{k+1} = (u_1, y_1, \ldots, u_k, y_k) \in \mathcal{I}_{hist}. \tag{2.8}$$

We occasionally abuse notation by writing (η_{k-1}, u_k, y_k) for the history I-state formed by appending u_k and y_k to η_{k-1}.

How is the state space related to the robot's history I-space? One connection is by way of the notion of states consistent with an I-state:

Definition 2.1 *A state $x \in X$ is* consistent with *a history I-state $\eta_k = (u_1, y_1, \ldots, u_k, y_k)$ if there exists some $x_1 \in X$ such that $x = f(x_1, u_1, \ldots, u_k)$ and $y_j = h(f(x_1, u_1, \ldots, u_{j-1}), u_j)$ for each $j = 1, \ldots, k$.*

The intuition is that the consistent states x_k are those for which there is some starting state from which executing the given action sequence would produce the given observation

sequence and leave the robot at x_k. The set of consistent states provides a concise way of describing the information available to the robot.

We may define a *policy* $\pi : \mathcal{I}_{hist} \rightarrow U$ over history I-space. Note that, given a state x_k and a history I-state η_k, the history I-states reached by repeatedly executing π are fully determined. As a shorthand, we define a function F that applies a policy several times in succession, so that m applications of a policy π, starting at state x_k and information state η_k, lead to a new history I-state given by

$$\eta_{m+k} = F^m(\eta_k, \pi, x_k). \tag{2.9}$$

Note that $F^m(\eta_k, \pi, x_k)$ depends on the true state x_k (which is unknown to the robot) because x_k influences the observation sequence the robot receives.

2.2.2 Information maps and derived information spaces

The history I-space is not particularly useful by itself, because it provides no insight into the conclusions that the robot might make based on the action and observation histories. Furthermore, the length of a history I-state grows linearly with the number of stages, potentially causing complications for storage and computation. For these reasons, we consider information mappings (I-maps) of the form

$$\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I} \tag{2.10}$$

that "compress" history I-states in some way. The target space $\mathcal{I}$ is called a derived information space (derived I-space). In an informal sense, κ indicates how the robot "interprets" its sensor information. An important special case is the value of κ for an empty history, which gives an *initial condition* for the robot. This initial condition reflects any knowledge the robot may have before its execution begins.

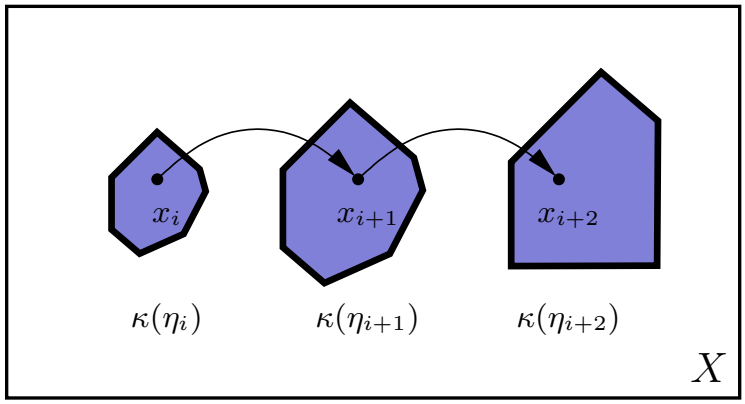


Figure 2.7: Under nondeterministic uncertainty, the derived information state is a subset of the state space, indicating a minimal set of "possible states".

In principle, we may select $\mathcal{I}$ and κ arbitrarily. The usefulness of a derived I-space depends on its ability to capture the information relevant to the task of interest. Example 2.6 presents a derived I-space that we revisit frequently.

Example 2.6 *In one useful derived I-space, the* nondeterministic I-space $\mathcal{I}_{ndet}$*, derived I-states are nonempty subsets of* X*. The derived I-state is the set of states consistent with the history I-state. The interpretation is that the robot's derived I-state is a minimal subset of state space guaranteed to contain the true state. The I-map* $\kappa_{ndet} : \mathcal{I}_{hist} \to \mathcal{I}_{ndet}$ *can be expressed recursively in terms of* f *and* h*:*

$$\kappa_{ndet}(\,) = X \tag{2.11}$$

$$\kappa_{ndet}(\eta, u, y) = \{f(x, u) \mid x \in \kappa_{ndet}(\eta), y = h(x, u)\} \tag{2.12}$$

See Figure 2.7.

Note the initial condition. In Equation 2.11, we assume the robot initially has no information about its state. One might form similar I-maps by using smaller sets for $\kappa_{ndet}(\,)$*, corresponding to situations in which the robot starts its execution with some knowledge about its state.* ◇

The next two examples show how other well-known approaches can be understood as particular kinds of I-maps.

Example 2.7 *Consider I-maps of the form $\mathcal{I}_{hist} \rightarrow X$. By mapping the history I-state into the underlying state space, the robot performs* state estimation*, a technique used, for example, in some forms of control [22]. The advantage of such an approach is that, after estimating its state, the robot can make use of all of the tools and algorithms designed for the perfect information case.* ◇

Example 2.8 *Another possibility is to use probability models to map the history I-state to a posterior distribution over X. This approach has proven very successful in robotics. (See, for example, [49, 79, 99, 107, 136, 152, 153, 160].) Note that such techniques are most interesting when the robot's motions and sensing are subject to stochastic noise, for which the extensions described in Section 3.6.1 are needed.* ◇

Finally, in some situations, certain parts of the available information may be of more interest than others. In such cases, choosing the right I-map allows us to isolate the relevant information.

Example 2.9 *Recall the formulation given in Example 2.3. The completion of some tasks, such as delivery or clean up tasks, can be defined in terms of the world state in $\mathcal{M}$, regardless of the robot's configuration in $\mathcal{C}$. Given a state space $X = \mathcal{C} \times \mathcal{M}$, let $\omega : X \rightarrow \mathcal{M}$ denote a map that "forgets" the robot's configuration, so that (q, m) maps to m. Then set of possible world states is given by an I-map $\kappa_{pw} : \mathcal{I}_{hist} \rightarrow \mathrm{pow}(\mathcal{M})$ under which*

$$\kappa_{pw}(\eta_k) = \{\omega(x) \mid x \in \kappa_{ndet}(\eta_k)\}. \tag{2.13}$$

With this I-map, the robot's configuration is ignored in the derived I-state. ◇

We conclude our presentation of I-maps by identifying and defining a certain class of these maps that is particularly relevant for reasoning about planning problems. For a given I-map $\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}$, suppose there exists an *information transition function* $f_{\mathcal{I}} : \mathcal{I} \times U \times Y \rightarrow \mathcal{I}$, such that for any $\eta_k \in \mathcal{I}_{hist}$, $u_k \in U$, and $y_k \in Y$,

$$f_{\mathcal{I}}(\kappa(\eta_k), u_k, y_k) = \kappa(\eta_k, u_k, y_k). \tag{2.14}$$

If such a function exists, then κ is a *sufficient I-map*. The intuition is that the I-states derived by κ retain enough information about the history to determine future derived I-states. This concept is closely related to the idea of a sufficient statistic [20]. The practical importance is that the robot can then "live in" the derived I-space, discarding the histories.

Example 2.10 *The nondeterministic I-map from Example 2.6 is a sufficient I-map. The information transition function can be defined directly in terms of f and h:*

$$f_{\mathcal{I}}(\eta_k, u_k, y_k) = \bigcup_{x \in \kappa(\eta_k)} \{f(x, u_k)\} \cap \{f(x, u_k) \mid x \in X, y_k = h(x, u_k)\}. \tag{2.15}$$

As a result, the robot is able to maintain the set of states consistent with its sensor and action histories, without recording the histories themselves. ◇

Example 2.11 *For most systems, the "possible world states" I-map κ_{pw} from Example 2.9 is not a sufficient I-map, because changes in the world state cannot generally be predicted without some knowledge of configuration of the robot.* ◇

2.3 Tasks and solutions

The traditional definitions for planning problems, in which a subset of the state space is given as a goal region, are inadequate in this context. Instead, we define a goal region $\mathcal{I}_G \subset \mathcal{I}_{hist}$ in the history I-space. Goal regions of this form can express, for example, problems for which the goal is to reduce uncertainty, without regard for the final state. A goal region fully defines a *task* for the robot.

A solution to such a task is *policy* over a derived I-space:

$$\pi : \mathcal{I} \rightarrow U. \tag{2.16}$$

If, for any starting state, repeated applications of π lead the I-state into $\mathcal{I}_G$, then π is said to be a *solution* to the problem.

Example 2.12 *Suppose the robot's task is to reach a certain region X_G in state space, and to recognize the achievement of this goal. This task can be represented as a goal region in $\mathcal{I}_{hist}$ by choosing*

$$\mathcal{I}_G = \{\eta \in \mathcal{I}_{hist} \mid \kappa_{ndet}(\eta) \subseteq X_G\}. \tag{2.17}$$

See Figure 2.8. ◇

Example 2.13 *As an example of the expressive power of forming goal regions in I-space, suppose a robot must visit a certain, known target region $\mathcal{C}_T$ in its configuration space $\mathcal{C}$. The robot, however, need not recognize that it has achieved this goal until potentially after it has left $\mathcal{C}_T$. The state space is $X = \mathcal{C} \times \mathcal{E}$, with the environment is chosen from some $\mathcal{E}$. No other information about the environment is available at the start. See Figure 2.9. This situation might arise, for example, in some surveillance or delivery tasks. How can this type of goal be expressed as a region in the history I-space?*

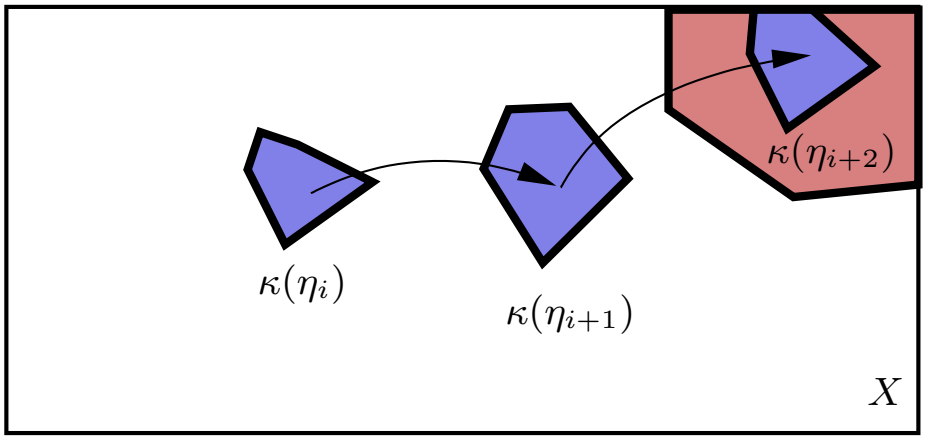


Figure 2.8: A goal region in $\mathcal{I}_{ndet}$, in which the robot must enter X_G in state space and recognize this goal.

Start with a derived I-space $\mathcal{I} = \text{pow}(X - (\mathcal{C}_T \times \mathcal{E}))$. *The intuition is that derived I-state contains the final states of all histories that (1) are consistent with the robot's history I-state and (2) have not yet visited* $\mathcal{C}_T$. *The I-map can be defined recursively:*

$$\kappa(\,) = X - (\mathcal{C}_T \times \mathcal{E}) \tag{2.18}$$

$$\kappa(\eta, u, y) = \{f(x, u) \mid x \in \kappa(\eta), y = h(x, u)\} - (\mathcal{C}_T \times \mathcal{E}) \tag{2.19}$$

At each step, κ *discards any possibilities that have visited* $\mathcal{C}_T$. *Note the similarity to the recursive definition of* κ_{ndet} *(Equation 2.11) but the difference from the (similar but incorrect) definition* $\kappa(\eta) = \kappa_{ndet}(\eta) - (\mathcal{C}_T \times \mathcal{E})$. *Observe that* κ *is a sufficient I-map. The relevance of this I-map is that the goal region in* $\mathcal{I}_{hist}$ *is precisely*

$$\mathcal{I}_G = \{\eta \in \mathcal{I}_{hist} \mid \kappa(\eta) = \emptyset\}\,. \tag{2.20}$$

Therefore, to recognize completion of this sort of visitation problem, it suffices to track the robot's I-state in $\mathcal{I}$ *and report when this set becomes empty.* ◇

Example 2.14 *An important special case occurs when the observation space* Y *is a single-*

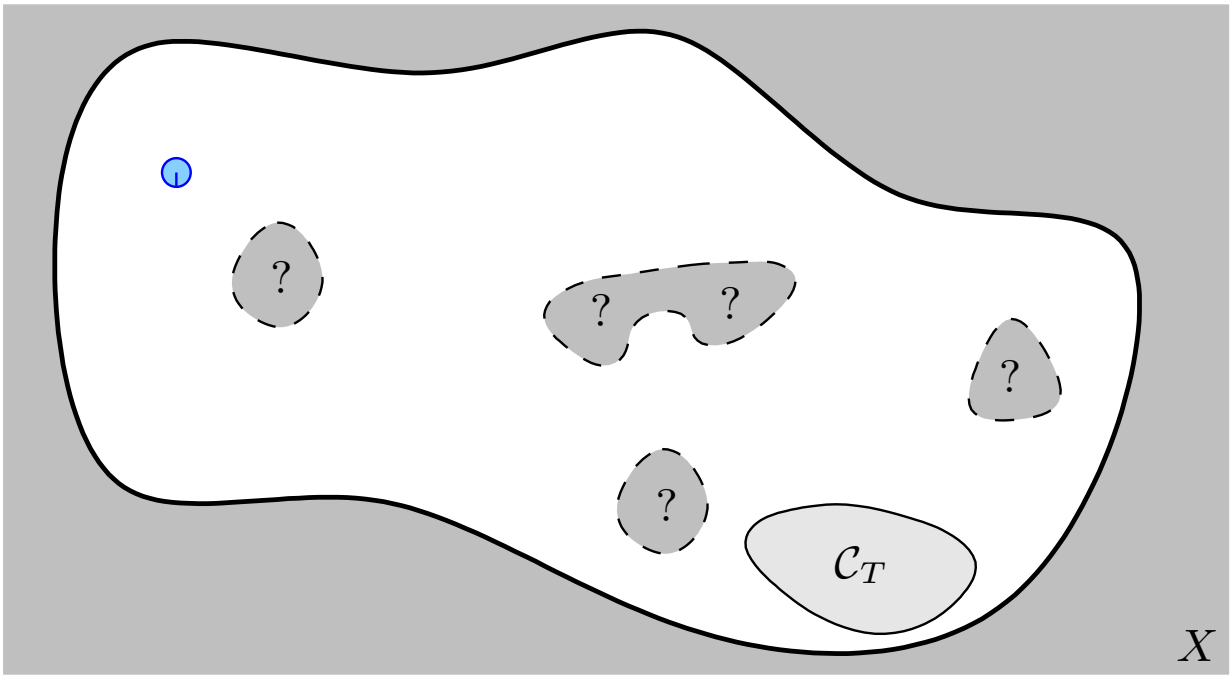


Figure 2.9: A visitation problem. The robot, without knowing its configuration or environment ahead of time, must visit $\mathcal{C}_T$.

ton, perhaps containing only a single observation returned at every stage regardless of the current state or the action chosen. The action sequence alone fully determines the I-states reached. This situation models a sensorless planning task for which it is sufficient to specify a sequence of actions

$$u_1, \ldots, u_K \tag{2.21}$$

rather than a complete policy. ◇

Figure 2.10 summarizes the basic model presented in this chapter.

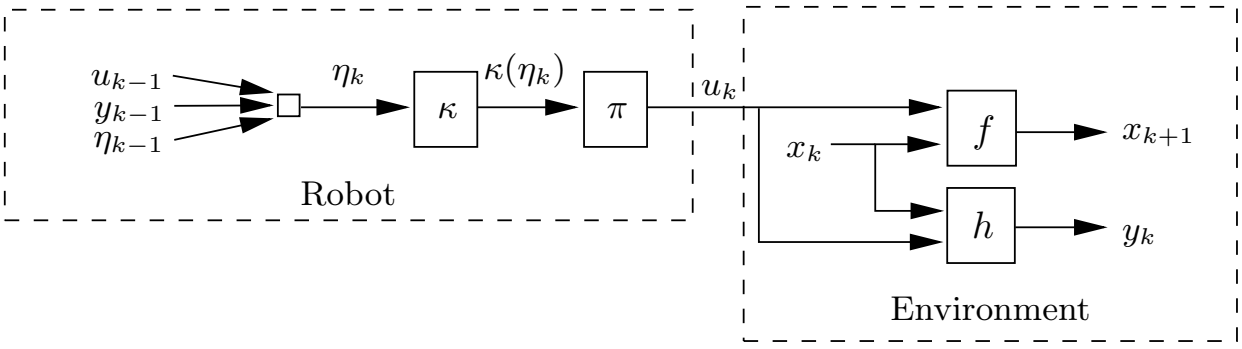


Figure 2.10: A single execution stage for a robot with uncertainty in its state.

Chapter 3

Comparing the power of robots

Suppose we want a robot to complete some task, such as navigating to a goal, manipulating an object, or localizing itself within its environment. Many different combinations of sensing and motion modalities have been used to complete each of these tasks. Indeed, much of the robotics literature is concerned with finding *sufficient conditions* on the sensing and actuation capabilities needed to complete such tasks.

In this chapter we take a complementary approach. For a given task, we are interested in determining the *necessary conditions*: What sensors and actuators are needed? What are the *information requirements* of robotic tasks? The long-term goal of this research is to develop a theory of robots and sensing that helps in answering such questions. Answers to these questions are important because we expect that a deep understanding of the difficulty of tasks in terms of their information requirements will lead to simpler and less expensive robot designs.

This work is inspired in part by the theory of computation, which begins with precisely defined models of abstract machines, such as finite automata, Turing machines, and so on [80, 144]. In this context, a *problem* is usually a language of strings; to solve the problem is to accept strings in this language and reject all others. The theory of computation gives answers to several kinds of basic questions about these machines and problems.

1. *Solvability*: Can a given machine solve a given problem?

2. *Complexity*: If the machine can solve the problem, how efficiently (in terms of time or space, for example) can it do so?

3. *Comparison*: Are some machines strictly more powerful, in terms of the problems they can solve, than others? It is known, for example, that pushdown automata can accept a strictly larger set of languages than can finite automata. Likewise, Turing machines are more powerful than pushdown automata.

4. *Equivalence*: Are there apparently dissimilar machines that can solve the same set of problems? For example, it is a standard result that a Turing machine with multiple tapes is functionally equivalent to an ordinary single-tape Turing machine. Less obviously, Turing machines and recursive functions have been shown to have equivalent computation power.

These ideas are well understood. In the sense that they form the formal foundation of the discipline, they are part of the core of computer science. Current robotic science lacks a comparable foundation; the field needs a unified theory in which meaningful statements can be made about the complexity of robotic tasks and the robot systems we build to complete these tasks.

Can we adapt standard models of computation to the robotics context? Unfortunately, these models are fundamentally ill-suited for studying robotics problems, because they assume that all of the relevant information is supplied ahead of time on the machine's tape. Sensing and uncertainty are central, defining issues in robotics; this structure is destroyed by an *a priori* encoding of the problem on a machine's tape. Traditional models of online computation (see, for example, [32, 90, 145]) are also inadequate, because they assume that some fixed encoding of the problem is revealed incrementally. In contrast, robotics problems are generally interactive, in the sense that the robot's decisions influence the information that becomes available in the future. Others study robotics problems using similar tools [51, 68, 133], but do not explicitly consider the effects of varying sensing and motion capabilities.

The aim of this chapter is to develop a "sensor-centered" theory for analyzing and com-

paring robot systems. The central idea we present is a notion of *dominance* of one robot model over another. In informal terms:

> A robot R_2 *dominates* another robot R_1 if R_2 can "simulate" R_1, collecting at least as much information as R_1.

We make three primary contributions in developing this idea. First, we present the idea of *robotic primitives* for modeling robot systems as collections of independent components. A single robotic primitive represents a self-contained "instruction set" for the robot that may involve sensing, motion, or both. A robot model is defined by a set of primitives that the robot can use to complete its task. By selecting a "catalog" of primitives from which complete robot systems are constructed, we effectively determine a set of robot systems to consider. For clarity, we define these models in an idealized setting in which time is modeled as a series of discrete stages and the robot has perfect knowledge of its environment, perfect control, and perfect sensing. Second, we give a definition for dominance of one robot system over another that formalizes the imprecise definition above. This definition is based on comparing reachability in a derived information space. By mapping sensor-action histories from a variety of robots into the same derived information space, we can compare the abilities of these robots in a concrete, formal way. We prove some basic properties of this dominance relation and give some examples, including a detailed investigation of the global localization problem. Third, we demonstrate the generality of our ideas by showing how to remove several of the simplifying assumptions we make in the initial presentation.

The challenge of robotics lies in the interactions between sensing, actuation, and computation. In this paper, we focus the effects of varying choices for the robot's sensing and actuation capabilities. The robot's computational abilities (as measured, for example, by processing power or memory limitations) are also relevant, but we do not consider them here.

The remainder of this chapter is organized as follows. Section 3.1 reviews related research. Section 3.2 introduces the concept of robotic primitives and defines the set of robots induced

by a catalog of primitives. In Section 3.3, we describe the information preference relation. The definition of dominance and some basic properties thereof appear in Section 3.4. In Section 3.5, we apply the results from Sections 3.2-3.4 to the global localization task. In Section 3.6, we present several generalizations our basic results to account for environment uncertainty, imperfect control and sensing, and continuous time. We explore the relationship between dominance and reachable sets in Section 3.7. Section 3.8 discusses the limitations of this work and describes some open problems.

This work appears in its current form in [130]. Preliminary versions appear in [128] and [131].

3.1 Related work

Our goals are similar to those of Donald [52]. The reductions in that work are similar to our dominance relation; Donald's notion of calibration is related to our idea of initial conditions. The most fundamental difference is that our analysis is rooted in the information space. We claim that for robotic problems in which sensing is a crucial issue, the information space is the space in which the problem can most naturally be posed. The work of Erdmann [63] is grounded in the preimage planning ideas due to Lozano-Perez, Mason, and Taylor [111]. In Erdmann's work, sensors are modeled by giving a partition of state space. The problem of sensor design is to choose a partition so that from each region in the partition, the robot knows what action to select in order to make progress toward its goal. Others in artificial intelligence [34] and control theory [1, 57, 71] have addressed related issues.

Although the examples in this chapter use nondeterministic uncertainty, which is based on set membership, the basic structure of our analysis is compatible with probabilistic uncertainty models like those of [154]. Many probabilistic methods (for example, [14, 106]) can be characterized as operating in an information space whose members are probability distributions over state space. Our methods can be viewed as axiomatic because they can be

applied in any situation that satisfies the definitions. In this sense, the model of uncertainty used is orthogonal to the questions addressed in this work.

3.2 Defining a set of robot systems

In this section we discuss how a set of robots can be defined in terms of a set of independent components.

At the most concrete level, a robot is a collection of motors and sensors connected to some sort of computer. Between these components there may be interactions via open- or closed-loop controls. We abstract this complexity by defining the notion of a *robotic primitive*. Each robotic primitive defines a "mode of operation" for the robot. When primitives are implemented, they may draw on one or more of the robot's physical sensors or actuators. Every kind of motion or sensing available to the robot must be modeled as a robotic primitive. Robotic primitives correspond roughly to the *oracles* that appear in the theory of computation [144, 146], in the sense that they provide the ability to make certain transitions and collect certain observations, without specifying how these abilities are implemented.

Formally, we define robotic primitives in terms of the action and observation abilities they provide.

Definition 3.1 *A* robotic primitive *(or simply a* primitive*) P_i is a 4-tuple*

$$P_i = (U_i, Y_i, f_i, h_i) \tag{3.1}$$

giving an action set U_i, an observation set Y_i, a state transition function $f_i : X \times U_i \to X$, and an observation function $h_i : X \times U_i \to Y_i$.

Let $\mathcal{RP} = \{P_1, \ldots, P_N\}$ denote a catalog of primitives. We may form a robot model by selecting nonempty subset of $\mathcal{RP}$. A robot defined by the primitive set $R = \{P_{i_1}, \ldots, P_{i_m}\} \subseteq \mathcal{RP}$ has action set $U_R = U_{i_1} \sqcup \cdots \sqcup U_{i_m}$ and observation set $Y_R = Y_{i_1} \sqcup \cdots \sqcup Y_{i_m}$. The $\sqcup$

notation indicates a disjoint union operation, under which identical elements from different source sets remain distinct. The state transition function $f_R : X \times U_R \to X$, and observation function $h_R : X \times U_R \to Y_R$, are formed by unioning the f and h maps from the relevant primitives. When it can be done without ambiguity, we use the phrase *robot model* to refer directly to the set of primitives, rather than to the 5-tuple (X, U, Y, f, h) formed by these primitives. With this usage, it is meaningful to apply set operations such as union or intersection directly to robots.

Note that, given a catalog of primitives $\mathcal{RP}$, we can form a "master" robot model $\widehat{R}$ that includes every primitive in $\mathcal{RP}$. Then the history I-space of $\widehat{R}$ contains as a subset the history I-space of every other robot model that can be formed from $\mathcal{RP}$. As a result, any I-map for $\widehat{R}$ can also be used as an I-map for any robot model formed from $\mathcal{RP}$.

We now give several examples to illustrate the intuition of Definition 3.1. Examples 3.2-3.6 apply to a point robot with orientation in a bounded planar environment E, so $X = E \times S^1$. Illustrations of these primitives appear in Figures 3.1-3.3. We revisit these examples in Sections 3.4 and 3.5.

Example 3.1 *Let $P_A = (S^1, \{0\}, f_A, h_A)$. Let f_A compute relative rotations, so that from a state $x = (x_1, x_2, \theta)$, we have $f_A(x, u) = (x_1, x_2, \theta + u)$. Since $Y_A = \{0\}$ contains only a dummy element, h_A is a trivial function always returning 0. This primitive can be implemented with an angular odometer on a mobile robot capable of rotating in place.* ◇

Example 3.2 *Let $P_C = (S^1 \sqcup \{0\}, S^1, f_C, h_C)$. Define $f_C(x, u)$ to set the rotation coordinate of x to equal u if $u \in S^1$ or to leave x unchanged if $u \in \{0\}$. The observation function h_C returns the robot's final orientation. This primitive amounts to allowing the robot to orient itself with respect to a global reference frame, or to sense its current orientation without rotating. One might implement this primitive using a compass on a robot that can rotate in place.* ◇

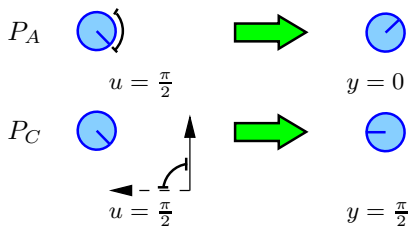


Figure 3.1: Sample executions of the primitives of Examples 3.1 and 3.2. [top] P_A allows the robot rotate relative to its current orientation. [bottom] P_C allows the robot to rotate relative to a globally defined "north" direction.

Example 3.3 *Let $P_T = (\{0\}, \{0\}, f_T, h_T)$. Define f_T to compute a forward translation to the obstacle boundary. This primitive can be implemented with a contact sensor on a mobile robot that can reliably move forward.* ◇

Example 3.4 *Let $P_L = ([0, \infty), [0, \infty), f_L, h_L)$. For $x \in X$ and $u \in U$, define $f_L(x, u)$ to compute a forward translation of distance at most u, stopping short only if the robot reaches an obstacle first. The observation $h_L(x, u)$ is the actual distance traveled. This primitive can be implemented with a linear odometer on a robot that can move forward reliably. Depending on implementation issues, a contact sensor may also be needed.* ◇

Example 3.5 *Let $P_R = (\{0\}, [0, \infty), f_R, h_R)$. For all $x \in X$, $f_R(x, 0) = x$, so that this primitive never changes the robot's state. The observation $h_R(x, u)$ is the distance to the nearest obstacle directly in front of the robot. This primitive models the capabilities of a forward-facing unidirectional range sensor.* ◇

Example 3.6 *Let $P_G = (\{0\}, \mathbb{R}^2, f_G, h_G)$. Again, $f_G(x, u) = x$ for all x and u. For a state $x = (x_1, x_2, \theta)$, let $h_G(x, 0) = (x_1, x_2)$. This primitive roughly corresponds to a GPS device that the robot can periodically poll to determine its location in the plane.* ◇

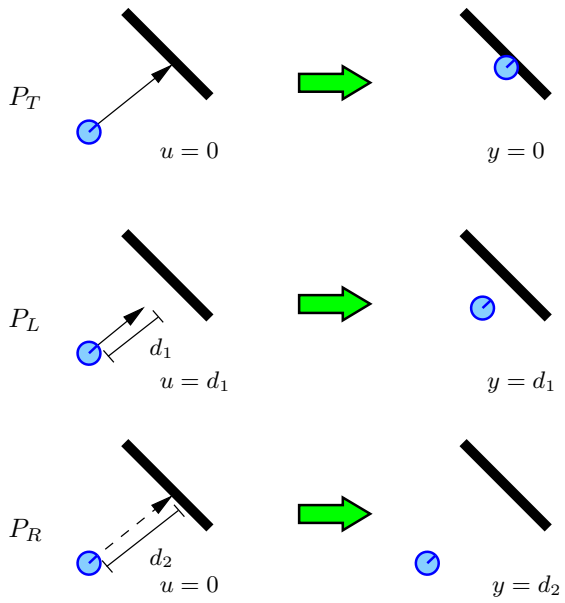


Figure 3.2: Sample executions of the primitives of Examples 3.3-3.5. [top] P_T allows the robot to translate forward until it reaches an obstacle. [middle] P_L allows a robot to specify a distance to translate. [bottom] P_R allows the robot to measure the distance forward to the nearest obstacle, but does not change the robot's state.

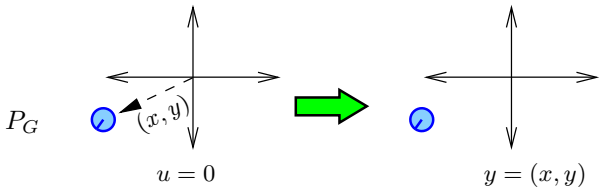


Figure 3.3: A sample execution of the primitive of Example 3.6. The robot senses its position, but its state does not change.

Other possibilities for primitives include landmark detectors, wall followers, visibility sensors, and so on. A more complete listing of sensors suitable for adaptation into robotic primitives appears in Section 11.5.1 of [102].

There are several benefits to modeling robot systems as collections of primitives. First, we claim that robotic primitives represent the right level of abstraction at which *planning* problems are interesting but manageable. If we consider sensors at too fine a level of detail, the problem takes on the character of a closed-loop control system. If the primitives are too

sophisticated, we risk trivializing the planning problem while creating an unbearable modeling burden. Second, by dividing time into discrete stages, we avoid the technical difficulties of describing the robot's progression through $\mathcal{I}$ in continuous time. This consideration is increasingly important if we allow noise to affect state transitions or observations. We address issues related to the modeling of time more completely in Section 3.6.2.

3.3 The information preference relation

Our goal is a dominance relation under which we can declare one robot "better than" another. To do so, we need a formal notion of one I-state being superior, in the sense of encoding better information, than another. To that end, choose a derived I-space $\mathcal{I}$ and an I-map κ into $\mathcal{I}$. Equip $\mathcal{I}$ with a partial order, which we call an *information preference relation*. Write $\kappa(\eta_1) \preceq \kappa(\eta_2)$ to indicate that $\kappa(\eta_2)$ is preferred to $\kappa(\eta_1)$. We require that for any $\eta_1, \eta_2 \in \mathcal{I}_{hist}$, and for any $u \in U$ and $y \in Y$,

$$\kappa(\eta_1) \preceq \kappa(\eta_2) \implies \kappa(\eta_1, u, y) \preceq \kappa(\eta_2, u, y). \tag{3.2}$$

This is a consistency property requiring preference for one I-state over another to be preserved across transitions in I-space.

Example 3.7 *Regardless of $\mathcal{I}$ or κ, it is well-defined (but perhaps unhelpful) to use a trivial relation under which $\kappa(\eta_1) \preceq \kappa(\eta_2)$ if and only if $\kappa(\eta_1) = \kappa(\eta_2)$.* $\diamond$

Example 3.8 *Under nondeterministic uncertainty, we can define $\kappa_{ndet}(\eta_1) \preceq \kappa_{ndet}(\eta_2)$ if and only if $\kappa_{ndet}(\eta_2) \subseteq \kappa_{ndet}(\eta_1)$. To show that (3.2) is satisfied, suppose $\kappa_{ndet}(\eta_1) \preceq \kappa_{ndet}(\eta_2)$. Let $x \in \kappa_{ndet}(\eta_2, u, y)$. The definition of κ_{ndet} ensures that there exists some*

$x' \in \kappa_{ndet}(\eta_2)$ *such that* $f(x', u) = x$ *and* $h(x', u) = y$. *However, because* $\kappa_{ndet}(\eta_2) \subseteq \kappa_{ndet}(\eta_1)$, *we have* $x' \in \kappa_{ndet}(\eta_1)$. *It follows that* $x \in \kappa_{ndet}(\eta_1, u, y)$. ◇

The information preference relation we choose affects the goal regions that are sensible to consider. We should select a region in which, for every I-state in the region, we also include any I-states preferable to it. This formalizes the intuition that a robot in the goal region should not prefer to be outside the goal. Definition 3.2 codifies this idea of a sensible goal region.

Definition 3.2 *Consider a set* $I \subset \mathcal{I}$ *of derived I-states. If, for any* $\kappa(\eta_1) \in I$ *and* $\kappa(\eta_2) \in \mathcal{I}$ *with* $\kappa(\eta_1) \preceq \kappa(\eta_2)$, *we have* $\kappa(\eta_2) \in I$, *then* I *is* preference closed.

Alternatively, one can view preference closure as a constraint on $\preceq$. Fixing a space $\mathcal{G}$ of potential goal regions, we admit a partial order $\preceq$ only if every region in $\mathcal{G}$ is preference closed under $\preceq$. Note that the trivial definition of $\preceq$ in Example 3.7 always passes this test, regardless of $\mathcal{G}$.

3.4 A dominance relation over robot systems

Now we turn our attention to a definition of dominance of one robot system over another. This dominance relation induces a partial order over robot systems, according to their sensing and actuation abilities. The intuition is that dominance is based on one robot's ability to "simulate" another.

Definition 3.3 (Robot dominance) *Consider two robots*

$$R_1 = (X^{(1)}, U^{(1)}, Y^{(1)}, f^{(1)}, h^{(1)}), \text{ and} \tag{3.3}$$

$$R_2 = (X^{(2)}, U^{(2)}, Y^{(2)}, f^{(2)}, h^{(2)}). \tag{3.4}$$

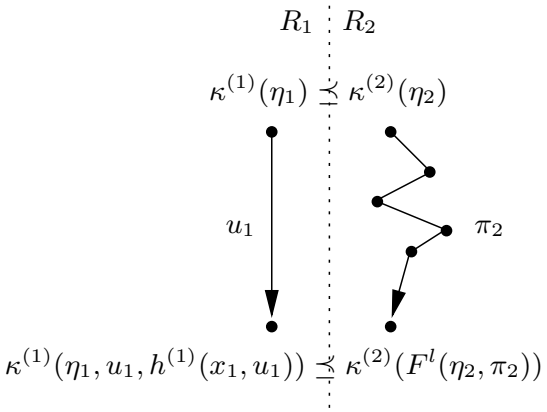


Figure 3.4: An illustration of Definition 3.3. If R_2 can always reach an I-state better than the one reached by R_1, then $R_1 \trianglelefteq R_2$.

Choose a derived I-space $\mathcal{I}$ *and I-maps* $\kappa^{(1)} : \mathcal{I}^{(1)}_{hist} \to \mathcal{I}$ *and* $\kappa^{(2)} : \mathcal{I}^{(2)}_{hist} \to \mathcal{I}$. *If, for all*

- $\eta_1 \in \mathcal{I}^{(1)}_{hist}$,
- $\eta_2 \in \mathcal{I}^{(2)}_{hist}$ *for which* $\kappa^{(1)}(\eta_1) \preceq \kappa^{(2)}(\eta_2)$, *and all*
- $u_1 \in U^{(1)}$,

there exists a policy $\pi_2 : \mathcal{I}^{(2)}_{hist} \to U^{(2)}$ *such that for all* $x_1 \in X^{(1)}$ *consistent with* η_1 *and all* $x_2 \in X^{(2)}$ *consistent with* η_2, *there exists a positive integer* l *such that*

$$\kappa^{(1)}(\eta_1, u_1, h^{(1)}(x_1, u_1)) \preceq \kappa^{(2)}(F^l(\eta_2, \pi_2, x_2)), \tag{3.5}$$

then R_2 dominates R_1 *under* $\kappa^{(1)}$ *and* $\kappa^{(2)}$, *denoted* $R_1 \trianglelefteq R_2$. *If* $R_1 \trianglelefteq R_2$ *and* $R_2 \trianglelefteq R_1$, *then* R_1 *and* R_2 *are* equivalent, *denoted* $R_1 \equiv R_2$. *If* $R_1 \not\trianglelefteq R_2$ *and* $R_2 \not\trianglelefteq R_1$ *then* R_1 *and* R_2 *are* incomparable, *denoted* $R_1 \boxtimes R_2$.

Informally, Definition 3.3 means that, for any transition made by R_1, there exists some strategy for R_2 to reach an information state at least as good, in the sense of information preference, as that reached by R_1. This is what we mean when we describe the statement $R_1 \trianglelefteq R_2$ as meaning that R_2 can simulate R_1. See Figure 3.4.

3.4.1 Dominance examples

Several examples will clarify the definition.

Example 3.9 *Let $R_1 = \{P_R\}$ and $R_2 = \{P_A, P_L\}$. Recall the definitions of these primitives from Examples 3.2, 3.4, and 3.5. We argue under nondeterministic uncertainty that $R_1 \trianglelefteq R_2$ by showing that R_2 can simulate R_1 in the precise sense of Definition 3.3. Let $\eta_1 \in \mathcal{I}^{(1)}_{hist}$ and $\eta_2 \in \mathcal{I}^{(2)}_{hist}$ with $\kappa_{ndet}(\eta_1) \preceq \kappa_{ndet}(\eta_2)$. Since $U^{(1)} = \{0\}$, there is only one choice for u_1. Let $l = 4$ and define π_2 so that R_2, starting from η_2, executes these actions in succession:*

(1) Use P_L with a very large input to move forward to the nearest obstacle. Let $d = h(x, u)$ denote the distance moved.

(2) Use P_A with $u = 180^\circ$ to perform a half turn.

(3) Use P_L with $u = d$ to return the robot to its initial position.

(4) Use P_A with $u = 180^\circ$ to perform a half turn, returning the robot to its original orientation.

This policy is illustrated in Figure 3.5. It is easy to verify that from any $x \in X$, we have

$$\kappa_{ndet}(\eta_1, u_1, h(x, u_1)) \preceq \kappa_{ndet}(F^4(\eta_2, \pi_2, x)),$$

and therefore $R_1 \trianglelefteq R_2$. Since R_1, which is completely immobile, cannot simulate the translations or rotations of R_2, we have $R_2 \not\trianglelefteq R_1$.

Note that these relationships are based on the robots' ability to move through $\mathcal{I}_{ndet}$, and do not consider any notion of the cost of motion or sensing. The introduction of such a cost function would likely lead to Pareto optima that express tradeoffs between the complexity of sensing built into the robot and the execution costs of particular plans executed by the robot. We do not consider such tradeoffs here. ◇

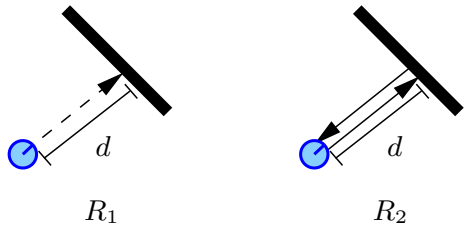


Figure 3.5: An illustration of Example 3.9. The robot $R_2 = \{P_A, P_L\}$ dominates the robot $R_1 = \{P_R\}$ because the former can simulate the latter. [left] A distance measurement made directly by R_1. [right] Distance is measured indirectly by R_2 using its linear odometer.

Example 3.10 *Let $R_1 = \{P_T\}$ and $R_2 = \{P_L\}$. We show under nondeterministic uncertainty that $R_1 \trianglelefteq R_2$. Let $\eta_1 \in \mathcal{I}^{(1)}_{hist}$ and $\eta_2 \in \mathcal{I}^{(2)}_{hist}$ with $\kappa(\eta_1) \preceq \kappa(\eta_2)$. There is only one choice for u_1. Choose $l = 1$ and define π_2 to choose an input for P_L larger than the diameter of the environment. This causes the motions of R_1 and R_2 to be identical. The resulting derived I-states $\kappa(\eta_1')$ and $\kappa(\eta_2')$ for R_1 and R_2 are the same, except that R_2 receives a meaningful sensor reading that may reduce the resulting nondeterministic I-state. This sensor information only makes $\kappa(\eta_2')$ smaller, so the preference $\kappa(\eta_1') \preceq \kappa(\eta_2')$ is maintained. Conclude that $R_1 \trianglelefteq R_2$.* ◇

It bears emphasis that the relation induced by Definition 3.3 depends on the I-maps used. The next three examples illustrate this.

Example 3.11 *Let $R_1 = \{P_A\}$ and $R_2 = \{P_C\}$. We argue that $R_1 \trianglelefteq R_2$ under the usual nondeterministic I-map with the initial condition of total uncertainty. Let $\eta_1 \in \mathcal{I}^{(1)}_{hist}$ and $\eta_2 \in \mathcal{I}^{(2)}_{hist}$ with $\kappa_{ndet}(\eta_1) \preceq \kappa_{ndet}(\eta_2)$. Let $u_1 \in U_1 = S^1$. Choose $l = 2$ and define π_2 to select the following two actions:*

(1) Use P_C with $u = 0$ to sense the robot's orientation without changing the state. Let θ denote this orientation.

(2) Use P_C to rotate the robot to orientation $\theta + u$ in the global frame.

As in Example 3.10, the resulting states for R_1 and R_2 are identical but, since R_2 knows its orientation, it may be able to eliminate some candidate states that R_1 cannot. This establishes that $R_1 \trianglelefteq R_2$. Are R_1 and R_2 equivalent under this I-map? No, because R_2 can, with a single action, sense its orientation, but this information can never be gathered by R_1. Therefore $R_2 \not\trianglelefteq R_1$ and $R_1 \not\equiv R_2$. ◇

Example 3.12 *Consider a situation identical to that of Example 3.11, but modify κ_{ndet} for a different initial condition $\kappa_{ndet}(\,) = \mathbb{R}^2 \times \{\pi/2\}$. That is, the robot begins its execution knowing its initial orientation. At every step, R_1 knows its orientation in the global frame, and can simulate R_2 using angle addition. Therefore we have $R_2 \trianglelefteq R_1$. But using the same reasoning as in Example 3.11, we know $R_1 \trianglelefteq R_2$. Therefore, for this I-map, we have $R_1 \equiv R_2$.* ◇

Example 3.13 *Recall the state space formulation from Example 2.3, which expresses the system state as a combination of the robot's configuration in $\mathcal{C}$ along with the relevant external state of the world in $\mathcal{M}$. Recall also the "possible worlds" I-map $\kappa_{pw} : \mathcal{I}_{hist} \to pow(\mathcal{M})$ from Example 2.9. In this context, the statement that $R_1 \trianglelefteq R_2$ under κ_{pw} implies that any change in the world state made by R_1 can be mimicked by R_2. This formulation is interesting because, under κ_{pw}, dominance is related only to results in the external world, without considering the intermediate or final configurations of the robots themselves.* ◇

3.4.2 Properties of the dominance relation

We conclude this section with some basic properties that follow from Definition 3.3.

Lemma 3.1 *The dominance relation $\trianglelefteq$ is a partial order. Likewise $\equiv$ is indeed an equivalence relation.*

Lemma 3.2 *Consider three robots R_1, R_2, and R_3 formed from primitives in $\mathcal{RP}$ and an I-map κ for the master robot model $\widehat{R}$ of $\mathcal{RP}$. If $R_1 \trianglelefteq R_2$ under κ, we have:*

(a) $R_1 \trianglelefteq R_1 \cup R_3$ *(Adding primitives never hurts)*

(b) $R_2 \equiv R_2 \cup R_1$ *(Redundancy doesn't help)*

(c) $R_1 \cup R_3 \trianglelefteq R_2 \cup R_3$ *(No unexpected interactions)*

Proof: **(a)** Let $\eta_1 \in \mathcal{I}^{(1)}_{hist}$, $\eta_{13} \in \mathcal{I}^{(13)}_{hist}$, and $u_1 \in U_1$. Assume $\kappa(\eta_1) \preceq \kappa(\eta_{13})$. Choose $l = 1$ and $\pi_{13}(\eta) = u_1$ for all η. For all x, we have $\kappa(\eta_1, u_1, h(x, u_1)) \preceq \kappa(\eta_{13}, u_1, h(x, u_1)) = \kappa(F^l(\eta_{13}, \pi_{13}, x))$, completing the proof.

(b) It follows from part (a) that $R_2 \trianglelefteq R_1 \cup R_2$. It remains to show that $R_1 \cup R_2 \trianglelefteq R_2$. Let $\eta_{12} \in \mathcal{I}^{(12)}_{hist}$, $\eta_2 \in \mathcal{I}^{(2)}_{hist}$, and $u_{12} \in U_2 \cup U_1$. Assume $\kappa(\eta_{12}) \preceq \kappa(\eta_2)$. Either $u_{12} \in U_1$ or $u_{12} \in U_2$. If $u_{12} \in U_1$, then because $R_1 \trianglelefteq R_2$ there exist π_2 and l satisfying the definition for $R_1 \cup R_2 \trianglelefteq R_2$. If $u_{12} \in U_2$, choose $l = 1$ and $\pi_2(\eta) = u_{12}$ for all η. For all x, we have $\kappa(\eta_{12}, u_{12}, h(x, u_{12})) \preceq \kappa(\eta_2, u_{12}, h(x, u_{12})) = \kappa(F^l(\eta_2, \pi_2, x))$, completing the proof.

(c) Let $\eta_{13} \in \mathcal{I}^{(13)}_{hist}$, $\eta_{23} \in \mathcal{I}^{(23)}_{hist}$, and $u_{13} \in U_1 \sqcup U_3$. Assume $\kappa(\eta_{13}) \preceq \kappa(\eta_{23})$. Either $u_{13} \in U_1$ or $u_{13} \in U_3$. If $u_{13} \in U_1$, then because $R_1 \trianglelefteq R_2$ there exist π_{23} and l satisfying the definition for $R_1 \cup R_3 \trianglelefteq R_2 \cup R_3$. If $u_{13} \in U_3$, then choose $l = 1$ and $\pi_{23}(\eta) = u_{13}$ for all η. For all x, we have $\kappa(\eta_{13}, u_{13}, h(x, u_{13})) \preceq \kappa(\eta_{23}, u_{13}, h(x, u_{13})) = \kappa(F^l(\eta_{23}, \pi_{23}, x))$, completing the proof. □

Corollary 3.3 *If $R_1 \equiv R_2$, then $R_1 \cup R_3 \equiv R_2 \cup R_3$.*

Proof: Apply Lemma 3.2c twice. □

Lemma 3.2c might be misleading. Certainly, hardware components can be made to interact in interesting ways. For example, a control system might combine information from linear and angular odometers to execute circular arc motions. This apparent contradiction results from the definition of robotic primitives, which execute *serially*, rather than in parallel. In

this sense, robotic primitives model sensing and actuation strategies as complete "packages," rather than the individual sensors or motors themselves.

Lastly, we connect the idea of dominance to the ability of robots to complete tasks.

Lemma 3.4 (Solution by imitation) *Consider two robots R_1 and R_2 with $R_1 \trianglelefteq R_2$ and a preference-closed goal region $\mathcal{I}_G$. If R_1 can reach $\mathcal{I}_G$, then R_2 can reach $\mathcal{I}_G$.*

Proof: Use the policy π_2 implied by Definition 3.3 to complete the task with R_2. □

This tight connection between dominance and task-completing ability provides some motivation for the form of dominance we propose.

3.5 Extended example: Global localization

In this section we present a detailed example using the definitions of Sections 3.3 and 3.4. We preview the *global localization* task of Chapter 4, in which the robot has an accurate map of its environment but has no knowledge of its position within that environment. The purpose of this example is to show how the results of Section 3.4 can be used to discover the information requirements of this particular problem in robotics. An analogy can be made to the classification of languages in the theory of computation. It is known, for example, that to accept the language of palindromes requires a machine with computation abilities at least as powerful as a pushdown automaton. In this section, we derive similar results regarding the sensing and motion abilities needed to complete the active global localization task.

3.5.1 Task definition

Let $E \subseteq \mathbb{R}^2$ denote a planar environment in which a point robot moves. Assume that E is polygonal, bounded, closed, and simply-connected and that the rotational symmetry group of E is trivial. As in previous examples, the robot's state space is $X = E \times S^1$. We consider a catalog $\mathcal{RP} = \{P_A, P_C, P_T, P_L\}$ of four primitives from Examples 3.1-3.3. From these

primitives we can form 15 distinct robots. For brevity, we use concatenation to indicate the primitives with which a robot is equipped, so that CT refers to a robot with primitive set $\{P_C, P_T\}$. Similar names apply to the other 14 robot models.

Use nondeterministic uncertainty, so that $\mathcal{I} = \mathcal{I}_{ndet}$. The initial condition is total uncertainty, so $\kappa(\ ) = X$. For the information preference relation, use the definition from Example 3.8, in which information preference is defined by subset containment. The goal region for the localization task is

$$\mathcal{I}_G = \{\eta \in \mathcal{I}_{hist} \mid |\kappa(\eta)| = 1\}. \tag{3.6}$$

That is, we want to command the robot so that only a single final state is consistent with its history I-state. If the robot can complete the task for any E consistent with the assumptions above, we say that the robot can localize itself.

3.5.2 Equivalences and dominances

Although $\mathcal{RP}$ generates 15 robot models, we can use the results of Section 3.4 to group them into equivalence classes.

Lemma 3.5 *The following equivalences hold:*

(a) $CA \equiv C$

(b) $CTA \equiv CT$

(c) $TL \equiv L$

(d) $TAL \equiv AL$

(e) $CAL \equiv CTL \equiv CTAL \equiv CL$

The three remaining robot models, A, T, and AT, are in singleton equivalence classes.

Proof: (a) Combine Example 3.11 and Lemma 3.2b. (b) Combine Example 3.11, Lemma 3.2b, and Corollary 3.3. (c) Combine Example 3.10 and Lemma 3.2b. (d) Combine Example 3.10,

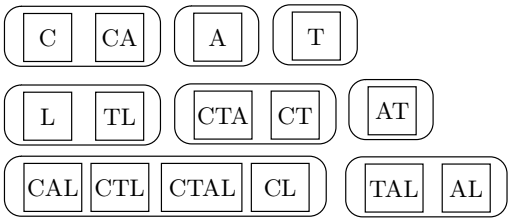


Figure 3.6: Fifteen robot models grouped into their eight equivalence classes.

Lemma 3.2b, and Corollary 3.3. (e) Combine Examples 3.10 and 3.11, Lemma 3.2b, and Corollary 3.3. □

These equivalences are illustrated in Figure 3.6. From each, select the unique robot with the fewest primitives and discard the remaining 7 robots. We can state several dominances between these classes.

Lemma 3.6 *Between representatives of the equivalence classes from Lemma 3.5, the following dominances hold:*

(a) $C \trianglelefteq CT \trianglelefteq CL$

(b) $A \trianglelefteq AT \trianglelefteq AL \trianglelefteq CL$

(c) $L \trianglelefteq AL \trianglelefteq CL$

(d) $T \trianglelefteq AT \trianglelefteq CT \trianglelefteq CL$

Proof: Combine Examples 3.10 and 3.11 with Lemma 3.2a. □

3.5.3 Completing the localization task

Which equivalence classes contain robots that can complete the localization task? First, notice that some robot models are so simple that we can rule them out immediately.

Lemma 3.7 *None of C, A, L, and T can localize themselves.*

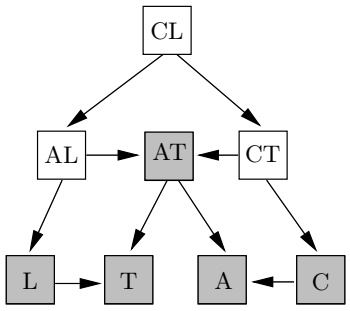


Figure 3.7: Classification of robot models under which the localization task can be completed. Shaded models do not admit a solution. Arrows indicate dominances.

Proof: For C and A, notice that no action changes the robot's position and no observation is influenced by position. Therefore neither robot can ever gather information about its position. For L and T, notice that the robot can never change its orientation. Information available to the robot is limited to the ray extending from its initial state to the nearest obstacle forward. Since E may contain continua of starting states consistent with this information, neither robot can localize itself. □

The next chapter, which considers this localization problem in greater detail for AL, CT, and AT, provides the following result.

Lemma 3.8 (Chapter 4) *AL and CT can localize themselves but AT cannot.*

Finally, we can finish the classification. The results of Lemmas 3.7-3.9 are summarized in Figure 3.7.

Lemma 3.9 *CL can localize itself.*

Proof: Combine Lemma 3.4 with Lemma 3.8. □

The result is a complete classification of the solvability of the localization problem over this hierarchy.

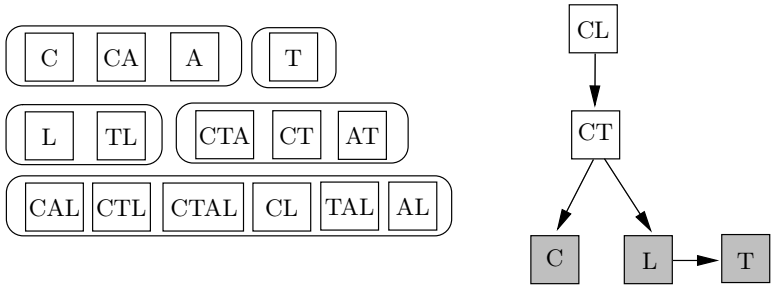


Figure 3.8: Equivalence classes, dominance, and ability to solve the localization problem when the robot's initial orientation is given. Compare to Figures 3.6 and 3.7.

3.5.4 The value of initial information

These results depend heavily on the assumption that the robot starts with no information about its initial state. What happens if this initial condition is improved? Examples 3.11 and 3.12 showed that knowledge of the initial orientation makes the difference between $A \trianglelefteq C$ and $A \equiv C$. Figure 3.8 shows how this initial condition affects the equivalence and dominance for the 15 robot models we generated from $\mathcal{RP} = \{P_C, P_A, P_L, P_T\}$. The eight equivalence classes are collapsed into only five. Note especially that AT can localize itself with knowledge of its initial orientation, but not without it.

This inquiry is in the same spirit as existing work that considers the competitive ratio [116, 145] of online algorithms with respect some optimality measure. There are at least two important differences. First, existing work makes only a binary distinction in the initial information: Either the robot has an accurate map of its environment, or it starts with no information. We are are able to consider a broader range of initial conditions, including those that are unrelated to the environment. Second, rather than studying how the *performance* of a single robot changes by the introduction of new information, we can examine how the *requirements* on the robot itself change. Does a simpler robot suffice if we provide additional initial information?

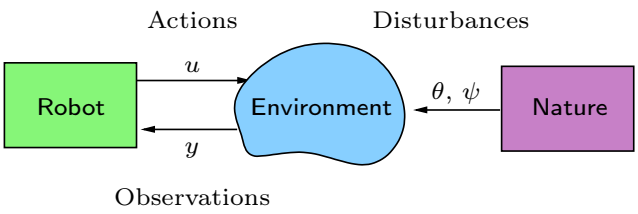


Figure 3.9: As the robot interacts with its environment, an artificial decision maker nature generates disturbances.

3.6 Extensions and generalizations

This section contains a series of extensions and generalizations to the techniques presented in Section 3.4. The intention is to illustrate that, although the preceding results are for a certain class of highly idealized systems, the general structure of our analysis is useful for a wider variety of problems with greater degrees of realism and generality. We propose methods for dealing with sensing and control uncertainty (Section 3.6.1), and with continuous time (Section 3.6.2). Although we present each method separately, the extensions are orthogonal in the sense that it is straightforward to apply both of them at once.

3.6.1 Imperfect sensing and control

We have assumed so far that the robot can execute all of its actions with perfect precision and complete reliability. The motions of real robots are imprecise and unpredictable. Moreover, although we have accounted for the importance of sensing by assuming that the robot is uncertain of its current state and must rely on sensing, we have assumed that sensor readings are uncorrupted by noise. A more realistic sensor model would allow information from sensors to be subject to error.

We propose to follow the approach used in game theory [26, 132] and some robotics research [150] by envisioning an abstract external decision maker called "nature." The current state, the action chosen by the robot, and the choices made by nature combine

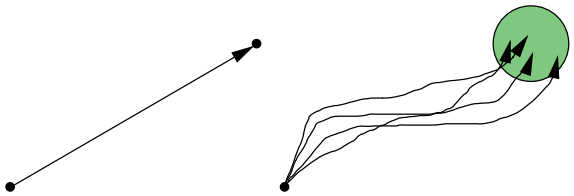


Figure 3.10: [left] The robot in Example 3.14 gives displacement inputs that determine a nominal trajectory. [right] Nature interferes with this motion, but error bounds ensure that the final state is contained in a circle of radius $k\theta_{max}$.

to determine how the state changes; given this information, the state trajectory is fully determined. See Figure 3.9. Formally, define a *nature action space* Θ and augment the state transition function f to depend on nature's choice of $\theta \in \Theta$ at each stage, so that $f : X \times U \times \Theta \to X$. Nature affects the robot's observations in a similar way. Define a *nature observation action space* Ψ and redefine the observation function $h : X \times U \times \Psi \to Y$. The policy application function F must be generalized to account for nature actions, so that

$$\eta_{m+k} = F^m(\eta_k, \pi, x_k, \theta_k, \ldots, \theta_{k+m}, \psi_k, \ldots, \psi_{k+m}). \tag{3.7}$$

Note that, in contrast to the simpler formulation of Equation 2.9, the robot's current state, history I-state, and policy are no longer sufficient to predict future history I-states.

The next examples illustrate how nature might interfere.

Example 3.14 *Consider a point robot that can move freely in the plane by issuing displacement commands, but whose motion is subject to noise. Let u_{max} denote a bound on the magnitude of the displacement in each stage, and let θ_{max} denote a bound on magnitude of the error in this displacement. Let $X = \mathbb{R}^2$, $U = \{u \in \mathbb{R}^2 \mid ||u|| \leq u_{max}\}$, $\Theta = \{\theta \in \mathbb{R}^2 \mid ||\theta|| \leq \theta_{max}\}$, and $f(x, u, \theta) = x + u + \theta$. At stage k, the robot can be certain that its state lies within a closed disk of radius $k\theta_{max}$, centered at the nominal (error free) final point. See Figure 3.10.* ◇

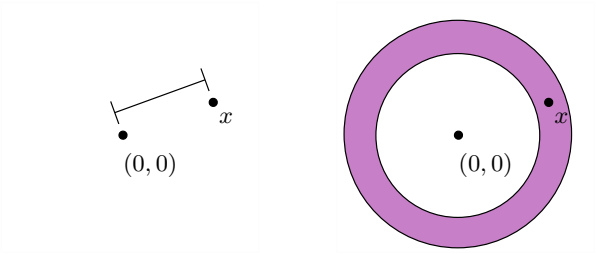


Figure 3.11: [left] The robot in Example 3.15 has a sensor that reports a noisy estimate of the distance to the origin. [right] Accounting for noise bounded by ψ_{max}, the observation confines the robot's state to an annulus of width $2\psi_{max}$.

Example 3.15 *Suppose a mobile robot has a sensor that reports the distance to some landmark. Let $X = \mathbb{R}^2$ and $Y = [0, \infty)$. Without loss of generality, position the landmark at the origin. Assume that the sensor has bounded additive error, so that $\Psi = [-\psi_{max}, \psi_{max}]$ and $h(x, \psi) = ||x|| + \psi$. See Figure 3.11. At each stage, the robot knows that its state is within an annulus of width $2\psi_{max}$, centered at the origin.* ◇

In the presence of interference from nature, there are at least two relevant solution concepts.

1. A strategy $\pi : \mathcal{I}_{hist} \rightarrow U$ is a *possible solution* if there exists some stage k and choices of $\theta_1, \ldots, \theta_k$ and $\psi_1, \ldots, \psi_k$ for which the robot reaches a derived I-state $\eta_k \in \mathcal{I}_G$. The robot may reach $\mathcal{I}_G$, but it is also possible that control or sensing errors will prevent it from achieving this goal.

2. A strategy $\pi : \mathcal{I}_{hist} \rightarrow U$ is a *guaranteed solution* if there exists some stage k such that for all choices of $\theta_1, \ldots, \theta_k$ and $\psi_1, \ldots, \psi_k$, the robot reaches a derived I-state $\eta_k \in \mathcal{I}_G$. The robot can always reach its goal, regardless of any interference by nature.

Other solution concepts, such as those based on performance bounds or on probabilistic guarantees of reaching the goal, are possible but we do not consider them here. In this context, Definition 3.3 must be generalized to include universal quantifiers over nature's actions.

Definition 3.4 (Robot dominance with sensing and control error) *Consider two robot systems*

$$R_1 = (X^{(1)}, U^{(1)}, Y^{(1)}, \Theta^{(1)}, \Psi^{(1)}, f^{(1)}, h^{(1)}), \text{ and} \tag{3.8}$$

$$R_2 = (X^{(2)}, U^{(2)}, Y^{(2)}, \Theta^{(2)}, \Psi^{(2)}, f^{(2)}, h^{(2)}). \tag{3.9}$$

Choose a derived I-space $\mathcal{I}$ and I-maps $\kappa^{(1)} : X^{(1)} \to \mathcal{I}$ and $\kappa^{(2)} : X^{(2)} \to \mathcal{I}$. If, for all

- $\eta_1 \in \mathcal{I}^{(1)}_{hist}$,
- $\eta_2 \in \mathcal{I}^{(2)}_{hist}$ *for which* $\kappa^{(1)}(\eta_1) \preceq \kappa^{(2)}(\eta_2)$, *and all*
- $u_1 \in U^{(1)}$,

there exists a policy $\pi_2 : \mathcal{I}^{(2)}_{hist} \to U^{(2)}$ such that for all $x_1 \in X^{(1)}$ consistent with η_1 and all $x_2 \in X^{(2)}$ consistent with η_2, there exists a positive integer l such that for all

- $\theta_1 \in \Theta^{(1)}$,
- $\psi_1 \in \Psi^{(1)}$,
- $\theta_{2,1}, \ldots, \theta_{2,l} \in \Theta^{(2)}$,
- $\psi_{2,1}, \ldots, \psi_{2,l} \in \psi^{(2)}$,

we have

$$\kappa^{(1)}(\eta_1, u_1, h^{(1)}(x_1, u_1, \psi_1)) \preceq \kappa(F^l(\eta_2, \pi_2, x_2, \theta_{2,1}, \ldots, \theta_{2,l}, \psi_{2,1}, \ldots, \psi_{2,l})) \tag{3.10}$$

then R_2 dominates *R_1 under $\kappa^{(1)}$ and $\kappa^{(2)}$, denoted $R_1 \trianglelefteq R_2$.*

The next example demonstrates that Definition 3.4 behaves reasonably.

Example 3.16 (Varying error bounds) *Recall the incompletely specified models in Examples 3.14 and 3.15. Consider two robot systems R_1 and R_2 with state transitions as in Example 3.14 and observations as in Example 3.15; R_1 and R_2 differ only in their error*

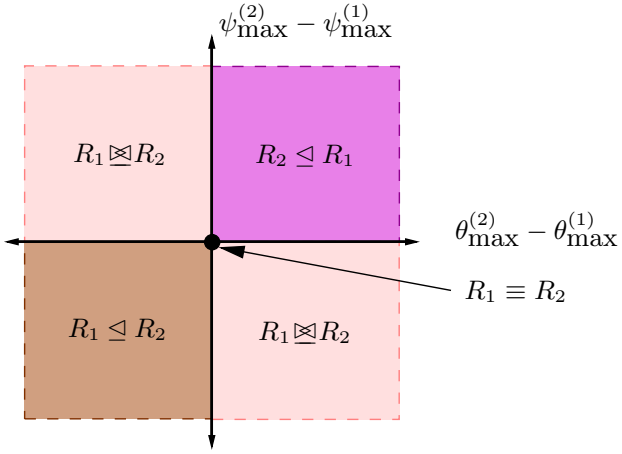


Figure 3.12: Effects of varying error bounds on dominance between two otherwise identical robots. The horizontal axis shows the difference in actuation error bounds. The vertical axis shows the difference in sensing error bounds.

bounds $\theta^{(1)}_{max}$, $\psi^{(1)}_{max}$, $\theta^{(2)}_{max}$, *and* $\psi^{(2)}_{max}$. *We compare these robots under* κ_{ndet}. *Comparing* $\theta^{(1)}_{max}$ *to* $\theta^{(2)}_{max}$, *and* $\psi^{(1)}_{max}$ *to* $\psi^{(2)}_{max}$, *there are four cases:*

1. *If* $\theta^{(1)}_{max} = \theta^{(2)}_{max}$ *and* $\psi^{(1)}_{max} = \psi^{(2)}_{max}$, *then* $R_1 \equiv R_2$.

2. *If* $\theta^{(1)}_{max} \leq \theta^{(2)}_{max}$ *and* $\psi^{(1)}_{max} \leq \psi^{(2)}_{max}$, *then* $R_2 \trianglelefteq R_1$.

3. *If* $\theta^{(2)}_{max} \leq \theta^{(1)}_{max}$ *and* $\psi^{(2)}_{max} \leq \psi^{(1)}_{max}$, *then* $R_1 \trianglelefteq R_2$.

4. *Otherwise,* $R_2 \bowtie\!\!\!\!\!\!= R_1$.

See Figure 3.12. These results follow in a straightforward manner from Definition 3.4. The intuition is that one robot system dominates the other if and only if its error bounds are not larger. ◇

3.6.2 Continuous time

The models presented to this point manage time in discrete stages, in which the robot makes a single decision at each stage. This discretization of time may be unsatisfactory for many

kinds of systems, especially those that require complicated control strategies. Continuous-time models have a more direct correspondence with reality. To make the appropriate generalizations, we replace the discrete sequences of states, actions, and observations with functions of a continuous time parameter t.

The state space X, action space U, and observation space Y remain unchanged from the discrete stage formulation. At each instant t, the robot chooses some $u(t) \in U$. Let $\widetilde{U}_t$ denote the space of all functions from $[0, t)$ into U, and let $\widetilde{U} = \bigcup_{t\in[0,\infty)} \widetilde{U}_t$. For simplicity of notation, adopt the convention that $[0, 0) = \emptyset$. Define $\widetilde{u} : [0, \infty) \to U$ as the robot's complete action history, and let $\widetilde{u}_t \in \widetilde{U}$ denote the robot's action history up to (but exclusive of) time t. We include a special *termination action* $u_T \in U$. The robot selects u_T to indicate that it has finished its task and intends to terminate execution. We require that if $u(t) = u_T$, then $u(t') = u_T$ for all $t' > t$. We describe changes in the state with a *state transition function*

$$\Phi : X \times \bigcup_{t\in[0,\infty)} \widetilde{U}_t \to X. \tag{3.11}$$

The intuition is that, given a starting state $x(0)$, and an action history $\widetilde{u}_t$, the state transition function computes the resulting state

$$x(t) = \Phi(x(0), \widetilde{u}_t). \tag{3.12}$$

This notation of a "black box" state transition function follows notation employed in control theory, for example by Chen [40].

Example 3.17 *A familiar special case of (3.12) occurs if $\widetilde{u}$ is a smooth function and there exists a function f such that*

$$\Phi(x(0), \widetilde{u}_t) = x(0) + \int_0^t f(x(s), u(s))\mathrm{d}s. \tag{3.13}$$

In this case, the system dynamics can be described by the differential equation $\dot{x} = f(x, u)$. $\diamond$

As time passes, the robot's sensors provide feedback in the form of observations drawn from an observation space Y. Let $\widetilde{Y}_t$ denote the space of functions mapping $[0, t]$ into Y and let $\widetilde{Y} = \bigcup_{t \in [0,\infty)} \widetilde{Y}_t$. The robot's complete observation history is $\widetilde{y} : [0, \infty) \rightarrow Y$. The observation history up to t (inclusive) is $\widetilde{y}_t \in \widetilde{Y}_t$. The observations received by the robot are governed by the observation function[1] $h : X \rightarrow Y$. The history I-state becomes

$$\mathcal{I}_{hist} = \bigcup_{t \in [0,\infty)} \widetilde{U}_t \times \widetilde{Y}_t, \tag{3.14}$$

and the history I-state at time t is $\eta(t) = (\widetilde{u}_t, \widetilde{y}_t) \in \mathcal{I}_{hist}$. A state x is consistent with an I-state $\eta(t) = (\widetilde{u}_t, \widetilde{y}_t)$ if and only if there exists some starting state $x(0)$ such that $\Phi(x(0), \widetilde{u}_t) = x$ and $h(x(t')) = y(t')$ for $t' < t$.

We describe the robot's strategy as a feedback policy $\pi : \mathcal{I}_{hist} \rightarrow U$ that specifies an action for each history I-state. We assume that a given strategy is executed until it selects u_T. The time when this occurs, the resulting final state, and the observations received along the way are all affected by the strategy π itself and the starting state $x(0)$. Assuming that the robot executes π, the termination time is

$$T(\pi, x(0)) = \inf\{t \in [0, \infty) \mid \pi(\eta(t)) = u_T\}. \tag{3.15}$$

The final state, denoted $F(\pi, x(0))$, is

$$F(\pi, x(0)) = \Phi(x(0), \widetilde{u}_{T(\pi, x(0))}). \tag{3.16}$$

[1] In our discrete-stage formulation, we used a slightly different observation model, in which $h : X \times U \rightarrow Y$. In a continuous-time adaptation, the time period over which observations are available is the half-open interval $[0, t)$; $\widetilde{y}_t$ would be undefined at t itself. As a result, the closest we could come to a memoryless strategy is to use the left-hand limit of $\widetilde{y}_t$ at t, $\kappa_{obs}(\eta(t)) = \lim_{t' \rightarrow t^-} y(t')$, provided the limit exists. (Compare to Example 3.19.) This technicality is part of the motivation for preventing y from depending directly on u, as we have done in this section. A more complete treatment of these kinds of sensor models appears in Section 11.1.1 of [102].

The next three examples illustrate that feedback over a derived I-space can be a natural way to express familiar kinds of strategies.

Example 3.18 (Open loop strategy) *Let $\mathcal{I}_{time} = [0, \infty)$ and consider the I-map $\kappa_{time}(\eta(t)) = t$. In this case, the derived I-state is simply the time elapsed. If the robot has an intended open loop action trajectory $\omega : [0, t_f) \rightarrow U$, a strategy to execute γ is $\pi(\eta(t)) = \omega(\kappa_{time}(\eta(t))$ if $t < t_f$ and $\pi(\eta(t)) = u_T$ otherwise.* ◇

Example 3.19 (Memoryless strategy) *Another possibility is that it is enough to know the "most recent" observation, so $\mathcal{I}_{obs} = Y$ and $\kappa_{obs}(\eta(t)) = y(t)$. Given a memoryless plan $\gamma : Y \rightarrow U$, the composed function $\kappa_{obs} \circ \gamma : \mathcal{I}_{hist} \rightarrow U$ is a memoryless information feedback strategy.* ◇

Example 3.20 (Concatenating strategies) *Given two strategies π_1 and π_2, a new strategy that concatenates them (that is, executes them in sequence) is expressed by $\pi(\eta(t)) = \pi_1(\eta(t))$ if $\pi_1(\eta(t)) \neq u_T$ and $\pi(\eta(t)) = \pi_2(\eta(t))$ otherwise. By nesting this construction, arbitrarily many strategies can be chained together.* ◇

Definition 3.3 generalizes in a natural way.

Definition 3.5 (Robot dominance in continuous time) *Consider two continuous-time robot systems*

$$R_1 = (X^{(1)}, U^{(1)}, Y^{(1)}, \Phi^{(1)}, h^{(1)}), \text{ and} \tag{3.17}$$

$$R_2 = (X^{(2)}, U^{(2)}, Y^{(2)}, \Phi^{(2)}, h^{(2)}). \tag{3.18}$$

If, for all

- $\eta^{(1)}(t_1) \in \mathcal{I}^{(1)}_{hist}$,

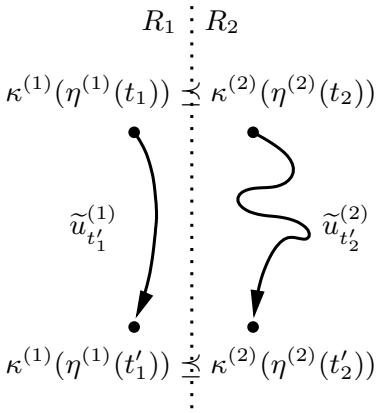


Figure 3.13: An illustration of Definition 3.5. Compare to Figure 3.4.

- $\eta^{(2)}(t_2) \in \mathcal{I}^{(2)}_{hist}$ *for which* $\kappa^{(1)}(\eta^{(1)}(t_1)) \preceq \kappa^{(2)}(\eta^{(2)}(t_2))$,
- $t'_1 \in [0, \infty)$, *and all*
- $\widetilde{u}^{(1)}_{t'_1} \in \widetilde{U}^{(1)}_{t'_1}$,

there exists an information feedback strategy $\pi_2 : \mathcal{I}^{(2)}_{hist} \to U^{(2)}$, *such that for all* $x^{(1)} \in X^{(1)}$ *consistent with* $\eta^{(1)}(t_1)$ *and* $x^{(2)} \in X^{(2)}$ *consistent with* $\eta^{(2)}(t_2)$, *there exists* $t'_2 \in [0, \infty)$ *such that if* R_1 *executes* $\widetilde{u}^{(1)}_{t'_1}$ *from time* t_1 *to* t'_1 *and* R_2 *executes* $\pi^{(2)}$ *from time* t_2 *to* t'_2, *we have*

$$\kappa^{(1)}(\eta^{(1)}(t'_1)) \preceq \kappa^{(2)}(\eta^{(2)}(t'_2)) \tag{3.19}$$

then R_2 dominates R_1 *under* $\kappa^{(1)}$ *and* $\kappa^{(2)}$, *denoted* $R_1 \trianglelefteq R_2$.

See Figure 3.13. The next two examples illustrate some implications of Definition 3.5.

Example 3.21 (Omniscient sensing and perfect control) *Consider a degenerate case with* $Y = X$, *and* $h(x) = x$. *This situation gives the robot complete information about its state. Choose* $\mathcal{I} = X$ *and* $\kappa(\eta(t)) = y(t) = x(t)$. *Let* $\kappa(\eta_1) \preceq \kappa(\eta_2)$ *if and only if* $\kappa(\eta_1) = \kappa(\eta_2)$, *as in Example 3.7. In this context, Definition 3.3 becomes a statement about the regions of state space reachable by different control systems.*

Suppose three such systems R_1, R_2, and R_3 differ only in their action spaces $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$. Let $Z(A)$ denote the subset of state space reachable by a robot with action space A, starting from some initial state $x(0)$. Suppose $R_1 \trianglelefteq R_2$. R_3 need not be comparable to either R_1 or R_2. Note that additional robot models can be constructed from unions of $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$. We have the following results:

$$Z(U^{(1)}) \subseteq Z(U^{(2)} \cup U^{(3)}) \tag{3.20}$$

$$Z(U^{(1)}) = Z(U^{(1)} \cup U^{(2)}) \tag{3.21}$$

$$Z(U^{(1)} \cup U^{(3)}) \subseteq Z(U^{(2)} \cup U^{(3)}) \tag{3.22}$$

These results are analogous to Lemma 3.2. Note that in combining action spaces in this way, we allow the robot to choose sequentially *the action set from which to choose its action. The results fail if the robot is somehow allowed to choose actions from each constituent set in parallel. We discuss reachable sets in greater detail in Section 3.7.* ◇

Example 3.22 (A Lost Cow) *A well-known problem in online algorithms is the* lost cow problem *[17, 89] in which a near-sighted cow moves along a fence searching for a gate, as illustrated in Figure 3.14. The difficulty under the standard sensing model is that the cow must systematically search in both directions from its initial position without any information about the distance or direction to the gate. The interest in this problem derives from potential applications in (or at least the potential for better understanding of) exploration in unbounded environments.*

We formulate the lost cow problem and consider how the sensing model affects the cow's searching ability. Let $X = \mathbb{R}$, in which $x(t)$ is the position of the gate relative to the cow at time t. Let the action space be $U = [-1, 1]$ with $\Phi(x(0), \widetilde{u}_t) = x(0) + \int_0^t u(s)\mathrm{d}s$. We compare three distinct models C_1, C_2, and C_3 under κ_{ndet}.

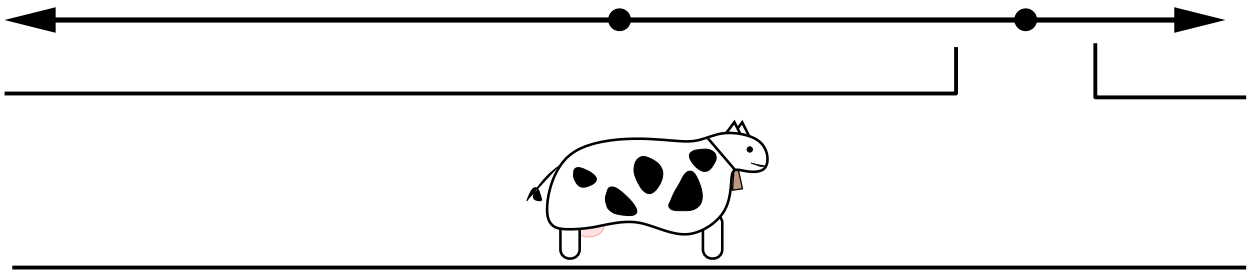


Figure 3.14: The lost cow of Example 3.22 searching for a gate.

1. *C_1: Let $Y^{(1)} = \mathbb{R}$ and $h^{(1)}(x) = x$. Here the cow can determine both the direction and distance to the gate.*

2. *C_2: Let $Y^{(2)} = \{-1, 0, 1\}$ and $h(x) = \text{sign}(x)$. This allows the cow to determine the direction it must move to reach the gate, but not the distance.*

3. *C_3: Let $Y^{(3)} = \{0, 1\}$ and $h^{(2)}(x) = 1$ if $x = 0$ and $h^{(2)}(x) = 1$ otherwise. This is the standard lost cow sensing model, in which the cow cannot see the gate from a distance, but can detect the gate when it arrives.*

Perhaps surprisingly, these three models are equivalent in the sense of Definition 3.5. This is because each can eventually determine its state (by finding the gate) and after the state is known, state uncertainty cannot recur. To simulate C_1 with C_3, first execute the algorithm of [17], then move to the state occupied by C_1.

Note, however, that this result fails if the cow's motion is subject to control error, as described in Section 3.6.1. In this case, for example, C_1 can reach states arbitrarily far from the gate without state uncertainty, whereas the uncertainty for C_3 increases as it moves farther from the gate, its only reference point. Observe also that allowing the cow to give a precisely-executed velocity input is essentially equivalent to giving the cow a precise linear odometer. LaValle and Egerstedt address this issue in [103]. ◇

We conclude our discussion of continuous-time models by showing how a discrete stage model in the form of Chapter 2 can be constructed from a continuous-time model in the

form presented above. Consider a division of time into variable length stages, in which, in each stage, the robot executes a single information feedback strategy to completion. We require of each of these strategies the following special property:

Definition 3.6 (History invariance) *If, for all $\eta(t) \in \mathcal{I}_{hist}$, all $x \in X$ consistent with $\eta(t)$, and all $y(0) \in Y$, we have $F(\pi, x, \eta(t)) = F(\pi, x, \eta(0))$, then π is a* history-invariant *strategy.*

The intuition of the definition is that the robot executing π is free to use the observation and action history generated during its own execution, but it cannot peer into the past before its execution began in order to make decisions. Given a continuous-time robot system $R = (X, U, Y, \Phi, h)$ (as defined in this section) and a set Π of history-invariant information feedback strategies, construct a discrete-stage system (as in Chapter 2) $\overline{R} = (X, \overline{U}, \overline{Y}, \overline{f}, \overline{h})$ as follows:

1. The state space X is the same.
2. The action space is $\overline{U} = \Pi$.
3. The observation space is $\overline{Y} = \widetilde{Y}$.
4. The state transition function is $f : X \times \overline{U} \to X$, with $f(x, \pi) = F(\pi, x, \eta(0))$.
5. The observation function is $h : X \times \overline{U} \to \overline{Y}$.

The system starts at some (unknown) initial state $x_1 \in X$. Let $x_k \in X$, $u_k \in \overline{U}$, and $y_k \in \overline{Y}$, denote the appropriate values at stage k. These sequences are related to each other by $x_{k+1} = f(x_k, u_k)$ and $y_k = h(x_k, u_k)$. The history I-state consists of the action and observation histories: $\eta_k = (u_1, y_1, \ldots, u_{K-1}, y_{K-1})$. This construction gives a discrete-stage system faithful to the dynamics in both state space and I-space of the underlying continuous time system.

Lemma 3.10 *Any action sequence $u_1, \ldots, u_K$ executed by $\overline{R}$ reaches the same final state x and the analogous final history I-state as does R.*

Note, however, that in making this transformation, we must choose a set Π of strategies and may therefore restrict the space of plans that the robot can execute. If Π does not contain a sufficiently rich selection of information feedback strategies, there may be regions of I-space that are no longer reachable under the discretized model. In this way, Π is analogous to the catalog of robotic primitives $\mathcal{RP}$ introduced in Section 3.2.

3.7 Dominance and reachable sets

Definition 3.3 is local in an important sense. Comparisons are made based on a robot's ability to simulate another robot's trajectory in $\mathcal{I}$, step by step. This section explores a more global view, defined in terms of reachable sets and preference closure. We begin with a few definitions.

Definition 3.7 *The* reachable set *for a robot R starting at history I-state η, denoted $Z(R, \eta)$, is the set of all history I-states that (1) are consistent with at least one state and (2) can be formed by appending actions and observations to η.*

Definition 3.8 *The* reverse preference closure *$Pc^-(N)$ of a set $N \subseteq \mathcal{I}_{hist}$ is the set of all I-states $\eta_1 \in \mathcal{I}_{hist}$ for which there exists an $\eta_2 \in N$ with $\kappa(\eta_1) \preceq \kappa(\eta_2)$. Equivalently, we can define*

$$Pc^-(N) = \bigcup_{\eta_2 \in N} \{\eta_1 \in \mathcal{I}_{hist} \mid \kappa(\eta_1) \preceq \kappa(\eta_2)\}. \tag{3.23}$$

Now we can establish the relationship between dominance and reachable sets. See Figure 3.15.

Lemma 3.11 *$R_1 \trianglelefteq R_2$ if and only if, for all $\eta_1 \in \mathcal{I}^{(1)}_{hist}$ and $\eta_2 \in \mathcal{I}^{(2)}_{hist}$ such that $\kappa(\eta_1) \trianglelefteq \kappa(\eta_2)$, we have $Z(R_1, \eta_1) \subseteq Pc^-(Z(R_2, \eta_2))$.*

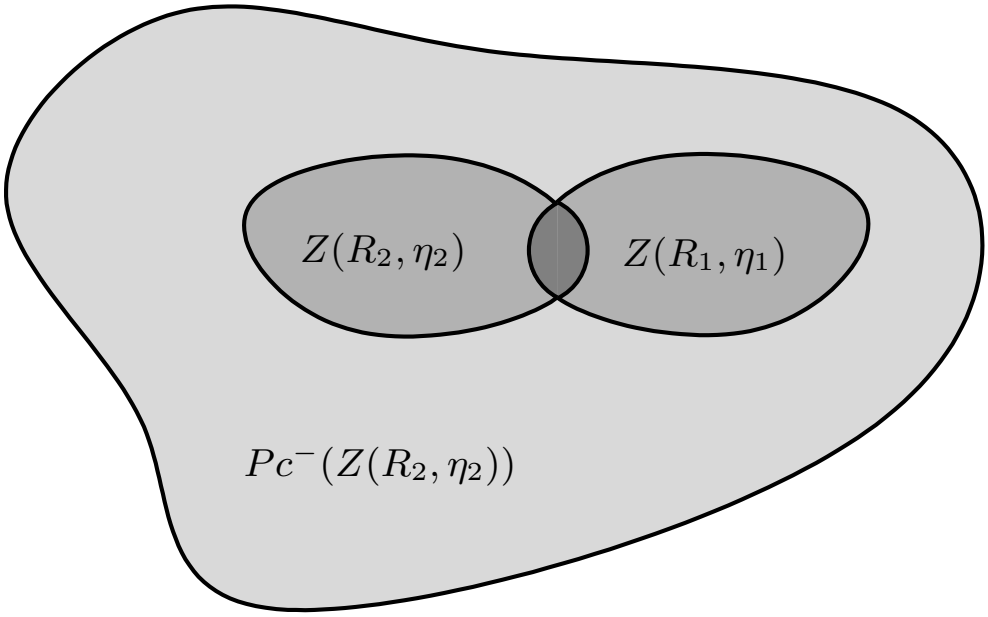


Figure 3.15: The relationship between dominance, reachable sets, and preference closure.

Proof: Let $\eta_1 \in \mathcal{I}^{(1)}_{hist}$ and $\eta_2 \in \mathcal{I}^{(2)}_{hist}$ with $\kappa(\eta_1) \preceq \kappa(\eta_2)$.

Forward direction: Suppose $R_1 \trianglelefteq R_2$ and let $\eta_1' \in Z(R_1, \eta_1)$ to show that $\eta_1' \in Pc^-(Z(R_2, \eta_2))$. Since η_1' is reachable by R_1 from η_1, there exists an action-observation sequence $u_k, y_k, \ldots, u_{k+m}, y_{k+m}$ such that $\eta_1' = (\eta_1, u_k, y_k, \ldots, u_{k+m}, y_{k+m})$. We argue that there exists $\eta_2' \in Z(R_2, \eta_2)$ such that $\kappa(\eta_1') \preceq \kappa(\eta_2')$. See Figure 3.16. Use induction on m:

- When $m = 0$, choose $\eta_2' = \eta_2$. Then we have $\kappa(\eta_1) \preceq \kappa(\eta_2)$ and trivially $\eta_2' \in Z(R_2, \eta_2)$.
- Assume the statement holds for $m - 1$ to show for m. By this inductive hypothesis, there exists $\eta_2'' \in \mathcal{I}^{(2)}_{hist}$ such that

$$\kappa(\eta_1, u_k, y_k, \ldots, u_{k+m-1}, y_{k+m-1}) \preceq \kappa(\eta_2''). \tag{3.24}$$

Equation 3.24, combined with $R_1 \trianglelefteq R_2$, implies that for all x, there exist π_2 and l such that

$$\kappa(\eta_1, u_k, y_k, \ldots, u_{k+m}, y_{k+m}) \preceq \kappa(F^l(\eta_2'', \pi_2, x)). \tag{3.25}$$

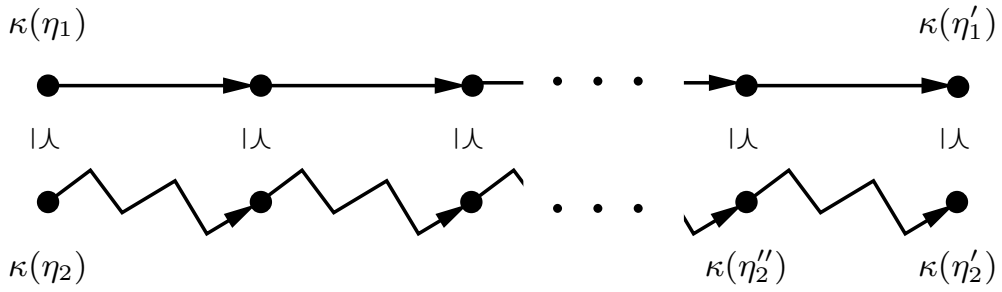


Figure 3.16: An illustration of the proof of the forward part of Lemma 3.11.

Choose $\eta_2' = F^l(\eta_2'', \pi_2, x)$, so that $\kappa(\eta_1') \preceq \kappa(\eta_2')$ and $\eta_2' \in Z(R_2, \eta_2)$.

Given such an η_2', it follows immediately that $\eta_1 \in Pc^-(Z(R_2, \eta_2))$.

Backward direction: Suppose that $\eta_1' \in Pc^-(Z(R_2, \eta_2))$ to show that $R_1 \trianglelefteq R_2$. Let $\eta_1 \in \mathcal{I}^{(1)}_{hist}$, $\eta_2 \in \mathcal{I}^{(2)}_{hist}$, with $\kappa(\eta_1) \preceq \kappa(\eta_2)$, $u_1 \in U^{(1)}$, $x \in X$, and $\eta_1' = (\eta_1, u_1, h(x, u_1))$. By the definition of Z, we have $\eta_1' \in Z(R_1, \eta_1)$. This implies $\eta_1' \in Pc^-(Z(R_2, \eta_2))$, which in turn, implies that there exists $\eta_2' \in Z(R_2, \eta_2)$ such that $\kappa(\eta_1') \preceq \kappa(\eta_2')$. Since η_2' is reachable by R_2 from η_2, there exists an action-observation sequence $u_k, y_k, \ldots, u_{k+m}, y_{k+m}$ such that

$$\eta_2' = (\eta_2, u_k, y_k, \ldots, u_{k+m}, y_{k+m}). \tag{3.26}$$

Let $l = m$ and construct an policy π_2 that executes the actions $u_k, \ldots, u_{k+m}$ in sequence. Conclude that $R_1 \trianglelefteq R_2$. □

It may be tempting to use a simpler definition that omits the universal quantifiers over η_1 and η_2, and instead considers reachability only from the initial condition. That is, one might consider situations in which

$$Z(R_1, (\,)) \subseteq Pc^-(Z(R_2, (\,))). \tag{3.27}$$

Note, however, that this condition is strictly weaker than the dominance relation of Definition 3.3. See Example 3.23.

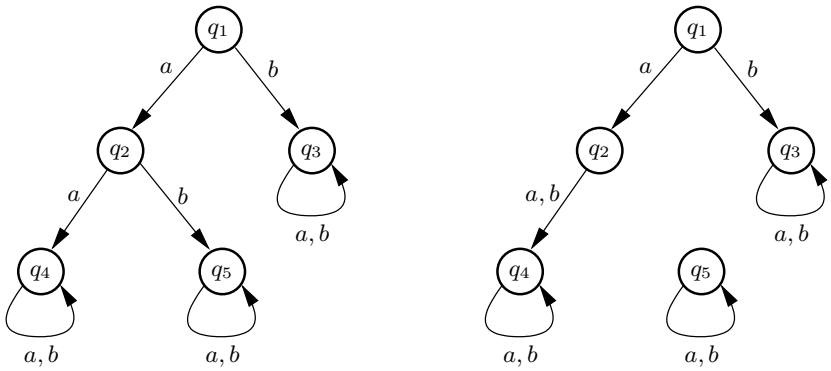


Figure 3.17: Two transition systems from Example 3.23 demonstrating the importance of the directedness transitions in $\mathcal{I}$.

Example 3.23 *Consider a five-state system with* $X = \{q_1, q_2, q_3, q_4, q_5\}$ *and complete sensing, that is,* $h(x, u) = x$. *Let* $U = \{a, b\}$ *and define two robot systems that differ only in their state transition functions:*

f_1	a	b
q_1	q_2	q_3
q_2	q_4	q_5
q_3	q_3	q_3
q_4	q_4	q_4
q_5	q_5	q_5

f_2	a	b
q_1	q_2	q_3
q_2	q_4	q_4
q_3	q_3	q_3
q_4	q_4	q_4
q_5	q_5	q_5

See Figure 3.17. Let $\mathcal{I} = X$ *with the identity function for* κ. *Choose*

$$\preceq \;=\; \{(q_3, q_5), (q_1, q_1), (q_2, q_2), (q_3, q_3), (q_4, q_4), (q_5, q_5)\}, \tag{3.28}$$

so that $q_3 \preceq q_5$ *is the only nontrivial preference. It is easy to verify directly that Equation 3.2*

is satisfied. Observe that

$$Z(R_1, (\,)) = \{q_1, q_2, q_3, q_4, q_5\} \tag{3.29}$$

$$Z(R_2, (\,)) = \{q_1, q_2, q_3, q_4\}, \text{ and} \tag{3.30}$$

$$Pc^-(Z(R_2, (\,))) = \{q_1, q_2, q_3, q_4, q_5\}, \tag{3.31}$$

so $Z(R_1, (\,)) = Pc^-(Z(R_2, (\,)))$. *However, choosing* $\eta_1 = \eta_2 = q_2$, *and* $u_1 = b$, R_1 *reaches* q_5 *and* R_2 *(under all policies) reaches* q_4. *Since* $q_5 \not\preceq q_4$, *conclude that* $R_1 \not\preccurlyeq R_2$. *The difference results from the directedness of transitions in* $\mathcal{I}$. *Although* R_2 *can reach* q_5, *this option is no longer available when the robot is at* q_2. ◇

3.8 Discussion

The results of this thesis are intended to lay a foundation for a sensor-centered theory for comparing robotic problems and systems. Great potential exists to build on this foundation, particularly by developing the analogy to the theory of computation even further.

The most obvious avenue for future work is to study a broader collection of problems. Although this thesis considers an active global localization problem in detail, other fundamental robotics problems warrant similar analysis of their information requirements. For example, results exist for limited-sensing versions of navigation [88, 88, 114, 115, 133], exploration [2, 42, 125, 156], and manipulation [4, 5, 6, 59, 72] tasks. Using the techniques we have presented, it should be possible to unify and extend these results to develop a more complete understanding of the sensing and motion abilities needed to solve these problems. Other problems and more complex sensing systems could also be investigated.

One of the most powerful ideas in the theory of computation that we have not explored here is the idea of *reductions*, which hold promise for comparing robotic problems themselves. The resulting statements would have the form "Problem A is at least as hard as Problem

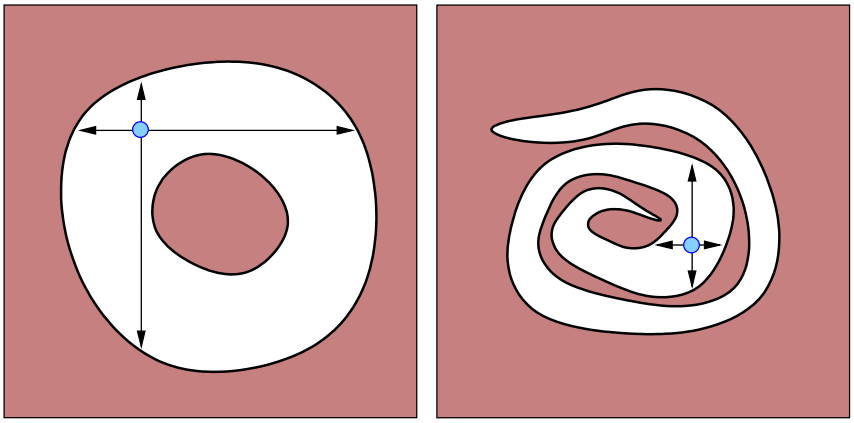


Figure 3.18: A sample decision problem. What sensing is required to decide if a planar environment is simply connected? What robots can distinguish the annulus environment on the left from the helix on the right?

B." To make things more concrete, we might consider *decision problems*, in which the robot with a state space defined as in Example 2.2 must determine if its environment $E \in \mathcal{E}$ has a certain property. Such problems can be expressed naturally as planning problems in I-space. To decide if E has a property $\Xi : \mathcal{E} \to \{0, 1\}$, the robot must reach the goal region

$$\mathcal{I}_{G,\Xi} = \{\eta \in \mathcal{I}_{hist} \mid \forall (q, E) \in \kappa_{ndet}(\eta), \Xi(E) = 1\} \\ \cup \{\eta \in \mathcal{I}_{hist} \mid \forall (q, E) \in \kappa_{ndet}(\eta), \Xi(E) = 0\}. \quad (3.32)$$

An example is in Figure 3.18. This problem, in which the robot must decide whether its environment is simply-connected, is considered in [148].

Another direction is to study the computational requirements of robotics problems. We expect that there exist rich tradeoffs between computation time, memory usage, sensing requirements, and solution quality. Some research has been done for certain tasks, for example exploration [19, 29, 140], pursuit-evasion [86], and coverage [159], but very little is known in general. One way to deal with such issues is to study sufficient I-maps. For example, if a problem can be solved under a given robot model using a sufficient I-map into a derived I-space of finite cardinality n, the memory required to solve the problem is

$O(\log n)$. The results of Blum and Kozen [29], for example, can be characterized as showing how a discrete exploration problem can be solved in a derived I-space with cardinality linear in the height of the area to be explored, meaning that only logarithmic memory is required. These computational issues must be approached with care, especially if those computations involve real numbers [28].

Chapter 4

Localization with limited sensing

Localization, the task of systematically eliminating uncertainty in the pose of a robot, is widely regarded as a central problem in mobile robotics. A wide spectrum of sensor systems have been proposed for the localization problem, ranging from visibility sensors [47, 54, 77], to landmark detectors [15, 50, 147], to cameras [30, 56, 142], and even to optical mice [105]. How complex a sensor system does localization truly demand? In this chapter, we apply the minimalist approach to this problem, describing two simple robots with which localization is possible and a third for which localization is provably impossible.

Suppose a robot is given an accurate map of its environment, but has no knowledge of its position within that environment. The robot's goal is to move about, gathering information about its location until the uncertainty is eliminated. See Figure 4.1. We consider the localization task for three distinct robot models:

- AL – A robot equipped with angular and linear odometers. This robot can accurately rotate and translate through its environment, measuring each of these motions.
- CT – A robot equipped with with a compass and contact sensor. This robot can, using its compass, orient itself with respect to a global reference frame, then move forward until its contact sensor detects the environment boundary.
- AT – A robot equipped with an angular odometer and contact sensor. This robot can rotate with respect to a local frame and then move forward until reaching the environment boundary.

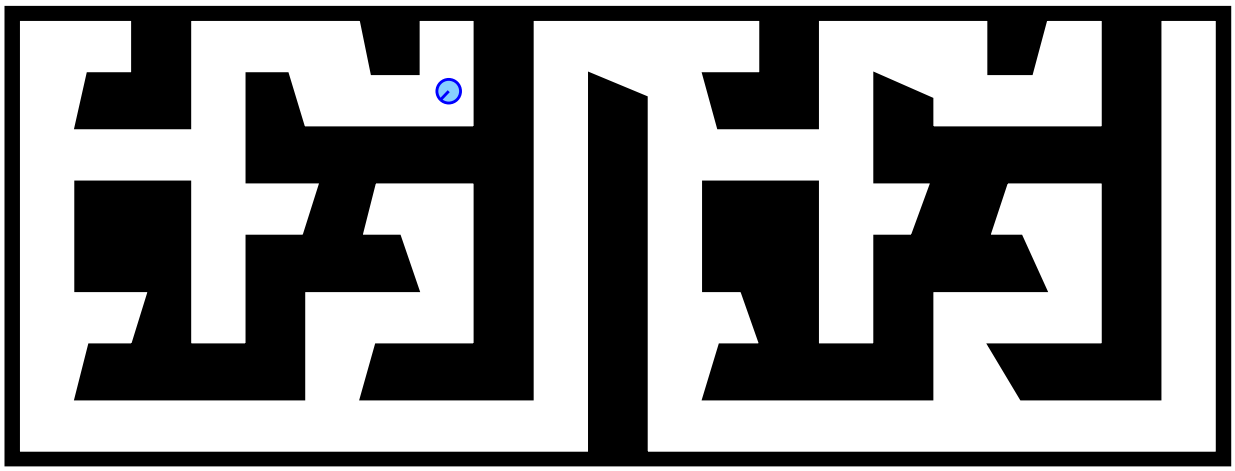


Figure 4.1: A robot in a serpentine environment. What sensing is required for the robot to eliminate uncertainty in is position?

The objective of this chapter is to classify these robots according to their ability to localize themselves. We show that AL and CT can localize themselves in polygonal environments, but AT cannot.

The motivation for this work is to identify basic sensing requirements for robotic tasks. For a given task, some robot systems are capable of completing the task whereas others are not. Our goal is to search the space of robot systems for the boundary between the "can localize" and "cannot localize" regions. This boundary gives an indication of the *necessary* conditions on robot models for localization. In this chapter we describe a very simple robot (AT) in the "cannot" set and show that small improvements to its angular sensing (CT) or linear sensing (AL) lead to models in the "can" set. See Figure 4.2.

The balance of the chapter is organized as follows. We present related work in Section 4.1. Section 4.2 formally defines our robot models and gives a problem definition. Sections 4.3 and 4.4 describe localization algorithms for robot models AL and CT, respectively. In Section 4.5 we show that no localization algorithm exists for AT. Concluding remarks appear in Section 4.6.

This work appears in its current form in [129]. Preliminary versions appear in [126] and

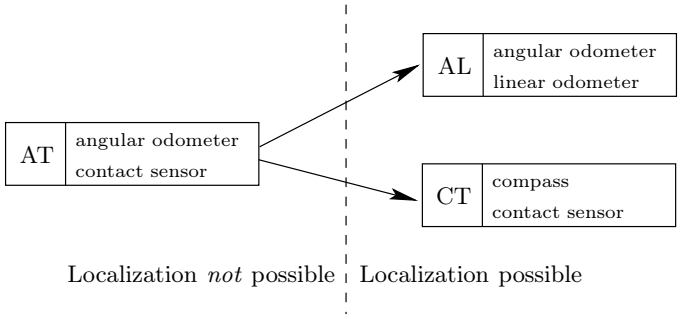


Figure 4.2: Although AL and CT have only slightly stronger sensing than AT, they are capable of localization whereas AT is not.

[127].

4.1 Related work

Localization research can be separated into two general flavors: *passive localization*, which concentrates on using any information available to the robot to draw conclusions about its position, and *active localization*, in which the goal is to prescribe motions for the robot in order to fully determine its position.

4.1.1 Passive localization

One common sensing model used in localization research is a *range sensor* that provides as input to the robot the distance to the nearest obstacle in each direction. This information can be used to compute the *visibility region* of the robot's position, which contains every point in the environment reachable by a single straight-line motion. The static problem of finding the set of candidate locations for a given visibility region in a polygonal environment was solved in [77]. Cox [47] gives an algorithm for candidate generation is given that places stronger emphasis on robustness to missing and spurious range data.

Another large body of work has focused on localization using *landmarks*. In [147], a problem is posed in which the environment contains a collection of landmark objects in fixed

locations. At any time, the robot's sensors can detect some subset of these landmarks. The robot is aware of the direction (but not distance) to each of these detected landmarks. The problem of finding the set of points in the environment consistent with this sensor data is solved for the case where the landmarks are distinguishable in [18]; the distinguishability requirement is relaxed in [15]. Others consider the problem of "landmark design" in which landmarks can be placed in locations in the environment carefully selected to facilitate localization [149]. One possible realization of this idea is to strategically place reflectors along the walls of the environment and equip the robot with a sensor that can detect the orientations of each reflector in the robot's visibility region [147]. An algorithm for computing a placement of reflectors such that no two points in the environment have identical reflector signatures appears in [50]. Betke and Gurvits give method that relies on noisy readings of bearing differences between identifiable landmarks [23]. The method of [99] is also essentially landmark-based, but the landmarks are wireless ethernet base stations whose signal strength informs the robot's position estimate. Other use vision systems to

Finally, a very large family of methods use probabilistic models to estimate the current state [49, 66, 79, 99, 107, 108, 136, 152, 153, 155, 160]. These methods employ a probability model for the robot's motion and sensing to form a distribution that represents the robot's "belief" about its current location.

4.1.2 Active localization

We now turn to methods that, rather than only reasoning about uncertainty in the robot's position, also generate motion plans to reduce or eliminate this uncertainty. Algorithms in this context are often considered in an online sense and are evaluated in terms of their *competitive ratio* [116, 145], which compares the lengths of paths generated by the algorithm to the length of the shortest possible path that could have been selected if the robot started with full information.

In [92], the environment is constrained to an embedding of a bounded-degree acyclic

graph into $\mathbb{R}^n$ with sensing limited to the orientations of incident edges. This algorithm has competitive complexity $O(n^{2/3})$, in which n is the number of leaves in the graph. Later improvements [137] shaved this to $O(n^{1/2})$, which is known to be optimal up to a constant factor[54]. Also addressed in [92] is the case where the robot can move among a collection of non-intersecting open axis-aligned rectangles in the plane; this problem is solved with a $O\left(n\sqrt{\frac{\log n}{\log\log n}}\right)$-competitive algorithm.

More generally, the problem of computing a localization strategy that minimizes the worst-case distance traveled by a robot equipped with a visibility sensor was proved NP-hard in [54]. In this case, a localization strategy has the form of a decision tree with branches at points where two or more candidate positions are disambiguated. The hardness proof proceeds by reduction from the Abstract Decision Tree problem [81]. The optimal decision tree can, however, be approximated and [54] gives an algorithm based on the visibility-cell decomposition that does this. An important weakness of this algorithm is that it relies on motion commands that direct the robot into visibility cells that may be arbitrarily small. In [134], this difficulty is addressed by introducing randomization. Other work considers the problem in the framework of approximation algorithms [93].

Building on the foundation of passive probabilistic localization techniques mentioned above, others use probabilistic techniques for active localization [65, 83, 139].

4.2 Problem statement

In this section, we formally define an active, global localization problem for robot models AL, CT, and AT.

4.2.1 Actions, transitions, and observations

Allow a point robot with orientation to move in a compact simply connected polygonal environment $E \subset \mathbb{R}^2$. Assume that the rotational symmetry group of E contains only the

identity symmetry[1]. Let ∂E denote the boundary of E, which is itself a subset of E. The robot has access to an accurate map of E, including its orientation in the plane. Since the robot's orientation is relevant, the state space is $X = E \times S^1$, in which S^1 is the set of directions in the plane, represented as unit vectors in $\mathbb{R}^2$.

For each robot model, we define a distinct action set and state transition function.

- Robot AL can, at each time step, issue either of two types of commands. First, the robot may rotate by a commanded amount. Since the robot has an angular odometer, we assume that rotation commands are executed precisely. Second, a translation command may be issued, instructing the robot to advance forward by a given distance. The actual distance traveled may be less than the commanded distance, if the robot reaches the boundary of the environment first. Formally, let $U = S^1 \sqcup [0, \infty)$ denote the robot's action space, in which the $\sqcup$ notation indicates a disjoint union operation, under which identical elements from different source sets remain distinct. Elements of S^1 denote relative rotation commands and elements of $[0, \infty)$ denote translation commands. If $u \in S^1$, then $f(x, u)$ is the appropriate change of orientation of x. If $u \in [0, \infty)$, then $f(x, u)$ computes the appropriate forward translation of x within E.

- The action space for CT is the unit circle $U = S^1$. A single $u \in U$ represents a rotation to orient the robot in a given direction, followed by a motion forward to the environment boundary. The state transition function f maps a state-action pair (x, u) to the opposite endpoint of the maximal segment in E starting at x and having direction u in the global frame. Note that because the robot has a compass, we assume it can orient itself as it wishes; therefore the current orientation (specified as part of its state) is not relevant to CT.

- The model for AT is similar to that of CT, but with the motion directions specified relative to the robot's *current* orientation, rather than with respect to a global reference

[1]This assumption is important because if E has a nontrivial symmetry group, the algorithm of Section 4.3 is effective only up to symmetry. This technicality is addressed in greater detail in Section 4.3.6.

frame. We still have $U = S^1$, but f is modified to interpret u as a motion direction relative to the robot's current heading.

The observation space and observation function also depend on the robot model.

- For AL, we must consider the feedback provided by the linear odometer. Choose $Y = [0, +\infty)$ as the observation space, in which an observation $y \in Y$ indicates that in executing the previous action, the robot's translation had magnitude y. Rotations always succeed without providing useful feedback, so $h(x, u) = 0$ when $u \in S^1$.

- Neither CT nor AT have sensors that provide useful feedback about the environment. For each, the capabilities of the sensors are instead modeled in the action sets. We assume that the compass (for CT) and the angular odometer (for AT) are used as part of a closed loop control system the correctly executes the desired rotation. Similarly, the contact sensor is used to stop the robot when it reaches the environment boundary, but does not provide sensor observations as such. Therefore, for both CT and AT, we select a dummy observation space $Y = 0$ and define $h(x, u) = 0$ for all states x and all actions u.

4.2.2 Planning in the information space

We approach the task of localization as a planning problem in the robot's nondeterministic information space $\mathcal{I}_{ndet}$, as defined in Section 2.2.2. Recall that κ_{ndet} is a sufficient I-map, so the robot does not need to retain its history I-states. As a result, throughout this chapter, we work exclusively in $\mathcal{I}_{ndet}$, using the notation η_k for the nondeterministic I-state at stage k.

Initially the robot has no knowledge of its state, so the initial information state $\eta_1 = X$ contains the entire state space. The goal region is

$$\mathcal{I}_G = \{\eta \in \mathcal{I}_{ndet} \mid |\eta| = 1\}. \tag{4.1}$$

A plan is a feedback strategy on $\mathcal{I}_{ndet}$: We want a function $\mathcal{I}_{ndet} \to U$ such that, regardless of the robot's initial state, repeatedly executing the actions chosen by this function leads in finite time to an information state in $\mathcal{I}_G$. For AL, we must specify a policy $\pi : \mathcal{I}_{ndet} \to U$. For CT and AT, there is no meaningful feedback, so it is sufficient to choose a sequence $u_1, \ldots, u_K$ of actions that eliminates the state uncertainty. We call such a sequence a *localizing sequence*.

4.3 Localization with odometry

In this section we present an algorithm to solve the localization problem described in Section 4.2, for robot model AL. Recall that AL is equipped with linear and angular odometers. An overview appears in Algorithm 4.1. The algorithm is "online" in the sense that the commands it issues depend on the observations obtained as the robot is executing. Indeed, there is no external "plan" computed ahead of time; instead we may regard Algorithm 4.1 itself as a plan in the sense that it defines a feedback strategy on the information space.

4.3.1 Algorithm overview

The algorithm tracks the robot's information state η_k throughout the execution. The first step, InitialActions, issues several commands to move from the initial condition ($\eta_1 = X$) to an information state of finite cardinality. This process is described in Section 4.3.2. For some degenerate but potentially interesting environments, InitialActions fails to generate a finite information state, instead possibly leaving one or more continua expressed as intervals on the boundary of E. The function EliminateSegments issues commands guaranteed to reach an information state devoid of such segments. This issue is dealt with in Section 4.3.3. The final section of the algorithm, detailed in Section 4.3.4, systematically reduces η_k until only a single state remains.

Algorithm 4.1 LocalizeAL(E)

```
(η_k, k) ← InitialActions(E)
(η_k, k) ← EliminateSegments(E, η_k)

while |η_k| > 1 do
  Select x_1, x_2 from η_k.
  E_{x_1} ← TransformToLocalFrame(E, x_1)
  E_{x_2} ← TransformToLocalFrame(E, x_2)
  p ← FindPointInOnlyOne(E_{x_1}, E_{x_2})
  (u_k, ..., u_{k'}) ← PathInPolygon(E_{x_1}, (0, 0), p)
  while x_1, x_2 ∈ η_k do
    y_k ← ExecuteCommand(u_k)
    η_{k+1} ← f_I(η_k, u_k, y_k)
    x_1 ← f(x_1, u_k)
    x_2 ← f(x_2, u_k)
    k ← k + 1
  end while
end while

return η_{k-1}
```

4.3.2 Generating a finite set of candidates

This section describes a technique for reaching an information state of finite cardinality. The central idea is to make two motions between points on the boundary of the environment, separated by a 90° turn. We show that if the environment has no pair of parallel edges, only finitely many states are consistent with such a sequence of motions. Section 4.3.3 addresses the more troublesome case when the environment violates this condition.

The robot, starting with no knowledge of its position, makes several motions:

1. Move forward until reaching the boundary.

2. Rotate 180°, then move forward until reaching the boundary. Let d_1 denote distance traveled on this motion.

3. Rotate 90°, then move forward until reaching the boundary. If the robot reaches the boundary immediately, rotate 180° and try again. Let d_2 denote distance traveled on this motion.

The commands to "move until reaching the boundary" can be realized by selecting a translation amount larger than the diameter of E. In order to continue in final step, the robot must make a net rotation of either 90° or -90°, depending on its angle of incidence with the boundary. Except when the robot reaches an environment vertex, at least one of these rotations allows the robot to continue. If the robot knows it has reached an environment vertex, then there are already only finitely many candidates. The use of 90° rotations is motivated by the simplifications it affords in Equation 4.12. In principle, rotations of other amounts would work equally well.

The problem remains to find the set of states consistent with these initial motions. For simplicity, we ignore the first translation and instead consider only the two boundary-to-boundary translation with lengths d_1 and d_2. A geometric interpretation of the problem is perhaps helpful here:

> Given E and the two odometer readings d_1 and d_2, we want to find all ways to pack into E a 2 link polygonal chain with edges having lengths d_1 and d_2 joined at a right angle, such that the initial and final endpoints rest on different boundary edges from the middle vertex.

The set of final endpoints of these chains can be used directly to compute a set of candidate states of the robot. Figure 4.3 shows an example.

Generating candidates for three fixed edges

The robot's initial motions visit three environment edges. Suppose these three edges $\overline{p_1p_2}$, $\overline{p_3p_4}$, and $\overline{p_5p_6}$, and the order in which the robot visits them are fixed. Let $p_a \in \overline{p_1p_2}$, $p_b \in \overline{p_3p_4}$, and $p_c \in \overline{p_5p_6}$ denote the three boundary points visited by the robot. See Figure 4.4.

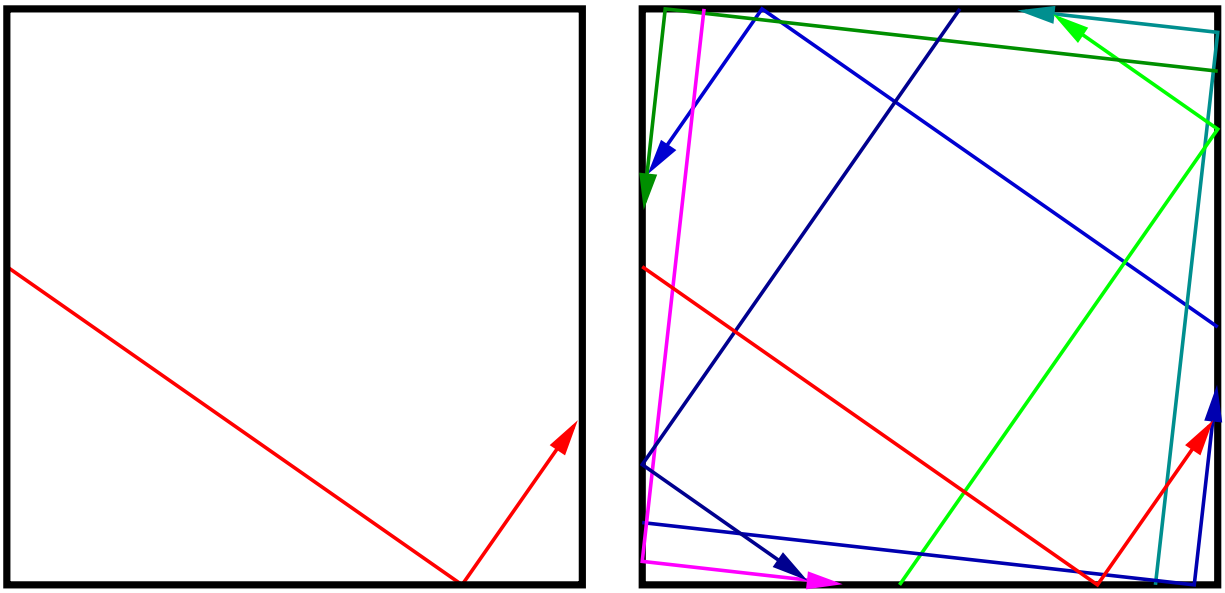


Figure 4.3: [left] Two boundary-to-boundary motions in a square shaped environment, separated by a turn of 90°. [right] The 8 possibilities for these motions in this environment.

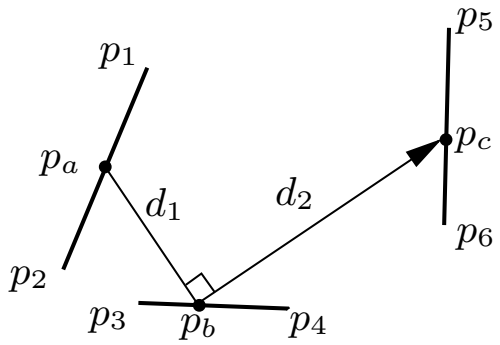


Figure 4.4: Three fixed segments $\overline{p_1p_2}$, $\overline{p_3p_4}$, and $\overline{p_5p_6}$ and translations of length d_1 and d_2 between them.

First, parameterize these three points as follows:

$$p_a = (1-a)p_1 + ap_2 \tag{4.2}$$

$$p_b = (1-b)p_3 + bp_4 \tag{4.3}$$

$$p_c = (1-c)p_5 + cp_6. \tag{4.4}$$

The first motion has length d_1, therefore $||p_a - p_b|| = d_1$. Expanding from the parameterization above gives a quadratic constraint in a and b:

$$Aa^2 + Bab + Cb^2 + Da + Eb + F = 0 \tag{4.5}$$

with constant coefficients

$$\begin{aligned}
A &= (x_2 - x_1)^2 + (y_2 - y_1)^2 && (4.6)\\
B &= -2(x_2 - x_1)(x_4 - x_3) - 2(y_2 - y_1)(y_4 - y_3) && (4.7)\\
C &= (x_4 - x_3)^2 + (y_4 - y_3)^2 && (4.8)\\
D &= -2(x_3 - x_1)(x_2 - x_1) - 2(y_3 - y_1)(y_2 - y_1) && (4.9)\\
E &= 2(x_3 - x_1)(x_4 - x_3) + 2(y_3 - y_1)(y_4 - y_3) && (4.10)\\
F &= (x_3 - x_1)^2 + (y_3 - y_1)^2 - d_1^2, && (4.11)
\end{aligned}$$

in which we use the convention that $p_i = (x_i, y_i)$. We also know that p_c must be distance d_2 from p_b, and that $p_b - p_c$ must be perpendicular to $p_a - p_b$. These constraints are satisfied when

$$p_c - p_b = s_1 \frac{d_1}{d_2} (p_b - p_a)^{\perp} \tag{4.12}$$

in which s_1 is either -1 or $+1$, depending on whether its net rotation was $90°$ or $-90°$ in step 3 above. This vector equation can be separated into a pair of scalar linear equations in a, b, and c. Eliminating c yields a single linear equation in a and b:

$$Ga + Hb + I = 0 \tag{4.13}$$

with constant coefficients

$$G = \frac{\frac{d_2}{d_1}(y_2 - y_1)}{x_5 - x_6} + \frac{\frac{d_2}{d_1}(x_2 - x_1)}{y_5 - y_6} \tag{4.14}$$

$$H = \frac{(x_4 - x_3) - \frac{d_2}{d_1}(y_4 - y_3)}{x_5 - x_6} - \frac{(y_4 - y_3) + \frac{d_2}{d_1}(x_4 - x_3)}{y_5 - y_6} \tag{4.15}$$

$$I = \frac{(x_3 - x_5) - \frac{d_2}{d_1}(y_3 - y_1)}{x_5 - x_6} - \frac{(y_3 - y_5) + \frac{d_2}{d_1}(x_3 - x_1)}{y_5 - y_6}. \tag{4.16}$$

Note that if either denominator is 0 (corresponding to $\overline{p_5p_6}$ being horizontal or vertical), the system can be solved trivially. Equations 4.5 and 4.13 form a linear-quadratic system in a and b. Barring degeneracies, this system has at most two solutions, which can be found analytically by standard methods.

The method described above gives candidate values for a, b, and c. Candidates for which any of a, b, or c are outside the interval $[0, 1]$ should be discarded, because they correspond to endpoints outside of $\overline{p_1p_2}$, $\overline{p_3p_4}$, or $\overline{p_5p_6}$ respectively. The final state (that is, position-orientation pair) of the robot resulting from such a candidate is $(p_c, \text{atan}(y_c - y_b, x_c - x_b))$.

Lastly, note that if $d_1 = 0$ or $d_2 = 0$, then the robot knows that its position is at some convex vertex of E. This does not, however, eliminate the uncertainty in the robot's orientation. In order to determine its orientation, the robot must move away from the vertex. To do so, the robot must rotate and attempt translations, at most $360/\theta$ times, in which θ denotes the measure of the smallest interior angle in E, measured in degrees.

Generating candidates over all of E

The previous section showed how to find candidate solutions, given d_1, d_2 and three fixed environment edges to be visited in sequence. Candidate positions over the complete environment can be computed by iterating over each ordered triple of environment edges. Since we must admit the case where $\overline{p_1p_2} = \overline{p_5p_6}$, there are $n(n-1)(n-1)$ such triples. The at most 2 candidates for each can be computed in constant time. In practice, the performance

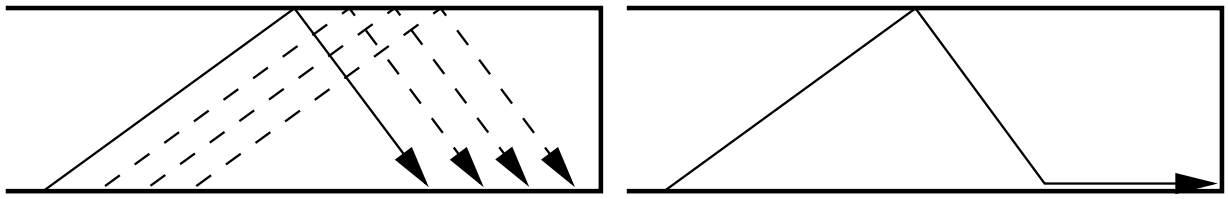


Figure 4.5: [top] Parallel edges of the environment admit continua of candidate states. [bottom] A motion parallel to one of these segments leaves only a single candidate point.

of this process may possibly be improved by a preprocessing step which, for each pair of environment edges, computes the minimum and maximum distances between mutually visible points on these edges. This information can be used to filter some edge triples as infeasible without explicit consideration.

As a final step, the candidate list must be pruned, retaining only those candidates that represent motions that lie entirely within E. For each, it is sufficient to ensure that $\overline{p_a p_b}$ and $\overline{p_b p_c}$ are fully contained in E. In a simple polygon, data structures are known to answer such queries in $O(\log n)$ time, with $O(n)$ preprocessing time and $O(n)$ space [39]. This final candidate set becomes the robot's information state η_k.

4.3.3 If some boundary edges are parallel

Although the preceding exposition made the assumption that the environment contains no pair of parallel edges, environments of practical interest often contain parallel edges. In particular, note the case where the environment contains a narrow strip bounded by two parallel edges. This situation would arise, for example, in a indoor corridor or narrow room. When parallel edges exist, continua of final states may be consistent with the robot's initial motions. See Figure 4.5. Each of these continua can be eliminated with a motion parallel to itself.

4.3.4 Localization from a finite set

The previous sections showed how to select actions to guarantee that η_k contains only finitely many states. How can we select additional actions to determine the robot's true position from among these candidates? The approach is to select two candidates and choose motions that are guaranteed to disambiguate them. More precisely, we want choose two states x_1 and x_2 from η_k and choose actions $u_k, \ldots, u_{k+j}$ so that η_{k+j+1} contains either x_1 or x_2 (or neither) but not both.

In Sections 4.3.2 and 4.3.3, we described a method for reaching an information state representable by a finite union of single states. Given an information state η_k, an action u_k, and an observation y_k, how can we compute the resulting information state $\eta_{k+1} = f_{\mathcal{I}}(\eta_k, u_k, y_k)$? We do so in two stages. First, we find the forward projection of η_k under action u_k, by ray shooting in E. Then we prune from the result any states for which the distance traveled differs from y_k using a simple constant time procedure.

For a given state x, let E_x denote the environment E transformed into so that the robot rests at the origin and faces the positive x-axis. Note that $(0,0) \in E_x$ if and only if the position portion of x is contained within E in the global frame.

Select x_1 and x_2 arbitrarily from η_k. Compute E_{x_1} and E_{x_2} and overlay them. See Figure 4.6. In this overlay, rotation and translation commands affect both x_1 and x_2 in the same way; we can choose a destination position in this frame and command actions that to navigate both x_1 and x_2 to this point in their respective local frames.

Since E has no nontrivial rotational symmetries, we have $E_{x_1} \neq E_{x_2}$. Therefore, there must exist some position p in E_{x_1} but not in E_{x_2}. Plan a path in E_{x_1} from $(0,0)$ to p. Since $(0,0) \in E_{x_2}$ but $p \notin E_{x_2}$, this path must cross the boundary of E_{x_2} at least once. The translation action corresponding to this crossing of the boundary of E_{x_2} necessarily distinguishes between x_1 and x_2. If the robot began at x_1, its odometry reading at this step will be greater than if it had begun on x_2. One of the two can be pruned after this step. A

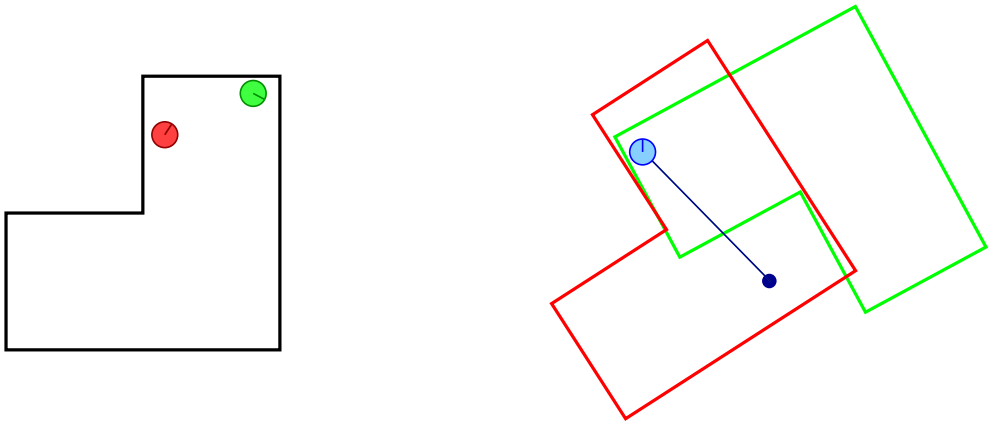


Figure 4.6: [left] Two states in an L shaped environment. [right] Two overlaid copies of the environment shown in the local frame of those states. Attempting to execute the path shown (which consists of one rotation and one translation) shown will result in different odometry readings for these two states.

third possibility is that both candidates are pruned before or during this step. This could happen if the robot's true state is neither x_1 nor x_2, but some third state in η_k. In this case, the remaining actions in the plan can be discarded, and new choices for x_1 and x_2 can be made from the reduced η_{k+1}.

Which path should the robot follow within E_{x_1} to reach p from $(0, 0)$? To disambiguate x_1 and x_2 requires only a path that stays within E_{x_1} but leaves E_{x_2}. Our implementation uses the *shortest path* between $(0, 0)$ and p, which can be computed in time $O(n)$ [75, 78]. Also of potential interest is the *minimum link path* [120], which minimizes the number of robot commands. The minimum link path can also be computed in time $O(n)$. In any case, a piecewise linear path in E_{x_1} can be trivially converted to a sequence of alternating translation and rotation commands.

4.3.5 Complexity

Let n denote the number of edges in E. In InitialActions, we execute fewer than $O(n^3)$ ray shooting queries, each taking time $O(\log n)$, so this step takes $O(n^3 \log n)$ time to generate $O(n^3)$ initial candidates. Let r denote the number of such candidates. If E has parallel

edges, each segment returned by InitialActions takes time $O(n \log n)$ to compute.

The outer while loop in Algorithm 4.1 eliminates at least one candidate in each iteration, so there are at most $r-1$ iterations. There are fewer than $r-1$ iterations if some candidates are pruned as a side-effect of distinguishing x_1 and x_2. The run time of each iteration is dominated by the time to compute $f_{\mathcal{I}}$, which is $O(r \log n)$. This computation must be done at each of the $O(n)$ steps of the of the path generated at each iteration. Therefore, the total computation time for the algorithm is $O((n^3 + r^2 n) \log n) = O(n^7 \log n)$.

It is possible that these bounds can be improved. The question remains unanswered whether $r = \Theta(n^3)$. Our informal experiments suggest that in practical situations, both r and the number of disambiguation iterations often fall far short of the upper bounds we present here.

4.3.6 Environment symmetries

We have thus far assumed that E has no nontrivial rotational symmetries. This is important in Algorithm 4.1 to ensure that there exists at least one point p in E_{x_1} but not in E_{x_2}. If this assumption does not hold, then we can still consider the problem of localization *up to symmetry*. This section makes the notion of localization up to symmetry more precise.

Definition 4.1 *A* symmetry *is function composed of rigid translations and rotations mapping E onto itself. Without ambiguity we can extend such a function to X by applying the appropriate change of orientation. Two states $x_1, x_2 \in X$ are* symmetric *if there exists a symmetry under which $x_1 \mapsto x_2$.*

Figure 4.7 shows several environments with varying numbers of symmetries. The number of symmetries of X can be computed in $O(n)$ time [163]. The following lemma will be useful for showing the relevance of these symmetries to localization.

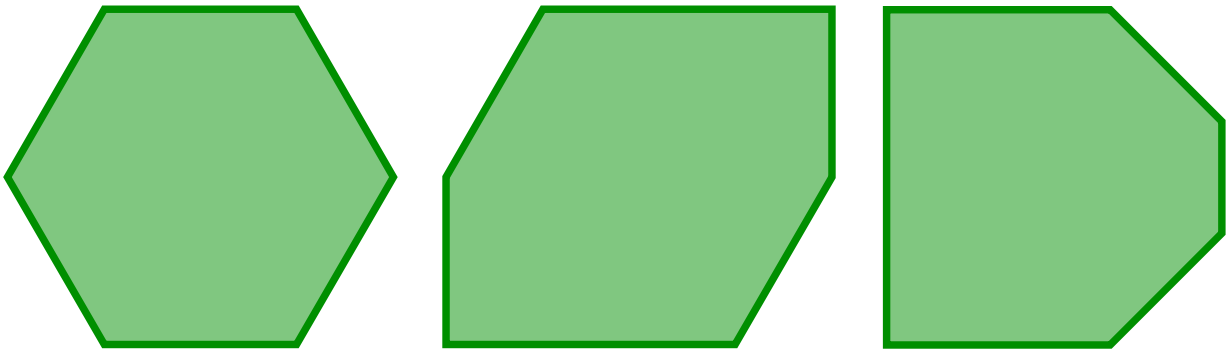


Figure 4.7: Sample environments with, from left to right, 6, 2, and 1 rotational symmetries.

Lemma 4.1 *The relation of symmetry between states is an equivalence relation, which we denote* $\sim$*. Each equivalence class of* $\sim$ *contains one state for each symmetry of the environment.*

Proof: Observe that the symmetries of a polygon form a group under function composition. In particular the identity is always a symmetry, and the set of symmetries is closed under composition and inverse. The reflexivity, transitivity, and symmetry of the $\sim$ relation all follow immediately. □

Now we show that AL cannot distinguish between symmetric states.

Lemma 4.2 *Consider an action sequence* $u_1, \ldots, u_{k-1}$*, an observation sequence* $y_1, \ldots, y_{k-1}$ *and the resulting information state* η_k*. For any* $x \in \eta_k$ *and* $x' \in X$ *with* $x \sim x'$*,* $x' \in \eta_k$*.*

Proof: Since $x \in \eta_k$, there exists some initial state x_1 for which executing $u_1, \ldots, u_{k-1}$ leads to x and generates $y_1, \ldots, y_{k-1}$. Since $x \sim x'$, there exists a symmetry τ under which $x' = \tau(x)$. But f acts only locally, so we know that a robot starting from $\tau(x_1)$ and executing $u_1, \ldots, u_{k-1}$ has state

$$f(\tau(x_1), u_1, \ldots, u_j) = \tau(f(x_1, u_1, \ldots, u_j)) = \tau(x) = x'. \tag{4.17}$$

Moreover, the observation sequences are identical, because the boundary edges of E are affected by τ in the same way as x_1 is. Consequently, $\tau(x_1)$ is an initial state that leads

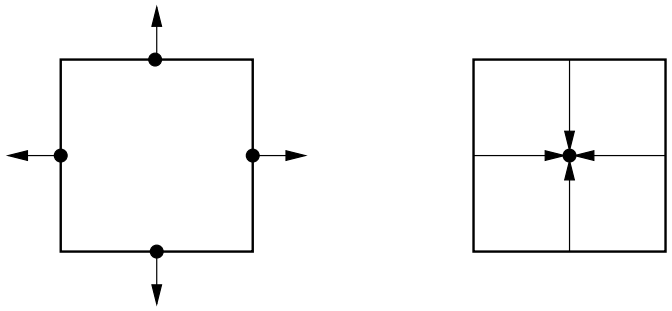


Figure 4.8: With nontrivial symmetries, the robot can reach a known position, but is unable to fully determine its orientation. [left] Four symmetric states in a square environment. [right] Motions from those symmetric states to a position fixed by the symmetries.

to x', thereby demonstrating that x' is consistent with $u_1, \ldots, u_{k-1}$ and $y_1, \ldots, y_{k-1}$. Hence $x' \in \eta_k$. □

The practical importance of this lemma is that for AL, the localization task can only be accomplished modulo the symmetries in the environment. No sequence of actions and observations can distinguish between a pair of symmetric states. Symmetry plays a similar role in some methods for part orientation [72]. We define the task of *localization up to symmetry*:

> Given E, select actions to reduce the robot's information state to a set of symmetric states.

Note that Algorithm 4.1 can be adapted for localization up to symmetry. The only modifications needed are to change the termination condition to stop when $|\eta_k|$ is equal to the number of symmetries, and to ensure that the states selected as x_1 and x_2 are not themselves symmetric. The rest of the algorithm remains unchanged.

The limitations arising from symmetry are no longer relevant if we are concerned only with determining the robot's *position* and are not interested in its final orientation. In this case we can, after reaching an information state consisting of symmetric points, issue additional commands to navigate to a point fixed by the environment's symmetries. See Figure 4.8. We will not revisit this variant problem.

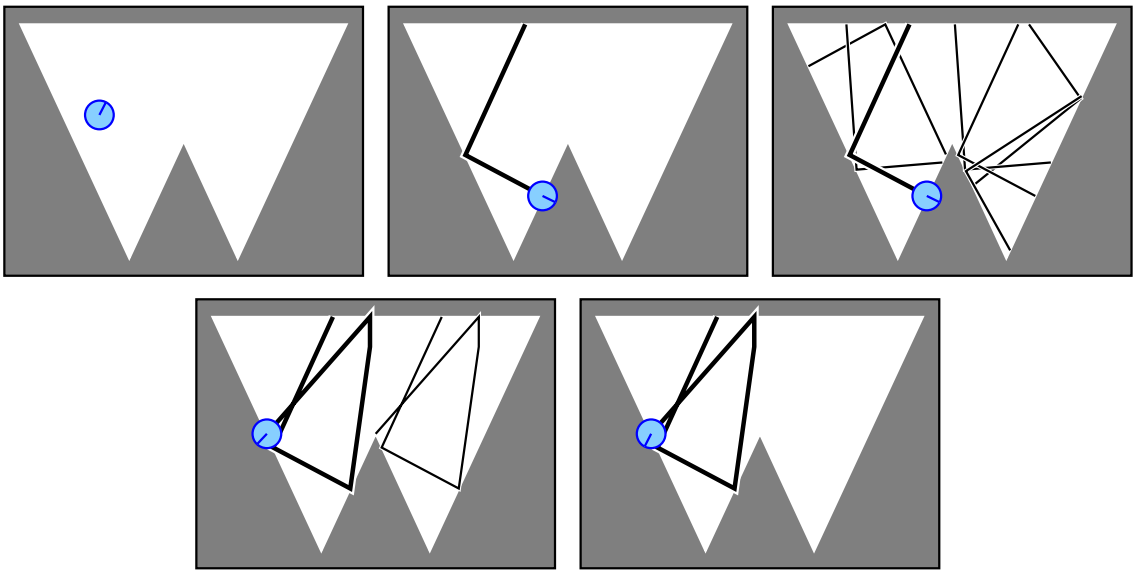


Figure 4.9: A sample execution of Algorithm 4.1 generated by our implementation in approximately 0.03 seconds. Top row: (a) The robot in its initial state. (b) The motions generated by InitialActions. (c) There are 7 states consistent with these initial motions, so $|\eta_6| = 7$. Bottom row: (d) One disambiguation results in $|\eta_{12}| = 2$. (e) The robot is fully localized after 13 commands, with final information state $|\eta_{14}| = 1$.

4.3.7 Computed examples

To illustrate its effectiveness, we have implemented Algorithm 4.1 in simulation, using simplified methods for many of the geometric computations. The implementation is in C++ on a 2.5GHz GNU/Linux system. Figure 4.9 shows a simple example in which the robot makes 13 motions to localize itself. In Figure 4.10, the environment is a regular pentagon, so the final information state contains one state for each of the 5 symmetries. The environment depicted in Figure 4.11 is serpentine and self similar, but has no symmetries.

This environment has 88 edges and geodesic diameter 65 meters. To gauge the efficiency of our implementation, we selected at random 100 states an executed the localization algorithm starting at each. The results of these runs are summarized in Table 4.1. These experiments indicate that in at least some non-pathological situations, the algorithm's performance is significantly better than the upper bounds in Section 4.3.5 might suggest.

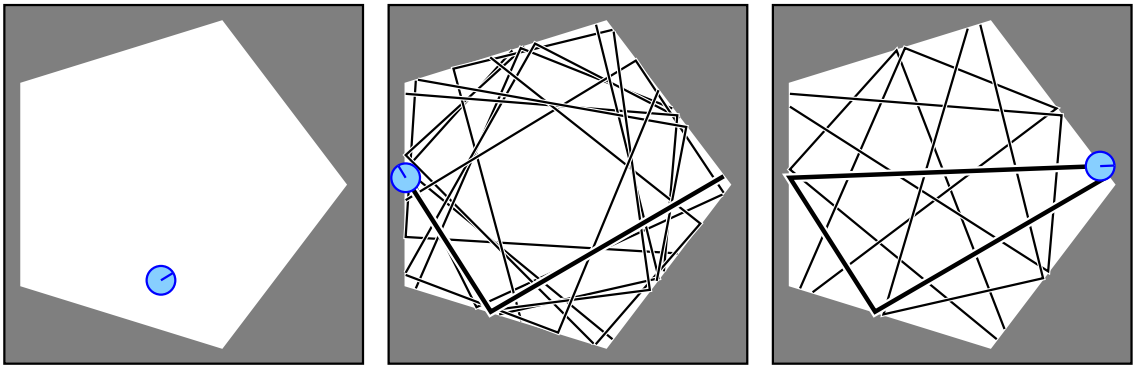


Figure 4.10: A robot localizing itself in an environment with 5 symmetries. From top to bottom: (a) The robot's initial state. (b) Executing InitialActions results in an information state η_8 containing 15 states. (c) One disambiguation iteration fully localizes the robot, leaving 5 states in η_{10}. Our implementation took approximately 0.1 seconds to solve this problem.

Table 4.1: Experimental results on the performance of Algorithm 4.1. One hundred initial states were randomly selected from the state space of the environment depicted in Figure 4.11.

	minimum	mean	maximum
Distance Traveled (m)	11.45	40.97	64.09
Initial Candidates	3	42.11	103
Actions Executed	9	21.84	45
Computation Time (s)	2.59	5.12	9.26

4.4 Localization with a compass and contact sensor

Having addressed the localization task for AL, we now consider CT, a robot equipped with only a compass and contact sensor. Once again we show constructively that the localization task can be completed. A simple example of our algorithm's execution appears in Fig. 4.12.

Recall that each action $u \in S^1$ represents a rotation to the given orientation, followed by a forward motion to the environment boundary. After its first action, the robot knows its true orientation. Also note that after the first motion, the robot's translations are all between points on the environment boundary. For these reasons, we can simplify the robot's state space to ∂E, ignoring orientation and the interior of E. In this context, the information

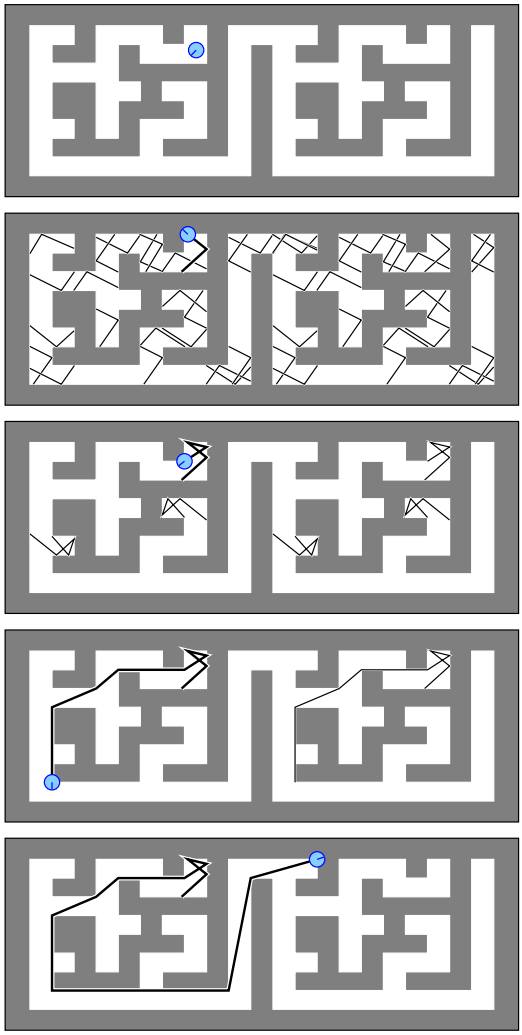


Figure 4.11: A robot localizing itself in a serpentine environment. From top to bottom: (a) The robot's initial state. (b) Executing InitialActions results in an information state η_6 containing 48 states. (c) After 2 iterations of the disambiguation algorithm, only 6 states remain in η_{10}. (d) There are only two states in η_{20}. (e) The robot is fully localized after 25 motions. Our implementation took approximately 3.8 seconds to solve this problem.

states are subsets of ∂E. We use this simplification throughout Sections 4.4 and 4.5.

4.4.1 Computing the information transition function

This section presents an algorithm for computing $f_{\mathcal{I}}(\eta, u)$ given E, η and u. We restrict our attention to information states that can be reached from the initial state $\eta_1 = \partial E$.

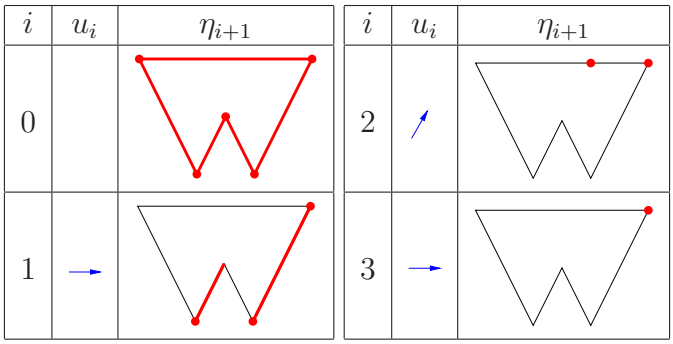


Figure 4.12: A localizing sequence generated by Alg. 4.2 for CT in a nonconvex polygon. The information state at each step is shaded. Compare to Fig. 4.9.

Consider an information state η that can be expressed as the union of a finite collection $s_1, \ldots, s_l$ of open segments and a finite set of points $p_1, \ldots, p_m$ on ∂E. To be precise, each s_i is a linear subset of ∂E not containing its endpoints. Each s_i need not be a complete edge of ∂E and since it is linear, cannot contain any vertex of ∂E. Without loss of generality, assume that the s_i's are pairwise disjoint. The next lemma shows that every reachable information state can be expressed in this form.

Lemma 4.3 *Every information state η reachable from ∂E by an action sequence $u_1, \ldots, u_k$ can be expressed as a finite union of open segments and points on ∂E.*

Proof: Use induction on k. When $k = 0$, $\eta = \partial E$, which is the union of the vertices and edges bounding E. Assume inductively that η_{k-1} can be expressed as a finite union of open segments and points. Because $f_{\mathcal{I}}$ maps each segment to a finite set of polygonal chains on ∂E and each point to another single point, η_k also has a representation as a finite set of points and segments. □

The intuition is that, given an action u and an information state η described as a finite union of points and segments, the resulting information state $f_{\mathcal{I}}(\eta, u)$ is simply the projection of those points and segments onto ∂E in direction u. For a point, this projection is a simple ray-shooting query. For a segment $\overline{ab}$, compute the projection by sweeping line parallel to u from a to b, generating a new segment each time the point on ∂E intersecting l closest to $\overline{ab}$

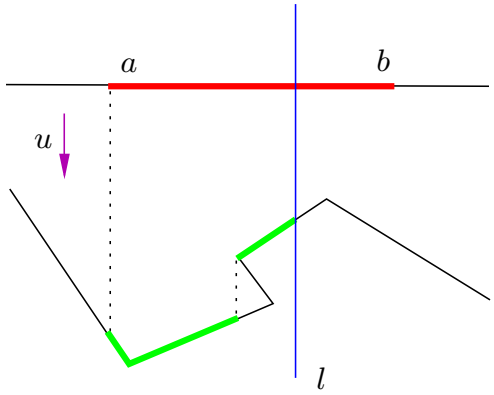


Figure 4.13: Computing $f_{\mathcal{I}}(\overline{ab}, u)$ by a line sweep algorithm. The diagram shows a snapshot of the algorithm as it runs. The sweep line l moves from left to right.

is a vertex of E. See Fig. 4.13. The time to perform this computation is $O((m + nl) \log n)$ for an information state described by m points and l segments in an environment with n vertices.

4.4.2 Algorithm overview

We now present the localization algorithm itself. The algorithm proceeds in two parts. First, actions are selected which reduce the uncertainty in the robot's position to a finite set of possibilities. Second, additional actions are chosen to reduce the uncertainty from this finite set to a single point. The complete localizing sequence $u_1, \ldots, u_K$ is divided into two parts $u_1, \ldots, u_{K_1}$ and $u_{K_1+1}, \ldots, u_{K_2}$ generated by the respective parts of the algorithm. The complete algorithm is shown in Algorithm 4.2.

4.4.3 From all the entire environment boundary to a finite subset

This section presents a sweep line algorithm for computing a sequence of actions to reduce the robot's information state to a finite set of points. The following lemma, whose intent is illustrated in Figure 4.14, provides the basis for the algorithm.

Algorithm 4.2 LocalizeCT(E)

$\eta_1 \leftarrow \partial E$
$k \leftarrow 1$
while η_k contains at least one segment **do**
  $\overline{ab} \leftarrow$ LeftmostSegment(η_k)
  if $(a-b).x > 0$ **then**
    $u_k \leftarrow (a-b)/||a-b||$
  else
    $u_k \leftarrow (b-a)/||b-a||$
  end if
  $\eta_{k+1} \leftarrow f_{\mathcal{I}}(\eta_k, u_k)$
  $k \leftarrow k+1$
end while

while η_k contains at least two points **do**
  Select p, q from η_k.
  $p_k \leftarrow p, q_k \leftarrow q$
  while $q_k \notin \text{Vis}(p_k, E)$ **do**
    $t_k \leftarrow$ first vertex of shortest path from p_k to q_k
    $u_k \leftarrow (t_k - p_k)/||t_k - p_k||$
    $\eta_{k+1} \leftarrow f_{\mathcal{I}}(\eta_k, u_k)$
    $p_{k+1} \leftarrow$ ShootRay(E, p_k, u_k)
    $q_{k+1} \leftarrow$ ShootRay(E, q_k, u_k)
    $k \leftarrow k+1$
  end while
  $u_k \leftarrow (q_k - p_k)/||q_k - p_k||$
  $\eta_{k+1} \leftarrow f_{\mathcal{I}}(\eta_k, u_k)$
  $k \leftarrow k+1$
end while

return $(u_1, \ldots, u_{k-1})$

Lemma 4.4 *For any segment $s = \overline{ab} \subset E$, $f_{\mathcal{I}}(s, u)$ is a single point if and only if $u = (a-b)/||a-b||$ or $u = (b-a)/||b-a||$.*

Proof: For the forward part, note that since $\overline{ab}$ is contained in E and is therefore itself collision free, the maximal collision free segment starting from each $x \in \overline{ab}$ is the same. Hence each $x \in \overline{ab}$ maps to the same point under f. For the backward part, suppose u is not parallel to $\overline{ab}$ and $f_{\mathcal{I}}(\overline{ab}, u)$ is a single point. Then a, b, and $f_{\mathcal{I}}(\overline{ab}, u)$ form a nondegenerate triangle. This is a contradiction because by definition of f, we must have $\overline{ax}$ parallel to $\overline{bx}$. □

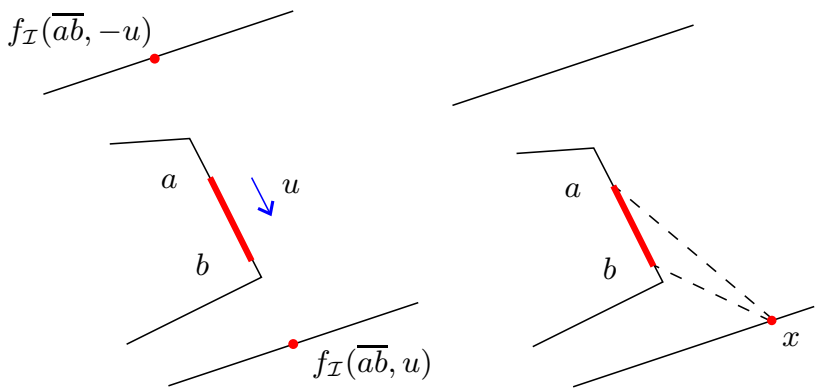


Figure 4.14: [left] A motion along $\overline{ab}$ collapses $\overline{ab}$ to a single point. [right] No motion not parallel to $\overline{ab}$ can collapse $\overline{ab}$.

Starting with $\eta_1 = \partial E$, the algorithm maintains a "current" information state η_k and a sequence of actions $u_1, \ldots, u_{k-1}$ mapping η_1 to η_k. Computation proceeds by sweeping a vertical line l from left to right across E, maintaining the invariant that η_k has no segments on the left side of l. Each time l reaches the endpoint of a segment $\overline{ab}$ in η_k, the sweep line stops and the algorithm selects as u_k whichever of $(a-b)/||a-b||$ and $(b-a)/||b-a||$ has nonnegative x coordinate. The resulting $\eta_{k+1} = f_{\mathcal{I}}(\eta_k, u_k)$ maintains the sweep invariant because the x-component of the motion of each segment in η_k is nonnegative; hence, no segment can cross l. When l passes the rightmost vertex of E, it is certain that no segments remain in η_k. It remains to show that this method generates a plan of finite length.

Lemma 4.5 *The above algorithm generates $K_1 = O(n^3)$ actions for an environment with n edges.*

Proof: Let $e_1, \ldots, e_n$ denote the edges of ∂E and let $v(e_i)$ denote a unit vector parallel to e_i and oriented so that its x component is nonnegative. For a fixed i and j, $f_{\mathcal{I}}(e_i, v(e_j))$ is a set of polygonal chains on ∂E with total complexity $O(n)$. Let R_{ij} denote the set of endpoints of segments in $f_{\mathcal{I}}(e_i, v(e_j))$ and let $R = \bigcup_{i,j} R_{ij}$. Observe that $|R| = O(n^3)$. Clearly every segment s reached by l is in the initial condition η_1, or is a subset of some $f_{\mathcal{I}}(e_i, v(e_j))$. There are n segments in η_1 and R is a set of earliest possible points at which an information state

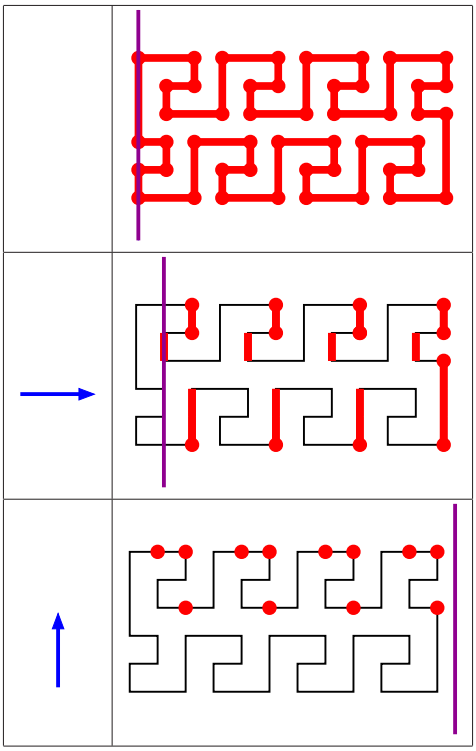


Figure 4.15: A sample execution of the first half of Algorithm 4.2.

segment projected from another edge may begin. These events are sufficient to maintain the sweep invariant, so $K_1 = O(n) + O(n^3) = O(n^3)$. □

An example execution of this procedure appears in Figure 4.15.

4.4.4 From a finite subset to a single point

The previous section showed how to select actions $u_1, \ldots, u_{K_1}$ that map $\eta_1 = \partial E$ to a finite set $\eta_{K_1} = \{p_1, p_2, \ldots, p_m\}$ of points on ∂E. It remains to generate additional actions $u_{K_1+1}, \ldots, u_{K_2}$ mapping $\{p_1, p_2, \ldots, p_m\}$ to a single point. We derive this part of the algorithm by reduction to the special case when $m = 2$. The more general problem for m points can be solved by iterating the algorithm for two points.

Let $\eta = \{p, q\}$. The ordering of the points is arbitrary but must be fixed. Our goal is to

design a sequence of actions $u_{K_1+1}, \ldots, u_{K_2}$ such that

$$f(p, u_{K_1+1}, \ldots, u_{K_2}) = f(q, u_{K_1+1}, \ldots, u_{K_2}). \tag{4.18}$$

That is, we want an action sequence mapping p and q to the same destination. For $K_1 < k \leq K_2$, let

$$p_k = f(p, u_{K_1+1}, \ldots, u_k)$$

and likewise

$$q_k = f(q, u_{K_1+1}, \ldots, u_k).$$

Our algorithm selects u_k using only p_k and q_k. We begin with the simple base case:

Lemma 4.6 *If $\overline{p_k q_k} \subset E$, then the action $u = (q_k - p_k)/||q_k - p_k||$ is a localizing sequence for $\{p_k, q_k\}$.*

Proof: Follows from Lemma 4.4 with $a = p_k$ and $b = q_k$. □

The intuition is that if p_k can "see" q_k in the sense that there is an unobstructed path between them, then a motion in the direction of this path maps both p_k and q_k to the same place.

Now suppose $\overline{p_k q_k} \not\subset E$. The following definition is useful in this case.

Definition 4.2 *For any $x \in E$, let* $\mathrm{Vis}(x, E)$ *denote the* visibility polygon *[12] of x in E, defined as*

$$\mathrm{Vis}(x, E) = \{x' \in E \mid \overline{xx'} \subset E\}. \tag{4.19}$$

We follow [77] in characterizing the boundaries visibility polygons in terms of non-spurious edges which are parts of ∂E and spurious edges which are not. Observe that since E is simply connected, the spurious edges subdivide E in such a way that every point $x' \notin \mathrm{Vis}(x, E)$ can be associated with exactly one spurious edge such that the shortest path

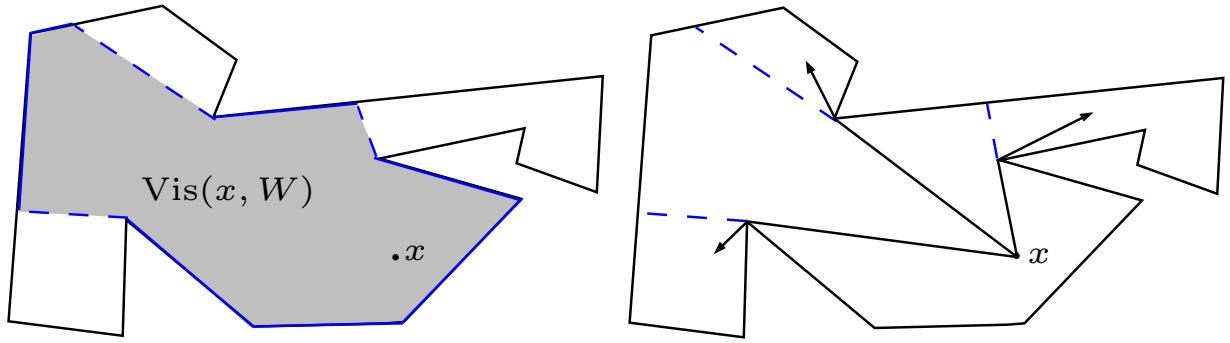


Figure 4.16: [left] A visibility polygon. Spurious edges are dashed. [right] The shortest path to any point not in the visibility polygon begins with a motion in the direction of a spurious edge.

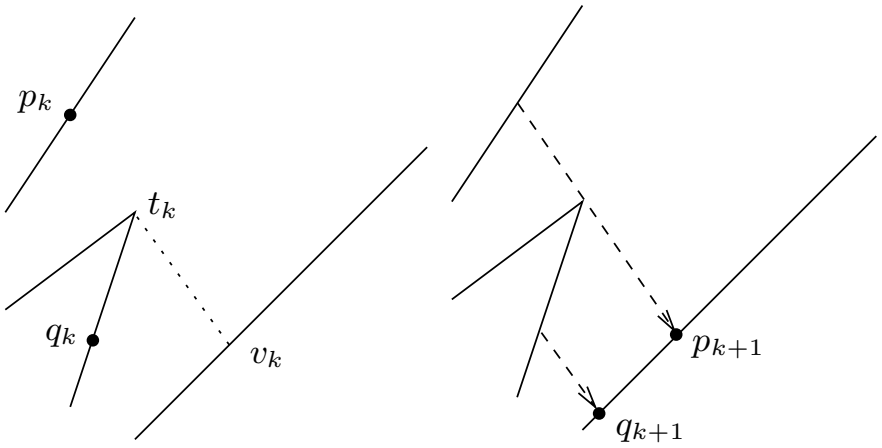


Figure 4.17: [left] The spurious edge $\overline{t_k v_k}$ hides p_k from q_k. [right] The point q_{k+1} cannot cross $\overline{t_k v_k}$ because its motion is parallel to $\overline{t_k v_k}$.

from x to x' crosses this spurious edge. Further, the first segment of the shortest path from x to x' is parallel to this spurious edge. See Figure 4.16. Let $\overline{t_k v_k}$ denote the spurious edge crossed by the shortest path from p_k to q_k.

Assume momentarily that $\overline{t_k v_k}$ is not a bitangent of E. Choose $u_k = (t_k - p_k)/||t_k - p_k||$. That is, select a motion in the direction of the spurious edge that hides q_k from p_k. Figure 4.17 illustrates this selection (and the intuition behind the proof of Lemma 4.7). This completes

the definition of our action sequence $u_{K_1+1}, \ldots, u_{K_2}$:

$$u_i = \begin{cases} (q_i - p_i)/||q_i - p_i|| & \text{if } q_i \in \text{Vis}(p_i, E) \\ (t_i - p_i)/||t_i - p_i|| & \text{otherwise} \end{cases}, \tag{4.20}$$

in which K_2 is the minimal i for which the first case applies. Clearly if K_2 exists, then this action sequence is a localizing sequence. It remains only for us to show that K_2 exists.

Let $Q_k = E - \bigcup_{i=K_1,\ldots,k} \text{Vis}(p_i, E)$ and observe that $Q_{k+1} \subset Q_k$. Informally, Q_k is the portion of E that p has never seen.

Lemma 4.7 *For all $k > K_1$, $q_k \in Q_k$.*

Proof: Use induction on k. The statement is true by construction when $k = K_1$. For the inductive step, note that q_k moves parallel to $\overline{t_k v_k}$, so that q_{k+1} is still behind this spurious edge. If $q_k \notin Q_k$, then q_k must be in a region visible to some p_i, or in some region not seen by any p_i but separated from q_k by $\overline{t_k v_k}$. In either case, we can form a nontrivial loop in E, contradicting the simply connected property of E. □

One informal way to understand Lemma 4.7 is to imagine that p is "chasing" q. With each motion, p takes a step in pursuit of q and eliminates a portion of the environment Q_k in which q could be "hiding". If K_2 exists, then p eventually "catches" q. An example appears in Figure 4.18.

Now we can prove the algorithm's correctness.

Theorem 4.1 *The sequence $u_{K_1+1}, \ldots, u_{K_2}$ is a localizing sequence for $\{p, q\}$.*

Proof: If K_2 exists, it follows from Lemma 4.6 that $u_{K_1+1}, \ldots, u_{K_2}$ is a localizing sequence for $\{p, q\}$. To show that K_2 exists, note that each p_k is in a different cell of the visibility cell decomposition [77] of E. There are only $O(n^2)$ such cells on the boundary, so $K_2 = O(n^2)$. □

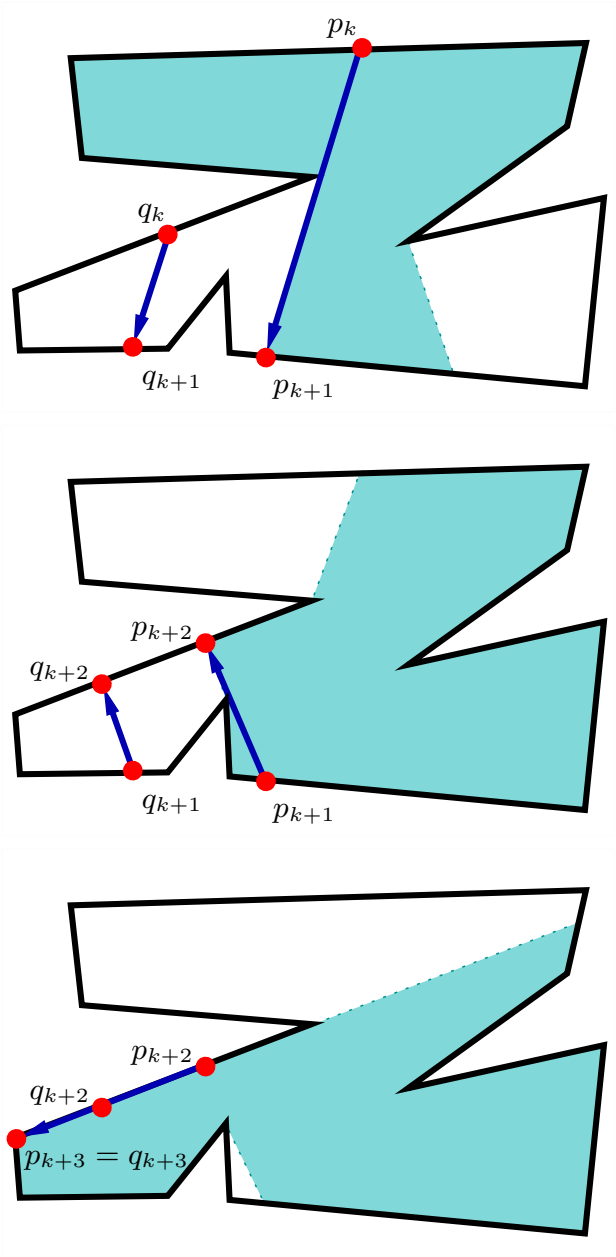


Figure 4.18: A sample execution of the second half of Algorithm 4.2.

Finally, we must consider the special case when $\overline{t_k v_k}$ is a bitangent. This case is problematic because choosing $u_k = (t_k - p_k)/||t_k - p_k||$ is no longer sufficient to ensure that $Q_{k+1} \subset Q_k$. The algorithm as stated would alternate between the actions $t_k - v_k$ and $v_k - t_k$. This problem can be avoided by rotating u_k by a sufficiently small ϵ ensuring that $\overline{q_k q_{k+1}}$

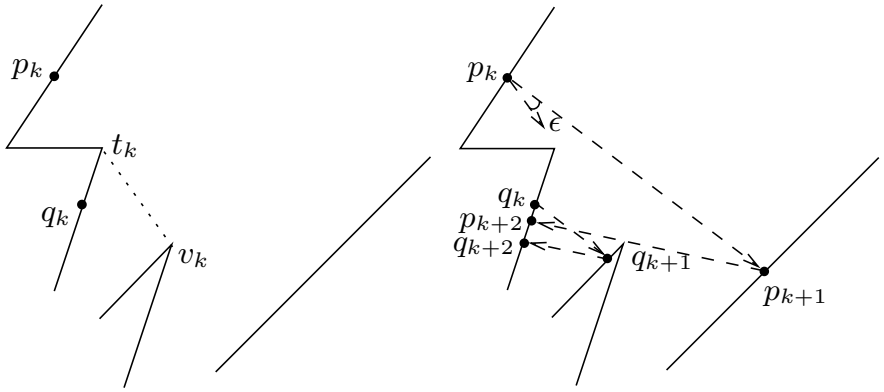


Figure 4.19: The special case when $\overline{t_k v_k}$ is a bitangent. An extra motion is needed.

does not intersect $\overline{t_k v_k}$. Then select $u_{k+1} = (v_k - p_{k+1})/||v_k - p_{k+1}||$. Figure 4.19 illustrates this situation. This modification adds an additional action each time p_k falls at the endpoint of a bitangent complement, but does not substantially change the analysis.

Now we can finally return to the general case with m points. If $m > n$ (recall n is the complexity of ∂E), then by the pigeonhole principle, at least two points must lie on the same edge of ∂E. This pair of points can see each other, and one motion collapses them to a single point. In this way, we can reduce the information state to a set of at most n points using only $m - n$ actions. Then select an arbitrary pair of points p and q from the current information state η_k. We have shown how to merge p and q in $O(n^2)$ steps. Repeating this process at most n times results in a plan of length $O(n^3)$ to map $\{p_1, \ldots, p_m\}$ to a single point. Combining this with the $O(n^3)$ steps from the first part of the algorithm (Section 4.4.3) yields a total plan length of $K = K_1 + K_2 = O(n^3)$.

4.4.5 Computed examples

We have implemented this algorithm in simulation. The top portion of Figure 4.20 shows an environment with many regularities for which Algorithm 4.2 generates a 5-step localizing sequence. In contrast, our algorithm needs 28 steps for the similar but irregular environment in the bottom portion of Figure 4.20. This is in sharp contrast to visibility based localization,

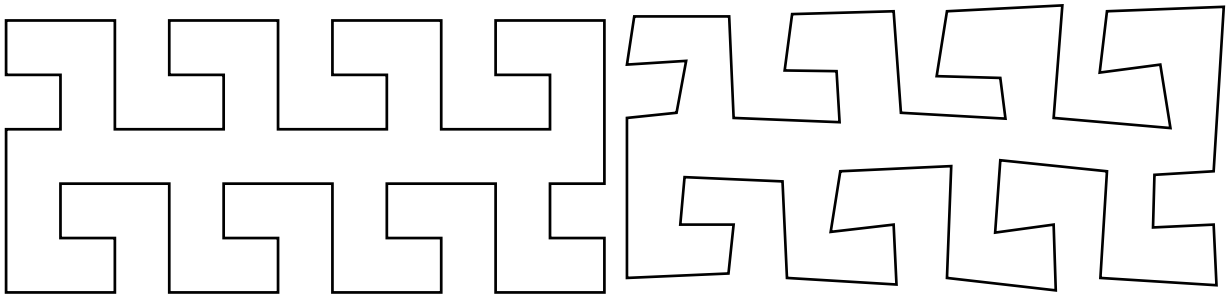


Figure 4.20: [left] An environment with many regularities. Algorithm 4.2 generates a 5-step localizing sequence for this environment, running in approximately 0.4 seconds. [right] A modified version of the environment from Figure 4.20 in which the regularities have been broken. Our algorithm generates a 26 step localizing sequence for this environment, running in approximately 1.0 seconds.

in which such regularities are precisely what make localization problems difficult. Figure 4.21 shows a very irregular environment for which our algorithm generates a 30 step localizing sequence. This sequence is executed from six different initial positions. The robot's final position is in the lower right.

4.5 Localization with an angular odometer and contact sensor

In Section 4.4, we showed that robot model CT, a robot with only a compass and a contact sensor, is capable of localizing itself within its environment. In this section we consider AT, a weaker version of CT in which the compass has been replaced by an angular odometer. This model is identical to that of Section 4.4, except that we now consider actions specified relative to an unknown initial orientation, rather than a global reference direction. Equivalently, we can consider the environment to have been rotated through an unknown angle θ, representing the difference between the global reference direction and the robot's initial orientation. A localizing sequence must map every $x \in X$ to the same x_f, regardless of θ. We show that, under this model, every sequence of actions fails.

Definition 4.3 *An information state-action pair* (η, u) *is a* collapsing transition *if* u *is parallel to some segment in* η.

Lemma 4.8 *Every localizing sequence contains at least one collapsing transition.*

Proof: Suppose there exists some localizing sequence $u_1, \ldots, u_K$ with no collapsing transitions. Arbitrarily pick a segment $s_1 \subseteq \eta_1 = \partial E$. Because of Lemma 4.4, at every step $1 \leq k \leq K$, $F(s_k, u_k)$ contains at least one segment s_{k+1}. We have constructed a segment $s_K \subseteq \eta_K$. Therefore $|\eta_K|$ is infinite, a contradiction. □

Theorem 4.2 *For a robot with only angular odometry and a contact sensor in any polygonal environment* E*, no localizing sequence exists.*

Proof: Suppose such a sequence $u_1, \ldots, u_K$ exists. Let $e_1, \ldots e_n$ denote the set of edges of ∂E, and let $\text{Rot}(v, \phi)$ denote the rotation of $v \subseteq \mathbb{R}^2$ by angle ϕ. If there exists no action-edge pair (u_i, e_j) with u_i and $\text{Rot}(e_j, \theta)$ parallel, then $u_1, \ldots, u_K$ contains no collapsing transitions. The sequence is required to work for all $\theta \in S^1$ but the subset of S^1 in which some u_i coincides with some $\text{Rot}(e_j, \theta)$ has measure 0. Therefore $u_1, \ldots, u_K$ fails for almost every θ. □

The intuition is that reaching a finite cardinality information state requires at least one motion parallel to some environment wall. No finite length localizing sequence can achieve this for all possible initial orientations. See Figure 4.22.

4.6 Discussion

This chapter presented a localization techniques for several robots with severely limited sensing capabilities. In this final section, we discuss these results.

4.6.1 Comparison of results

There are also some subtle but perhaps illustrative differences with the results we have presented for AL and CT. The algorithm for AL is effective only up to symmetry, whereas symmetries are not relevant to CT. This difference can be directly attributed to the fact that, for AL, angular information is only local, rather than global. Likewise, the algorithm for CT can only guarantee a known *final* state. For AL, each motion is precisely measured. This provides sufficient information to determine the initial state and indeed the robot's entire path.

4.6.2 Comparison between sensing models

Perhaps the most closely related localization model is that of [55], in which the robot uses an omnidirectional range sensor. The two phase approach described in that work – that of finding a finite set of candidates (*hypothesis generation*) followed by determination of the true state from among these candidates (*hypothesis elimination*) – is echoed in both Algorithm 4.1 and Algorithm 4.2.

Model AL is strictly weaker than the visibility based model used in [55]. The visibility polygon available to the robot in that work can be viewed as an omnidirectional measure of the distance to the environment boundary. By ignoring all of these distances except the distance to the boundary directly forward, their robot can accurately simulate AL. Moreover, the work of [55] is mainly concerned with *minimum distance* localization, a problem we have not addressed. A central result of [55] is that minimum distance localization with a visibility sensor is NP-complete. In Appendix A, we use very similar techniques to show that minimum distance localization by AL is also NP-complete.

Observe also that AL and CT are not directly comparable. Comparing AL to CT, we exchange the compass for an angular odometer and the contact sensor for a linear odometer. In doing so we have strengthened the linear (distance) sensing while reducing the robot's an-

gular sensing. This type of reasoning, if generalized, leads to the dominance ideas presented in Chapter 3.

4.6.3 Relationship to probabilistic methods

There is a large body of research on Bayesian methods for mobile robot localization (for example, [49, 65, 84, 141, 143]). One way to interpret our results is as a special case of techniques based on POMDPs (for example, [143]) in which sensing is perfect. However, our use of set-based uncertainty allows us to treat the continuous state space exactly, but existing POMDP methods generally require discretization to a finite state space. This sort of Bayesian approach is a very natural way of extending our robot models to account for errors in sensing and motion. Progress has already been made on probabilistic models for the some sensing capabilities considered here. For example, [141] presents probabilistic models for local odometry information. Our algorithms themselves, however, would require substantial adaptation. There is no clear analog to Lemma 4.4, so CT could not "collapse" intervals of probability mass to single points in the same way. Another consideration is that, because we would be forced to settle for accumulating a sufficiently large portion of the probability mass in a sufficiently small region, the basic argument of Theorem 4.2 fails.

Recent work by Erickson (in collaboration with the present author) makes some direct progress on this front [64]. That approach uses a robot model very similar to CT, but with probabilistic rotation errors with magnitude that increases during the robot's execution. A localization algorithm based on entropy-reduction heuristics successfully localized a Roomba autonomous vacuum cleaner robot (Figure 1.1) in several laboratory environments. See Figures 4.23 and 4.24.

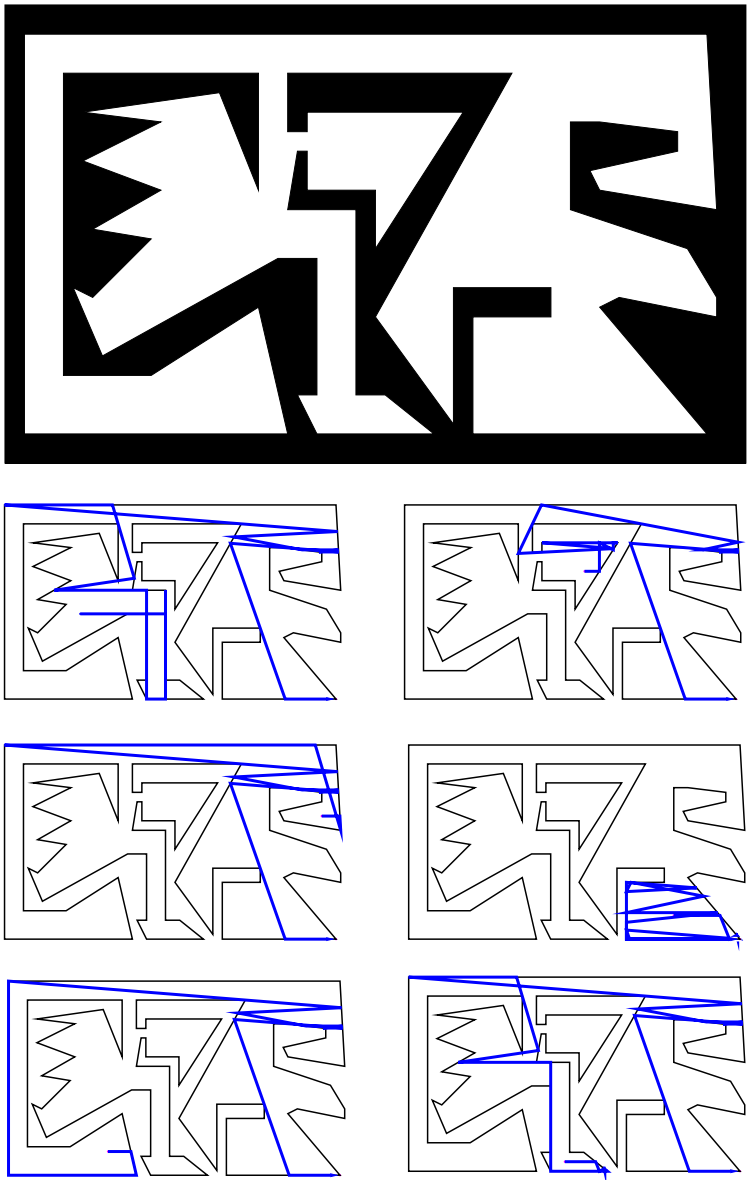


Figure 4.21: [top] An irregular environment for which the localizing sequence computed by our algorithm requires 30 steps. The computation took about 1.9 seconds. [bottom] Execution traces of this localization sequence for 6 different starting positions. For each starting position, the final position is the lower right corner of the environment.

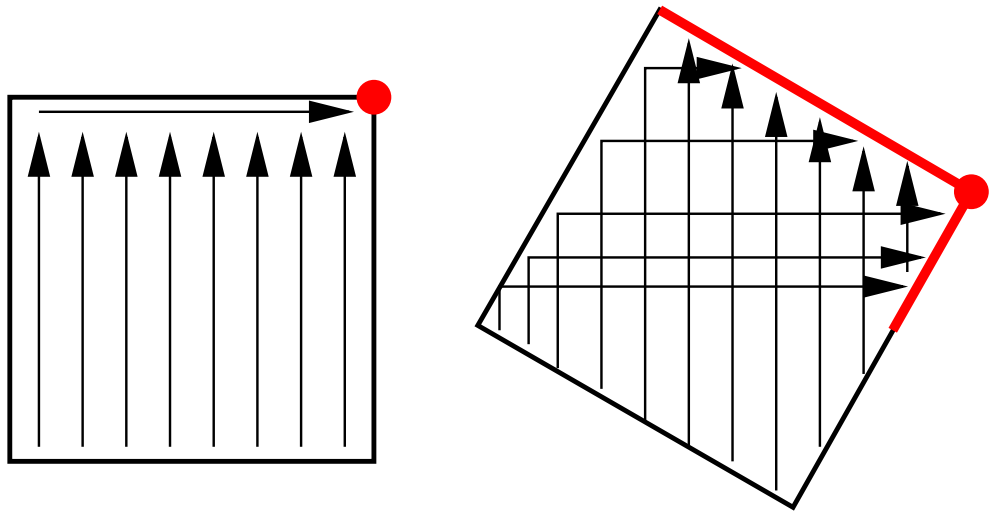


Figure 4.22: A plan must work for any initial orientation, but any plan can only work for finitely many of them, because there must always be at least one collapsing transition.

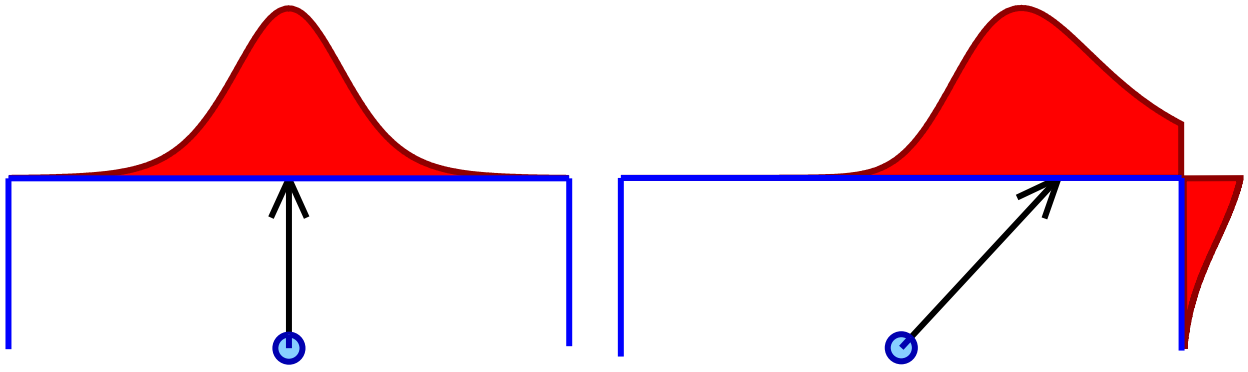


Figure 4.23: Probabilistic error models for a Roomba robot moving from the interior of its environment to the boundary. The variance of the distribution is exaggerated for illustration purposes.

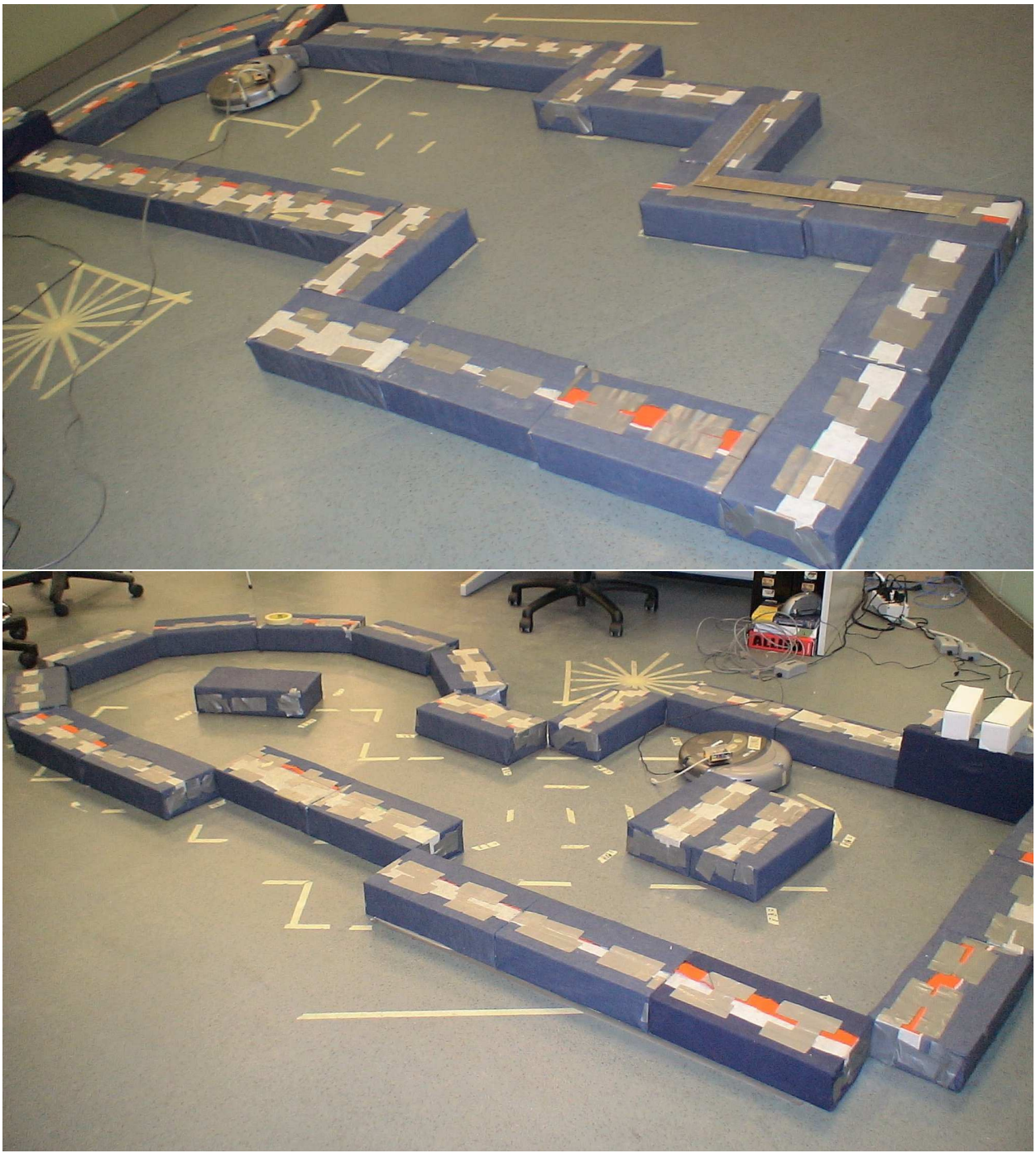


Figure 4.24: Two synthetic environments for which the algorithm of [64] allows a Roomba to solve the active global localization problem. Photos by Lars Erickson.

Chapter 5

Discussion and conclusion

The work presented in this thesis leaves open many possible lines of inquiry. The basic question we asked in Chapter 1 still commands attention: How can we push robots into contexts that are less predictable and more complex, while ensuring that the resulting systems are robust and inexpensive? This chapter contains some brief discussion intended to put the results of this thesis into perspective, and to highlight some directions for future research. The discussion is organized chronologically. We present some general lessoned learned from the current work in Section 5.1, then move in Section 5.2 to open problems that are sufficiently concrete that they can be tackled in the short term. We conclude in Section 5.3 with a longer range view of future directions.

5.1 Lessons learned

A defining aspect of this work is its emphasis on using the information space, explicitly and directly, to approach problems in which the robot is uncertain about its current state. Although we applied this thinking to several problems is great detail, this approach also suggests a few broader, more conceptual lessons.

5.1.1 Solve the passive problem first

A distinction exists between *passive* problems, in which the goal is simply to describe and update the state of the robot's knowledge in response to actions and observations, and *active* problems, in which the robot chooses actions to guide its I-state to a particular goal. We

claim that, for most problems, a good understanding of the passive version of the problem is a prerequisite to finding a good solution to the active version. For example, in Section 4.4, we gave an efficient solution to the passive problem (Section 4.4.1) before considering the related active problem (Sections 4.4.2-4.4.4). This parallels the development of other localization methods, as discussed briefly in Section 4.1. In this particular case, the passive solution provided a key insight – that the reachable derived I-states are finite unions of segments and points – that guided the design of the active algorithm. In other situations, such as the pursuit-evasion work of Guibas et al. [76], the passive solution may uncover enough structure that the active problem, properly posed, yields to direct search techniques.

5.1.2 Use abstraction to model robot systems

Robotics problems are defined by the interaction between sensing, actuation, and computation. In their full complexity, these interactions can be dauntingly complex. As a result, we claim that finding the right level of abstraction is crucial for understanding these problems. Compare, for example, the (relative) simplicity of Definition 3.3, in contrast to the messiness of the continuous time formulations in Section 3.6.2 and Definition 3.5. The difference arises from the abstraction made in modeling robot systems as collections of robotic primitives.

Note, however, that injudicious abstraction can lead to irrelevant or apparently contradictory results. Lemma 3.2c (which states that $R_1 \trianglelefteq R_2 \Rightarrow R_1 \cup R_3 \trianglelefteq R_2 \cup R_3$) becomes incorrect if primitives execute in parallel rather than serially. Moreover, although there is a loose connection between primitives and sensors, the techniques of Chapter 3 can only lead to statements about what can be accomplished with certain selections of *primitives*, and cannot be interpreted directly as statements about what can be accomplished with certain *sensors*.

5.1.3 Use partial orders instead of linear orders

Our results are centered the information preference relation $\preceq$ and the dominance relation $\trianglelefteq$. There may be some temptation linearize these features. Information preference, for example, might be replaced by a scalar measure of entropy [166] or utility [24]. We recommend, in the absence of persuasive motivation to the contrary, to resist such temptations. Although such partial orders admit the possibility that no meaningful comparisons can be made, this is ultimately desirable: physical tasks and robot systems exhibit complex relationships and tradeoffs that can potentially defy meaningful linear ordering.

5.1.4 Don't use unnecessarily specific uncertainty models

Much of the research on planning for robots (and for decision-making agents in general) commits very early to one specific method for representing uncertainty. There is little apparent overlap, for example, between methods that use probabilistic representations (such as [98, 154, 165]), others that use logic-based formulations of various kinds (such as [11, 25, 45, 118, 138]), and still others that reason directly about possible states (as considered throughout this thesis).

It seems likely that general purpose autonomous agents will need to employ some combination of these techniques, depending on the situation. These approaches mentioned above differ only in how the history I-state is interpreted. The I-map κ represents precisely this interpretation: It maps from the history of actions and observations into another space (the derived I-space) in which the implications of these histories is more clear. In this regard, the novelty of our work is that it does not assume any particular I-map or derived I-space. Instead, we take an axiomatic approach, stating conditions on $\mathcal{I}$, κ, and $\preceq$ under which certain results hold. The advantage of this kind of I-space centered approach is that (ideally) one can seek results that are independent of the particular way that uncertainty is modelled. Failing that, our work suggests at least to state precisely the range of uncertainty models to

which a given result applies.

5.2 Open problems

In spite of the progress we have made, the results presented in thesis thesis have important limitations. Of the many issues remaining to be addressed, we mention a few here.

5.2.1 Probabilistic uncertainty

We have focused our attention on nondeterministic uncertainty, but a large subset of contemporary work in robotics uses probabilistic models of uncertainty [14, 65, 84, 141, 143, 154]. Our results also apply, at least in principle, to probabilistic uncertainty. In this context, the relevant derived I-space is a space of probability distributions over X. However, it is not immediately clear what the "right" information preference relation over such a space would be. Depending on the models used, it may also be necessary to relax Definition 3.3 to require only that R_2 can simulate R_1 with sufficiently high probability. More generally, the differences between nondeterministic and probabilistic uncertainty models warrant further exploration. For example, nondeterministic uncertainty has the property that sensing can only help – actions from primitives like P_R (Example 3.5) or P_G (Example 3.6) that do not change the state always lead to a derived I-state at least as good as the current one. Under probabilistic uncertainty, this property does not obviously hold, and sensing can sometimes increase uncertainty.

5.2.2 Selecting the catalog of primitives

Although we believe that our robotic primitives provide a useful abstraction, any results derived using our methods are meaningful only if $\mathcal{RP}$ is diverse enough to faithfully represent the underlying system. It remains an open problem to systematically find small (or at least succinctly described) sets of robotic primitives that are complete (or nearly complete) in the

sense of not eliminating any reachable regions in I-space. There is, however, active interest in related problems for control systems [67, 70, 119, 122].

What happens if $\mathcal{RP}$ is not a finite set? For example, we may extend P_L (from Example 3.4) to a family $\{P_{L,\epsilon} = (S^1, \{0\}, f_{L_\epsilon}, h_{L_\epsilon}) \mid \epsilon \geq 0\}$ of primitives, each using a noisy linear odometer whose error is bounded by ϵ. If $\mathcal{RP}$ contains many such families of primitives, and we assume each robot has at most one primitive from each family, then the space of robot models is a cube in $\mathbb{R}^n$. The problem of identifying the region in which a given task can be solved is correspondingly more difficult.

5.2.3 Efficiency and optimality

Throughout this paper, we have neglected the question of the robot's efficiency in completing its tasks. This weakness is particularly evident, for example, in Example 3.9, in which the statement of dominance does not consider the differences in execution cost, which in this case are likely to be prohibitively large. One established technique for taking such costs into account is to use *competitive ratios* [68, 82], which compare the execution costs of online algorithms (which must gather information during their execution) to offline methods (which have complete information) for the same tasks. It may be fruitful to generalize this notion by considering "relative competitive ratios" that bound the additional cost accrued by replacing one robot system with another dominant robot system.

5.2.4 Parameterization of time

In Section 3.6.2, we parameterized the robot's observations by time. In doing so, we implicitly assumed that the robot has an accurate clock. Although such an assumption is generally not technologically impractical, it requires care in abstract models to ensure that the robot cannot acquire extra information "for free." A robot might, for example, use this implicit clock to parlay an accurate velocity sensor into a perfect odometer. One solution is to express

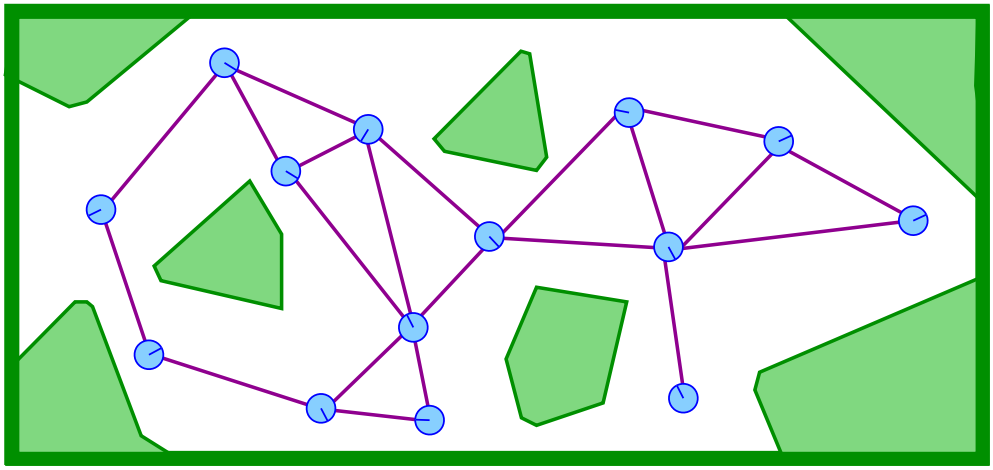


Figure 5.1: Good strategies for coordinating teams of unreliable robots may lead to systems that are reliable as a whole.

$\widetilde{u}$ and $\widetilde{y}$ as functions of some other abstract parameter p. To recover the original functions of time, the robot must determine a hidden mapping from $\mathbb{R}$ to $\mathbb{R}$ under which p maps to t. Such issues are addressed in detail in [103].

5.3 Future directions

We conclude by mentioning a few very broad directions for future research.

5.3.1 Communication, cooperation, and disposable robots

The minimalist philosophy advocated here is motivated partly by the promise of constructing robot systems very inexpensively. By equipping robots with only a few, carefully selected sensors, the cost of individual robots can be quite low. Equipped with good strategies for communication and cooperation, teams of such robots have the potential for robustness even if the individual units are very unreliable. By introducing a multiplicity of (possibly heterogeneous) robots, one may possibly reduce the sensing required for certain tasks. Figure 5.1 is a cartoon of such a situation. There are also a number of related problems for networks

of immobile sensing devices. Suppose each device has access to a choice of sensing modalities and communication techniques, but the use of these capabilities is limited by power constraints. What is the best way to activate sensors and communication links in order to achieve adequate coverage across the network?

5.3.2 Unknown and unstructured environments

Another direction is to consider mobile robots in dynamic, unstructured environments. Can a very simple robot reliably move through an outdoor area crowded with moving people and other irregularly shaped obstacles? One way of approaching the problem is to consider strategies guaranteed to make progress in some sense, even if the amount of progress cannot be predicted or measured. By chaining such strategies together in appropriate ways, we can guarantee that the robot will reach its goal. To properly examine these issues would require both new basic results and experimental validation.

5.3.3 Necessary computation power

Beyond the questions addressed in this thesis, related to the robot's sensing and motion capabilities, important issues remain with regard to the computation requirements of tasks. By equipping a robot with a more powerful sensor, we may potentially reduce the complexity of the computations the robot must perform. The field currently lacks the tools and vocabulary to analyze these tradeoffs systematically. We suggest that the information space, which gives formalisms for understanding sensing, motion, and the collection of information, provides the right language for understanding problems of this type. A starting point for this direction might be to examine the amount and types of information that the robot must retain (in contrast to information it must be able to collect but may be needed only for a short time) to complete a task.

Appendix A

Hardness of minimum distance localization with odometry

This appendix presents a hardness result for minimum-distance localization. Recall the robot model AL from Chapter 4, which is equipped with accurate odometers for measuring its translations and rotations. We show that, given a polygonal environment, the problem of computing a localization strategy for AL that minimizes the worst-case distance traveled is NP-hard. We do so using a technique very similar to that used in [55].

More precisely, we consider the following problem:

> Limited-Distance Odometry-based Localization(LDOL)
>
> Instance: A polygon E, a set $F \subseteq E \times S^1$ of k possible starting states, and a distance d.
>
> Question: Is there strategy that localizes robot AL, starting at some $x_1 \in F$, with worst-case distance traveled d or less?

The proof proceeds by reduction from the abstract decision tree problem (ADT), shown to be NP-complete by Hyafil and Rivest [81].

> AbstractDecisionTree (ADT)
>
> Instance: A set of k objects $X = \{x_1, \ldots, x_k\}$, a set of n membership tests $\mathcal{T} = \{T_1, \ldots, T_n\}$ with each $T_i \subseteq X$, and an integer h.
>
> Question: Is there a rooted binary decision tree of height h or less, in which each internal node is labeled with a test in $\mathcal{T}$, each edge is labeled with "Yes" or "No," and each leaf is labeled with an object in X, that correctly and uniquely identifies each object in X?

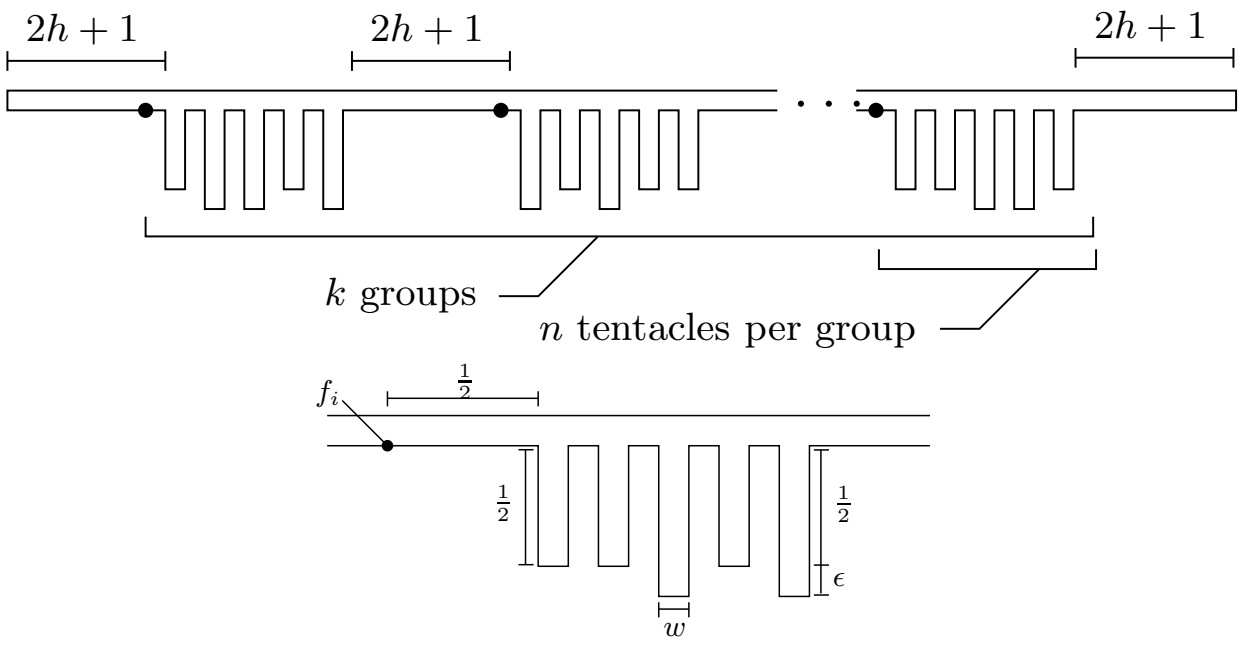


Figure A.1: Constructing a localization problem from an instance of ADT. Not to scale.

Consider an instance $(X, \mathcal{T}, h)$ of ADT to construct an instance of LDOL. For E, build a long, key-shaped polygon as illustrated in Figure A.1. The polygon has k groups of n "tentacles" each. The groups are separated by corridors of length $2h+1$, with an extra corridor at each end. Within each group, the tentacles have width $w = 1/(4nh)$, with separation of w between each tentacle. The tentacle lengths are determined by the test set $\mathcal{T}$. In group i, tentacle j has length $1/2$ if $x_i \in \mathcal{T}_j$ and length $1/2+\epsilon$ for some $\epsilon > 0$ if $x_i \notin T_j$. For F, choose a set of k states with identical orientations positioned distance $1/2$ from the left edge of each group. Choose $d = h+1/2$. This construction is certainly polynomial time.

Before presenting the reduction itself, we make a few observations. First, note that the groups of tentacles are spaced far enough apart that any strategy with worst case distance at most d must stay within the group in which it starts. Second, observe that each group is identical except at the tips of its tentacles. Therefore, it suffices to consider strategies that make a series of "probes" of particular tentacles to determine whether their length is $1/2$ or $1/2+\epsilon$. To do so requires one unit of total motion, except that the robot need not exit from

the last tentacle it probes. This exception is offset by the fact that the starting position is distance 1/2 from the nearest tentacle.

The distance traveled to make a single probe is at least 1 (to travel distance 1/2 down the tentacle and back) and at most $2wn + 1$ (to account for lateral motions between tentacles). Therefore, if the problem *can* be solved with worst case distance d, we can do so with a strategy that makes at most $\lfloor d/(2wn + 1) \rfloor$ probes in the worst case. Likewise, if the problem *cannot* be solved with worst-case distance d, it cannot be solved by any strategy that makes $\lfloor d/1 \rfloor$ probes or fewer.

Given a strategy that makes t probes in the worst case, we can construct a decision tree of height h for the corresponding ADT problem. Perform test T_j whenever the robot probes the jth tentacle in its current group. The leaves of the decision tree correspond to the starting position of the fully localized robot: If the robot determines that its starting position was in the ith group, the corresponding decision tree leaf is labeled x_i. This straightforward transformation can be done in polynomial time.

Now we can state the result.

Lemma A.1 *LDOL is NP-hard.*

Proof: By reduction from ADT, using the construction described above. It remains only to show that the constructed LDOL instance has the same solution (that is, "Yes" or "No") as the original ADT instance. If solution to the LDOL decision problem is "Yes," meaning that the localization problem can be solved with worst-case distance d, the argument above implies that there exists a decision tree for the original ADT problem with height

$$\left\lfloor \frac{d}{2wn+1} \right\rfloor \leq \frac{d}{2wn+1} = \frac{h+1/2}{\frac{2n}{4nh}+1} = h. \tag{A.1}$$

Similarly, if the LDOL solution is "No," meaning that no strategy can localize the robot

with worst-case distance d, then no decision tree exists with height

$$\lfloor d \rfloor = \lfloor h + 1/2 \rfloor = h. \tag{A.2}$$

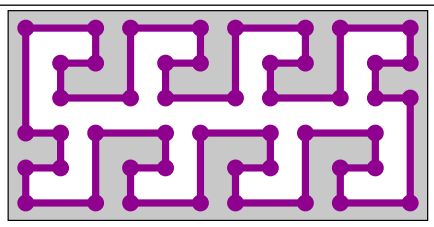
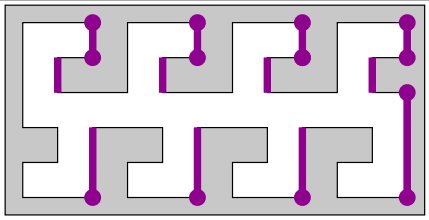
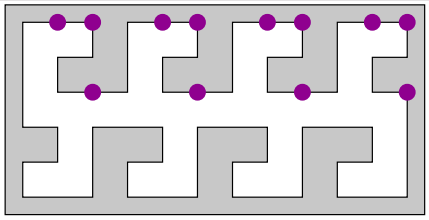
Conclude that ADT can be reduced in polynomial time to LDOL. Since ADT is NP-hard, LDOL is also NP-hard. □

Appendix B

Execution examples

This appendix presents detailed execution examples for Alg. 4.2. The intent is to illustrate, step by step, how the information states progress from the initial information state of total uncertainty to a final information state with which only a single state is consistent.

Table B.1: A localizing sequence computed by Algorithm 4.2 for a highly symmetric environment.

i	u_i	η_{i+1}
0		
1		
2		

Continued on next page.

Table B.1, cont.

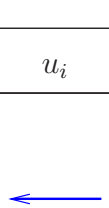
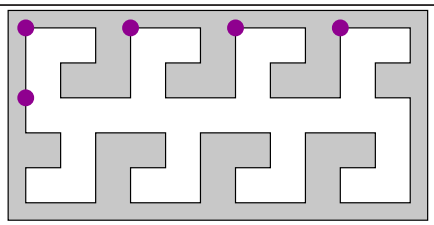
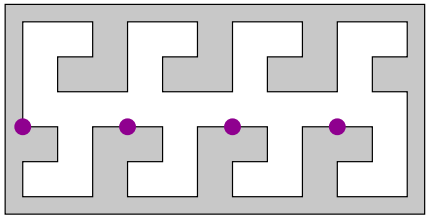
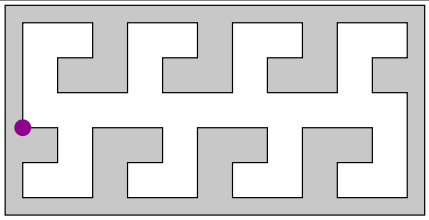
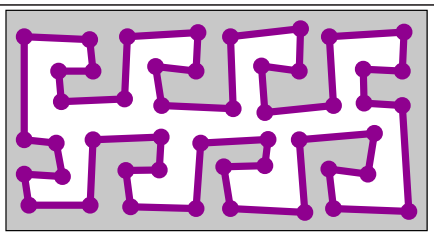
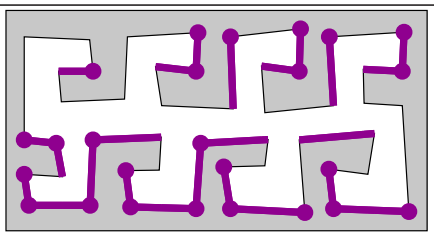
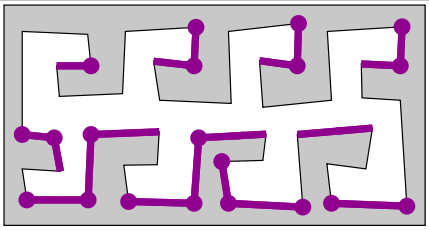
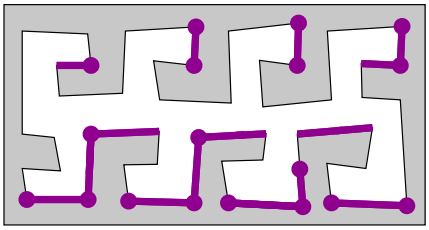
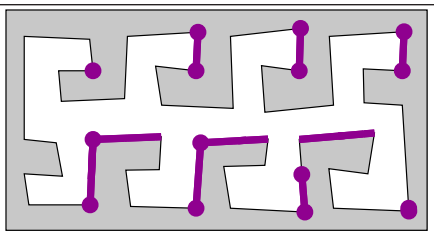
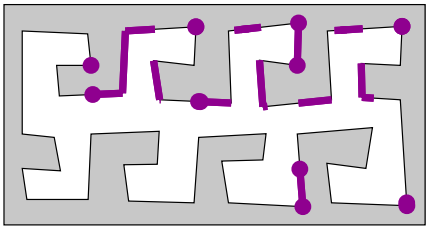
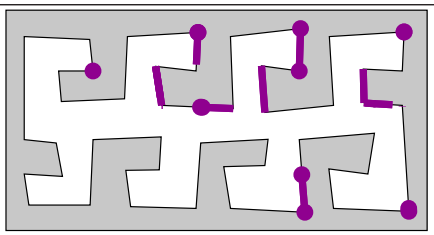
i	u_i	η_{i+1}
3		
4		
5		

Table B.2: A modified version of the environment from Table B.1 in which the symmetries have been broken. Our algorithm generates a 28 step localizing sequence for this environment.

i	u_i	η_{i+1}
0		

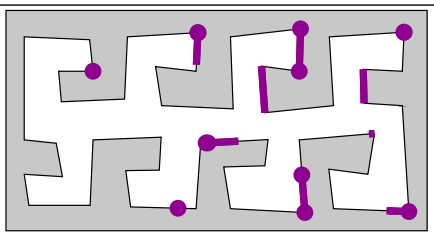
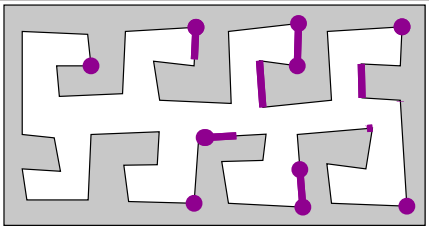
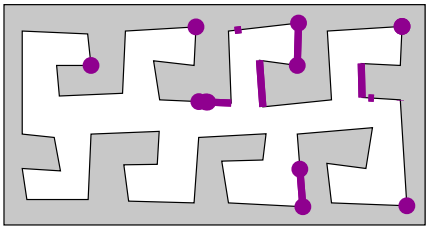
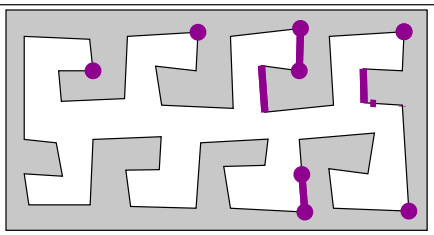
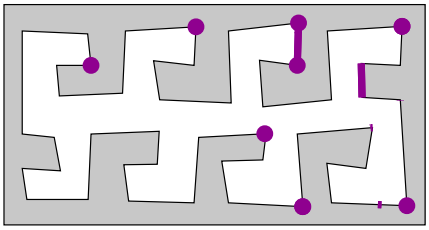
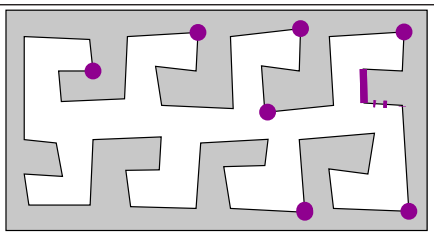
Continued on next page.

Table B.2, cont.

i	u_i	η_{i+1}
1		
2		
3		
4		
5		
6		

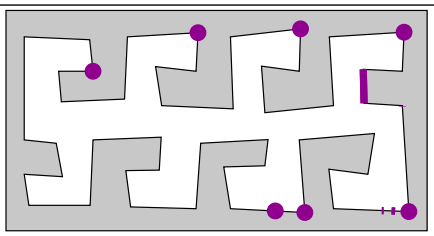
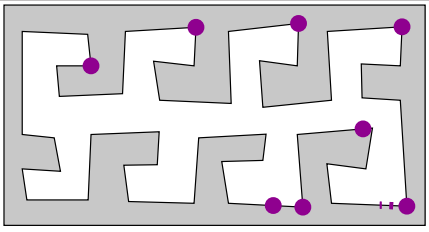
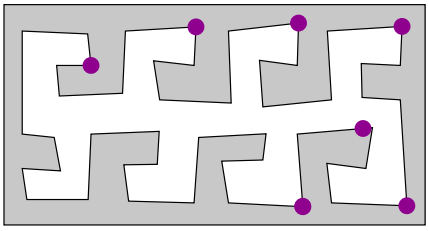
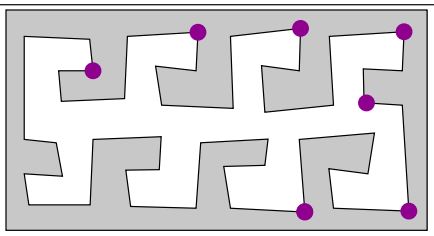
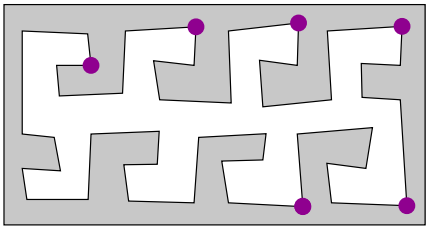
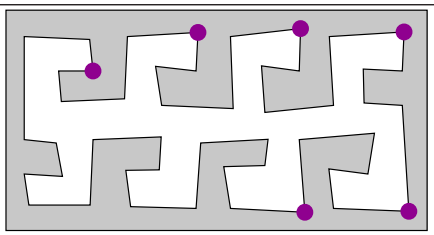
Continued on next page.

Table B.2, cont.

i	u_i	η_{i+1}
7		
8		
9		
10		
11		
12		

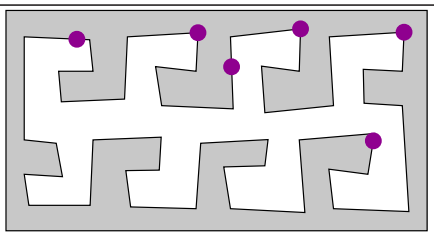
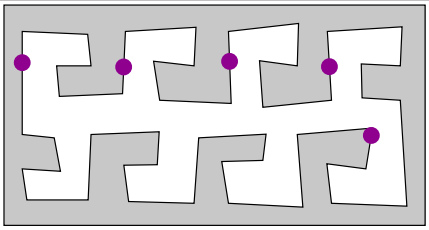
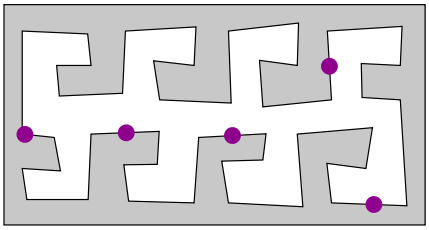
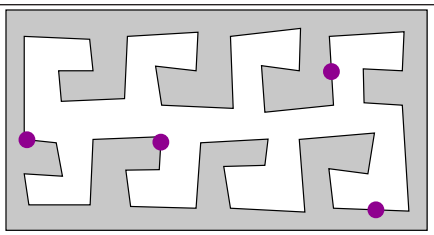
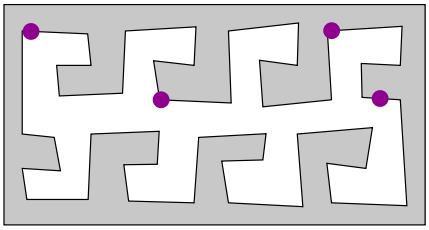
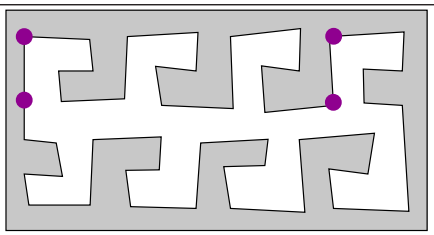
Continued on next page.

Table B.2, cont.

i	u_i	η_{i+1}
13		
14		
15		
16		
17		
18		

Continued on next page.

Table B.2, cont.

i	u_i	η_{i+1}
19		
20		
21		
22		
23		
24		

Continued on next page.

Table B.2, cont.

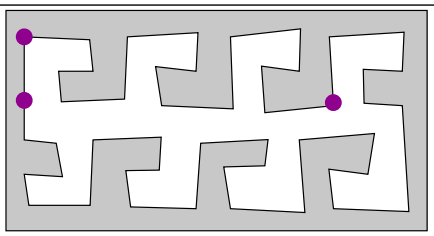
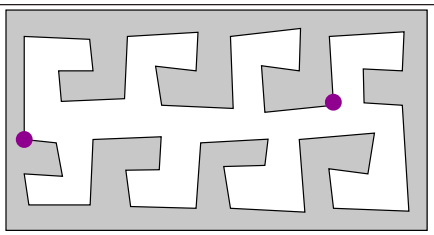
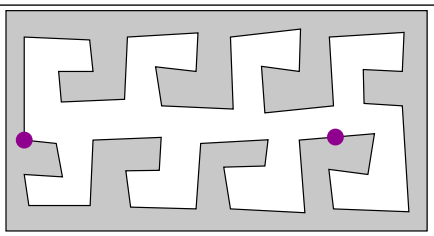
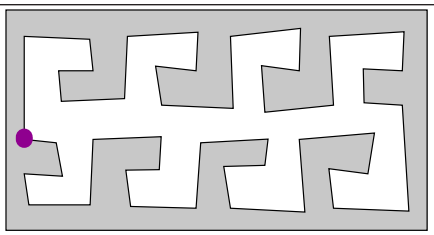
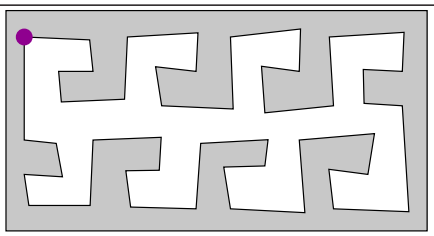
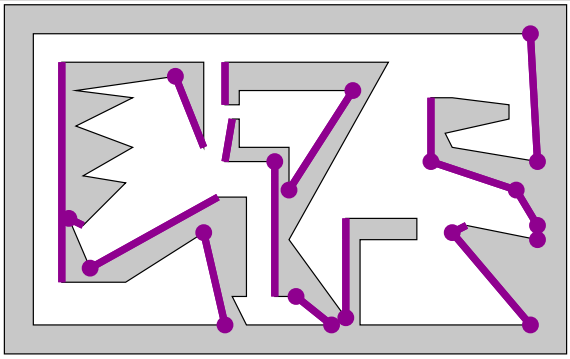
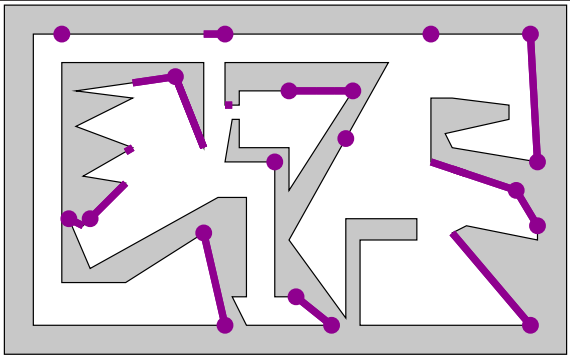
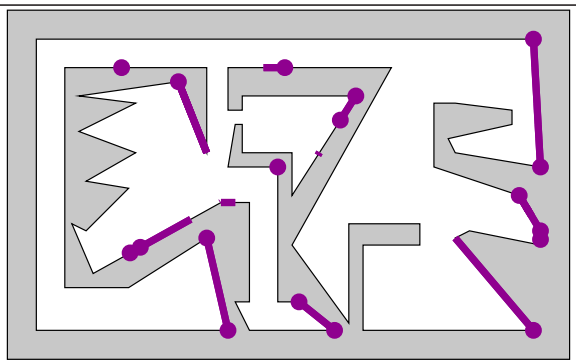
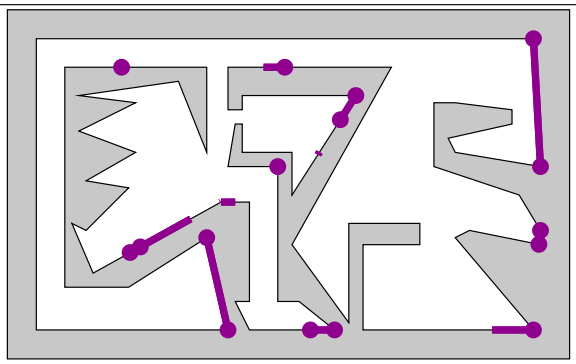
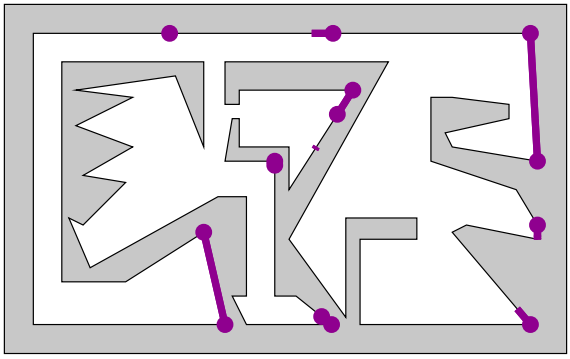
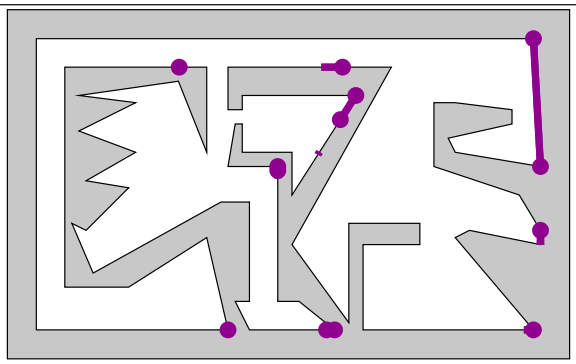
i	u_i	η_{i+1}
25		
26		
27		
28		
29		

Table B.3: An irregular environment for which the localizing sequence computed by our algorithm requires 30 steps.

i	u_i	η_{i+1}
0		
1		
2		

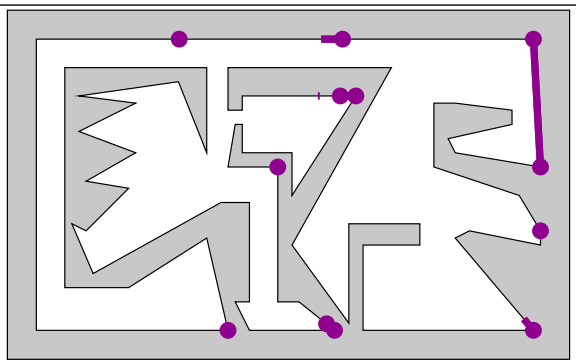
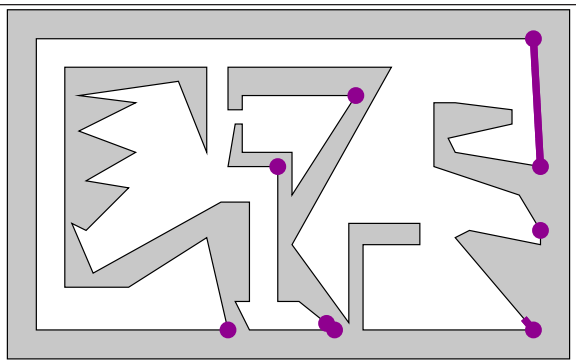
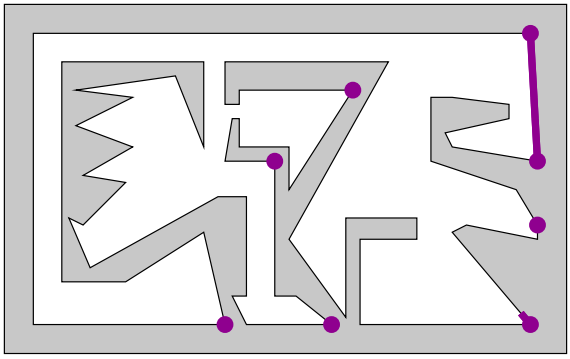
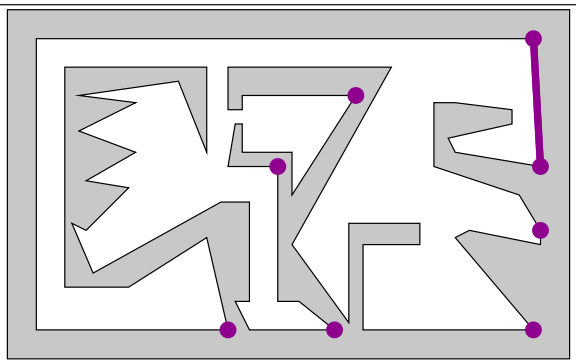
Continued on next page.

Table B.3, cont.

i	u_i	η_{i+1}
3		
4		
5		
6		

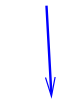
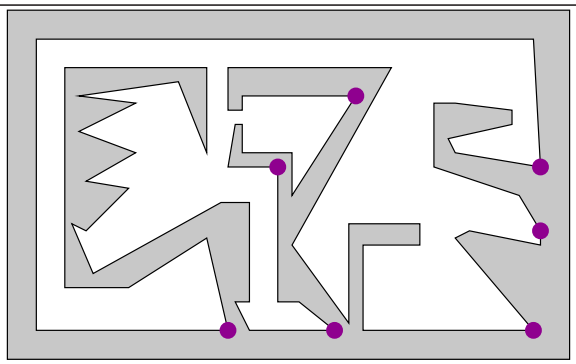
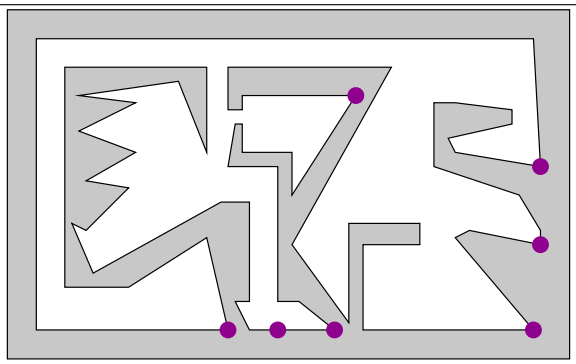
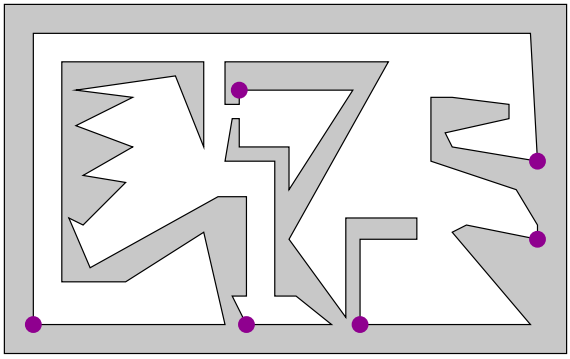
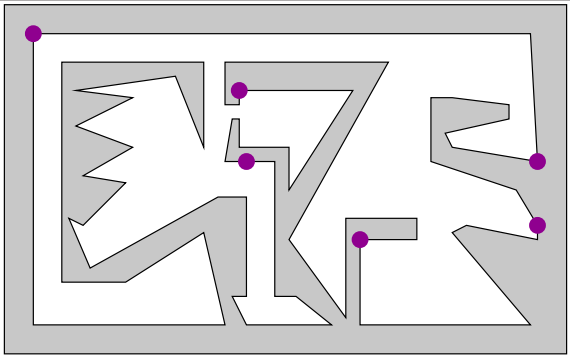
Continued on next page.

Table B.3, cont.

i	u_i	η_{i+1}
7		
8		
9		
10		

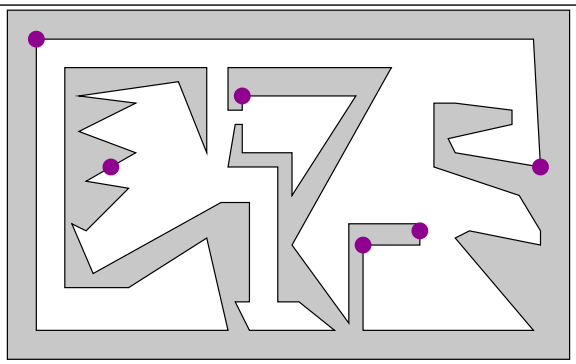
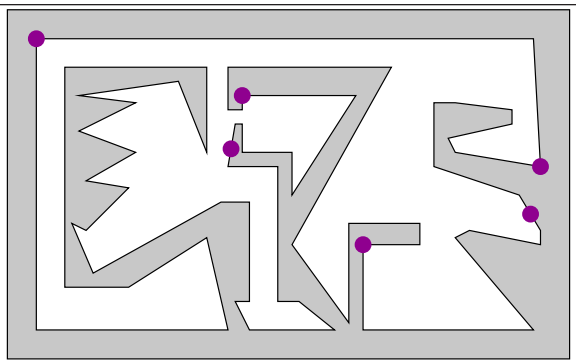
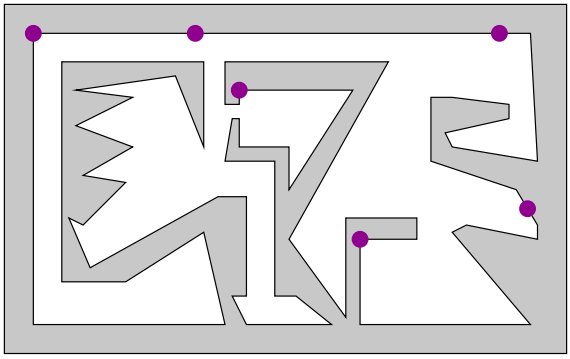
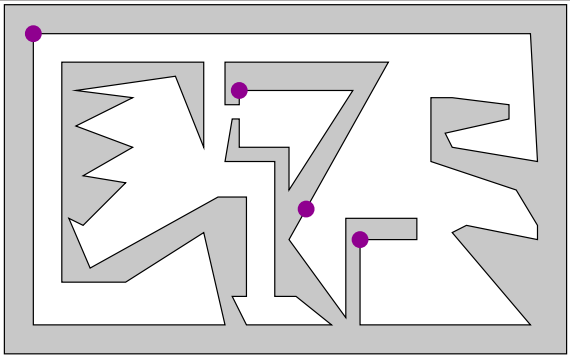
Continued on next page.

Table B.3, cont.

i	u_i	η_{i+1}
11		
12		
13		
14		

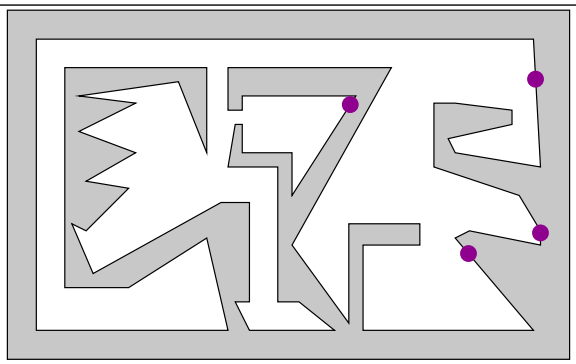
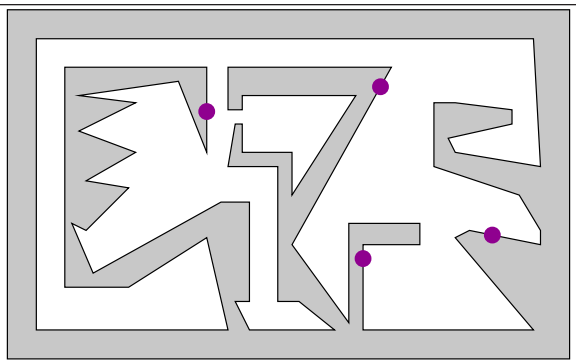
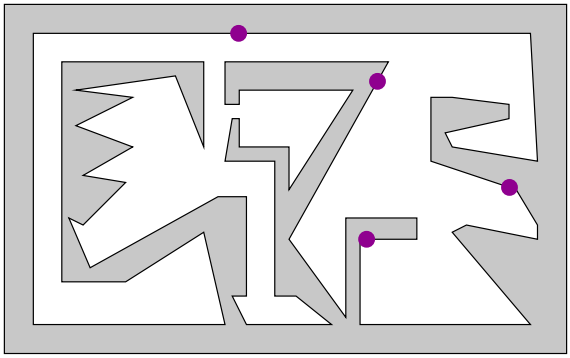
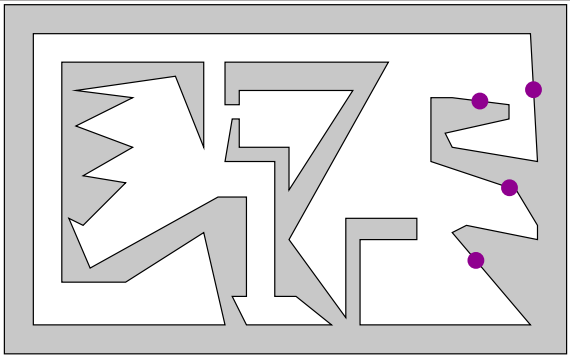
Continued on next page.

Table B.3, cont.

i	u_i	η_{i+1}
15		
16		
17		
18		

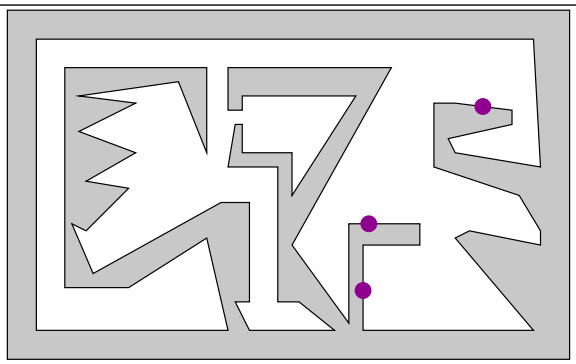
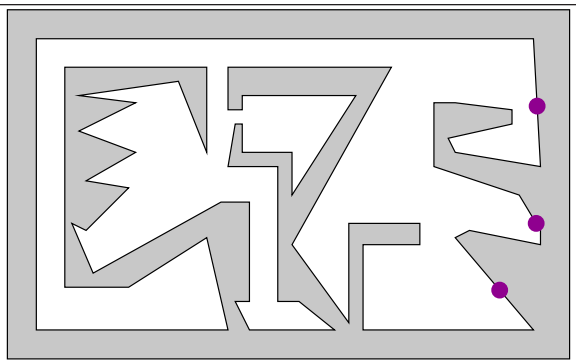
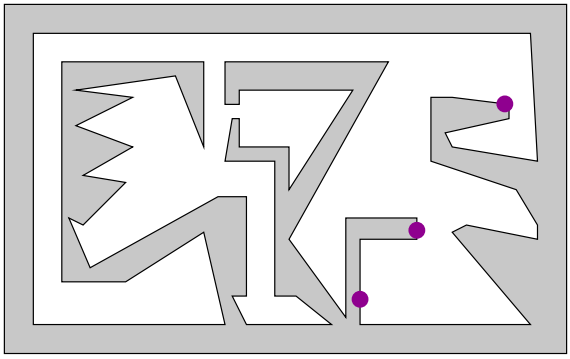
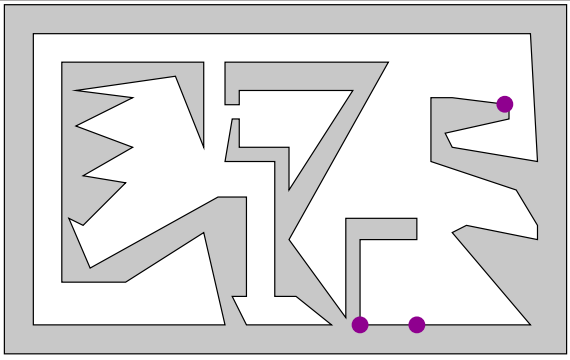
Continued on next page.

Table B.3, cont.

i	u_i	η_{i+1}
19		
20		
21		
22		

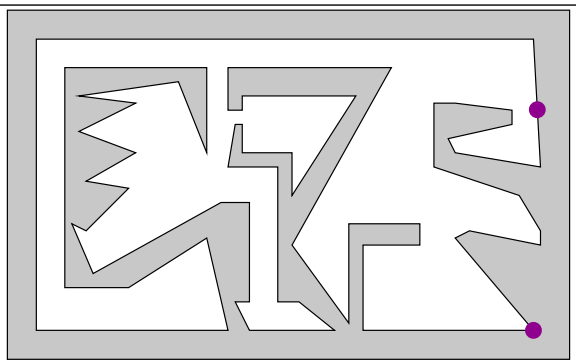
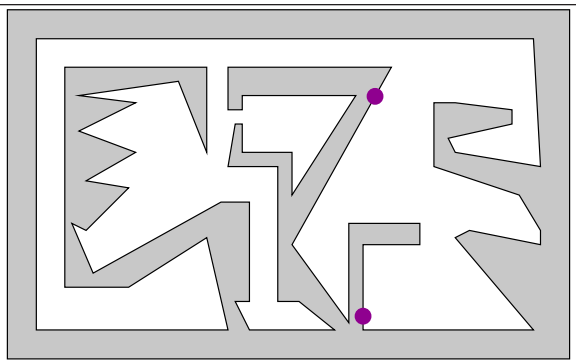
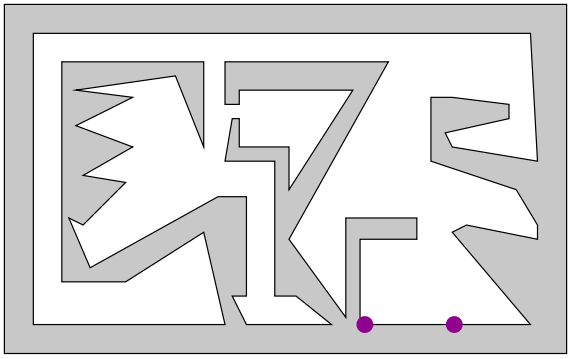
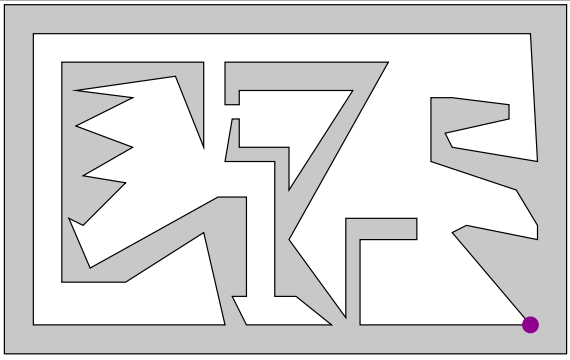
Continued on next page.

Table B.3, cont.

i	u_i	η_{i+1}
23		
24		
25		
26		

Continued on next page.

Table B.3, cont.

i	u_i	η_{i+1}
27		
28		
29		
30		

References

[1] A. D. Ames A. Abate and S. Sastry. Error-bounds based stochastic approximations and simulations of hybrid dynamical systems. In *American Control Conference*, 2006. 33

[2] E. U. Acar and H. Choset. Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of critical points and voronoi diagrams. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001. 7, 66

[3] E. U. Acar and H. Choset. Robust sensor-based coverage of unstructured environments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001. 7

[4] P. K. Agarwal, A. D. Collins, and J. L. Harer. Minimal trap design. In *Proc. IEEE International Conference on Robotics and Automation*, volume 3, pages 2243–2248, 2001. 6, 66

[5] S. Akella, W. Huang, K. M. Lynch, and M. T. Mason. Parts feeding on a conveyor with a one joint robot. *Algorthmica*, 26(3):313–344, March 2000. 6, 66

[6] S. Akella and M. Mason. Posing polygonal objects in the plane by pushing. *International Journal of Robotics Research*, 17(1):70–88, January 1998. 6, 66

[7] S. Akella and M. Mason. Using partial sensor information to orient parts. *IJRR*, 18:963–997, 1999. 6

[8] R. Aleliunas. A simple graph traversing problem. Master's thesis, University of Toronto, April 1978. 7

[9] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 218–223, 1979. 7

[10] N. Alon, Y. Azar, and Y. Ravid. Universal sequences for complete graphs. *Discrete Appl. Math.*, 27(1-2):25–28, 1990. 7

[11] E. Amir and S. Russell. Logical filtering. In *Proc. International Joint Conferences on Artificial Intelligence*, 2003. 110

[12] B. Aronov, L. Guibas, M. Teichmann, and L. Zhang. Visibility queries in simple polygons and applications. *Algorithms and Computation, 9th International Symposium, ISAAC '98*, 1533, 1998. 96

[13] P. N. Atkar, D. C. Conner, A. Greenfield, H. Choset, and A. A. Rizzi. Uniform coverage of simple surfaces embedded in r3 for auto-body painting. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2004. 2

[14] D. J. Austin and P. Jensfelt. Using multiple gaussian hypotheses to represent probability distributions for mobile robot localization. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1036–1041, 2000. 33, 111

[15] D. Avis and H. Imai. Locating a robot with angle measurements. *Journal of Symbolic Computation*, 10(3-4):311–326, 1990. 69, 72

[16] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory, 2nd Ed.* Academic, London, 1995. 5

[17] R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins. Searching in the plane. *Information and Computation*, 106:234–252, 1993. 59, 60

[18] K. Basye and T. Dean. Map learning with indistinguishable locations. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 331–342. North-Holland, 1990. 72

[19] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: exploring and mapping directed graphs. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 269–278, 1998. 67

[20] J. O. Berger. *Statistical Decision Theory.* Springer-Verlag, Berlin, 1980. 25

[21] R-P. Berretty, K. Goldberg, M. Overmars, and F. Van der Stappen. Trap design for vibratory part feeders. *International Journal of Robotics Research*, 20(11), November 2001. 6

[22] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientic, Belmont, MA, second edition, 2001. 5, 24

[23] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, April 1997. 72

[24] F. Bian, D. Kempe, and R. Govindan. Utility based sensor selection. In *Proc. International Conference on Information Processing in Sensor Networks*, 2006. 110

[25] P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic.* Elsevier, Amsterdam, 2006. 110

[26] D. Blackwell and M. A. Girshik. *Theory of Games and Statisitical Decisions.* Dover, New York, 1979. 50

[27] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. *SIAM Journal on Computing*, 26(1):110–137, 1997. 7

[28] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer Verlag, Berlin, 1998. 68

[29] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 132–142, 1978. 6, 67, 68

[30] D. L. Boley, E. S. Steinmetz, and K. T. Sutherland. Robot localization from landmarks using recursive total least squares. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1381–1386, 1996. 69

[31] G. Boothroyd, C. Poli, and L. E. Murch. *Automatic Assembly*. Marcel Dekker, Inc., 1982. 6

[32] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, UK, 98. 31

[33] A. Borodin, W. L. Ruzzo, and M. Tompa. Lower bounds on the length of universal traversal sequences. In *Proc. ACM Symposium on Theory of Computing*, pages 562–573. ACM Press, 1989. 7

[34] R. I. Brafman, J. Y. Halpern, and Y. Shoham. On the knowledge requirements of tasks. *Artificial Intelligence*, 98(1-2):317–349, 1998. 33

[35] A. Broggi, C. Caraffi, P. P. Porta, and P. Zani. The single frame stereo vision system for reliable obstacle detection used during the 2006 darpa grand challenge on TerraMax. In *IEEE Intelligent Transportation System Conference*, 2006. 1

[36] M. Brokowski, M. A. Peshkin, and K. Goldberg. Curved fences for part alignment on a belt. *ASME Journal of Mechanical Design*, 117(1), March 1995. 6

[37] J. F. Canny and K. Y. Goldberg. "RISC" industrial robots: Recent results and current trends. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1951–1958, 1994. 6

[38] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *Proc. ACM Symposium on Theory of Computing*, pages 574–586. ACM Press, 1989. 7

[39] B. Chazelle and L. G. Guibas. Visibility and intersection problems in plane geometry. *Discrete and Computation Geometry*, 4:551–589, 1989. 82

[40] C.-T. Chen. *Linear System Theory and Design*. Holt, Rinehart, and Winston, New York, 1984. 55

[41] T.-H. Chiang, M. S. Apaydin, D. L. Brutlag, D. Hsu, and J.C. Latombe. Predicting experimental quantities in protein folding kinetics using stochastic roadmap simulation. In *Proc. ACM International Conference on Computational Biology*, volume 3909 of *LNCS*, pages 410–424. Springer, 2006. 2

[42] H. Choset and J. Burdick. Sensor based planning, part I: The generalized Voronoi graph. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1649–1655, 1995. 66

[43] H. Choset and J. Burdick. Sensor based motion planning: Incremental construction of the hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):126–148, 2000. 7

[44] H. Choset and J. Burdick. Sensor based motion planning: The hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):96–125, 2000. 7

[45] A. Cimatti, M. Roveri, and P. Bertoli. Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence*, 159(1-2):127–206, 2004. 110

[46] S. Cook. The complexity of theorem proving procedures. In *Proc. ACM Symposium on Theory of Computing*, pages 151–158, 1971. 10

[47] I. J. Cox. Blanche – An experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991. 69, 71

[48] T. L. Dean and M. P. Wellman. *Planning and Control*. Morgan Kaufman, San Francisco, CA, 1991. 13

[49] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. IEEE International Conference on Robotics and Automation*, 1999. 24, 72, 104

[50] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Robot localization without depth perception. In *Scandinavian Workshop on Algorithm Theory*, 2002. 69, 72

[51] X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *Journal of the ACM*, 45(2):215–245, 1998. 7, 31

[52] B. R. Donald. On information invariants in robotics. *Artificial Intelligence*, 72:217–304, 1995. 5, 6, 33

[53] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *Proc. IEEE International Conference on Robotics and Automation*, pages 190–197, Sacramento, CA, 1991. 5

[54] G. Dudek, K. Romanik, and S. Whitesides. Localizing a robot with minimum travel. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1995. 69, 73

[55] G. Dudek, K. Romanik, and S. Whitesides. Localizing a robot with minimum travel. *SIAM Journal on Computing*, 27(2):583–604, 1998. 103, 115

[56] G. Dudek and C. Zhang. Vision-based robot localization without explicit object models. In *Proc. IEEE International Conference on Robotics and Automation*, pages 76–82, 1996. 69

[57] M. Egerstedt. Motion description languages for multi-modal control in robotics. In A. Bicchi, H. Cristensen, and D. Prattichizzo, editors, *Control Problems in Robotics*, Springer Tracts in Advanced Robotics, pages 75–90. Springer-Verlag, 2002. 33

[58] D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990. 6

[59] M. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics and Automation*, 4(4):369–379, August 1988. 6, 66

[60] M. Erdmann, M. T. Mason, and Jr. G. Vaněček. Mechanical parts orienting: The case of a polyedron on a table. *Algorthmica*, 10:206–247, 1993. 6

[61] M. A. Erdmann. Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1):19–45, 1986. 6

[62] M. A. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorthmica*, 10:248–291, 1993. 5

[63] M. A. Erdmann. Understanding action and sensing by designing action-based sensors. *International Journal of Robotics Research*, 14(5):483–509, 1995. 6, 20, 33

[64] L. Erickson, J. M. O'Kane, and S. M. LaValle. Probabilistic localization using only a clock and a contact sensor. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007. Under review. xiii, 104, 107

[65] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207, 1998. 73, 104, 111

[66] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 470–498, New York, 2001. Springer. 72

[67] E. Frazzoli. *Robust Hybrid Control of Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2001. 112

[68] Y. Gabriely and E. Rimon. Competitive complexity of mobile robot on line motion planning problems. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2004. 31, 112

[69] B. Gates. A robot in every home. *Scientific American Magazine*, January 2007. 2

[70] A. Girard and G. J. Pappas. Hierarchical control using approximate simulation relations. In *Proc. IEEE Conference on Decision and Control*, 2006. 112

[71] A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, May 2007. 33

[72] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorthmica*, 10:201–225, 1993. 6, 66, 87

[73] K. Y. Goldberg and M. T. Mason. Bayesian grasping. In *Proc. IEEE International Conference on Robotics and Automation*, 1990. 5, 6

[74] D. D. Grossman and M. W. Blasgen. Orienting parts by computer controlled manipulation. *IEEE Transactions on Systems, Man, and Cybernetics*, 5(5):561–565, 1975. 6

[75] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and Systems Sciences*, 39(2):126–152, 1989. 84

[76] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *International Journal on Computational Geometry and Applications*, 9(5):471–494, 1999. 109

[77] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Proc. Workshop on the Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995. 69, 71, 96, 98

[78] J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38:231–235, 1991. 84

[79] R. Hinkel and T. Knieriemen. Environment perception with a laser radar in a fast moving robot. In *Proceedings of Symposium on Robot Control*, 1988. 24, 72

[80] J. E. Hopcroft, J. D. Ullman, and R. Motwani. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, second edition, 2000. 30

[81] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:15–17, 1976. 73, 115

[82] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. On the competitive complexity of navigation tasks. In *Sensor Based Intelligent Robots*, pages 245–258, 2002. 112

[83] P. Jensfelt and H. I. Christensen. Pose tracking using laser scanning and minimalistic environmental models. *IEEE Transactions on Robotics and Automation*, 17(2):138–147, April 2001. 73

[84] P. Jensfelt and S. Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, 17(5):748–760, October 2001. 104, 111

[85] J. D. Kahn, N. Linial, N. Nisan, and M. E. Saks. On the cover time of random walks in graphs. *Journal of Theoretical Probability*, 2(1):121–128, January 1989. 7

[86] T. Kameda, M. Yamashita, and I. Suzuki. On-line polygon search by a seven-state boundary 1-searcher. *IEEE Transactions on Robotics*, 22(3):446–460, June 2006. 67

[87] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13(6):814–822, December 1997. 6

[88] I. Kamon, E. Rivlin, and E. Rimon. Range-sensor based navigation in three dimensions. In *Proc. IEEE International Conference on Robotics and Automation*, 1999. 6, 66

[89] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 441–447, 1993. 59

[90] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *Proceedings World Computer Congress*, 1992. 31

[91] H. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In *Proc. International Joint Conferences on Artificial Intelligence*, 1999. 10

[92] J. M. Kleinberg. The localization problem for mobile robots. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 521–531, 1994. 72, 73

[93] S. Koenig, A. Mudgal, and C. Tovey. An approximation algorithm for the robot localization problem. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2006. 73

[94] H. W. Kuhn. Extensive games and the problem of information. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, pages 196–216. Princeton University Press, Princeton, NJ, 1953. 5

[95] P. R. Kumar and P. Varaiya. *Stochastic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1986. 5

[96] K. N. Kutulakos, C. R. Dyer, and V. J. Lumelsky. Provable strategies for vision-guided exploration in three dimensions. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1365–1371, 1994. 6

[97] K. N. Kutulakos, V. J. Lumelsky, and C. R. Dyer. Vision-guided exploration: a step toward general motion planning in three dimensions. In *Proc. IEEE International Conference on Robotics and Automation*, pages 289–296, 1993. 6

[98] M. Littman L. Kaelbling and and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998. 13, 110

[99] A. M. Ladd, K. E. Bekris, A. P. Rudys, D. S. Wallach, and L. E. Kavraki. On the feasibility of using wireless Ethernet for indoor localization. *IEEE Transactions on Robotics and Automation*, 20(3):555–559, June 2004. 24, 72

[100] M. Lau and J. Kuffner. Behavior planning for character animation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2005. 2

[101] J.-P. Laumond. Kineo CAM: a success story of motion planning algorithms. *IEEE Robotics and Automation Magazine*, 13(2):90–93, June 2006. 2

[102] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/. 5, 21, 37, 56

[103] S. M. LaValle and M. B. Egerstedt. On time: Clocks, chronometers, and open-loop control. Technical Report UIUCDCS-R-2007-2861, University of Illinois at Urbana-Champaign, 2007. 60, 113

[104] A. Lazanas and J. C. Latombe. Landmark-based robot navigation. In *Proc. National Conference on Artificial Intelligence (AAAI)*, 1992. 7

[105] S. Lee. Mobile robot localization using optical mice. In *IEEE Conference on Robotics, Automation and Mechatronics*, volume 2, pages 1192–1197, 2004. 69

[106] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proc. IEEE International Conference on Robotics and Automation*, 2000. 33

[107] J. Leonard, H. Durrant-Whyte, and I. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):89–96, 1992. 24, 72

[108] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, June 1991. 72

[109] L. Levin. Universal search problems. In B. A. Trakhtenbrot, editor, *A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms*, volume 6 of *Annals of the History of Computing*, pages 384–400. IEEE, 1984. Translated from Russian. 10

[110] D. Lieb, A. Lookingbill, and S. Thrun. Adaptive road following using self-supervised learning and reverse optical flow. In *Proc. Robotics: Science and Systems*, 2005. 1

[111] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984. 5, 6, 33

[112] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979. 14

[113] R. D. Luce and H. Raiffa. *Games and Decisions: Introduction and Critical Survey*. Dover, Mineola, New York, 1957. 13

[114] V. J. Lumelsky and A. A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorthmica*, 2:403–430, 1987. 6, 66

[115] V. J. Lumelsky and S. Tiwari. An algorithm for maze searching with azimuth input. In *Proc. IEEE International Conference on Robotics and Automation*, pages 111–116, 1994. 6, 66

[116] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *Proc. ACM Symposium on Theory of Computing*, pages 322–333, 1988. 49, 72

[117] Matthew Mason. Kicking the sensing habit. *AI Magazine*, 14(1):58–59, 1993. 3

[118] D. McDermott and J. Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13:41–72, 1980. 110

[119] T. Mehta, F. Delmotte, and M. Egerstedt. Motion alphabet augmentation based on past experiences. In *Proc. IEEE Conference on Decision and Control*, 2006. 112

[120] J. S. B. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles in the plane. *Algorthmica*, 8:431–459, 1992. 84

[121] M. Moll and M. Erdmann. Manipulation of pose distributions. *International Journal of Robotics Research*, 21(3):277–292, 2002. 6

[122] T. Murphey. Motion planning for kinematically overconstrained vehicles using feedback primitives. In *Proc. IEEE International Conference on Robotics and Automation*, 2006. 112

[123] B.K. Natarajan. An algorithmic approach to the automated design of part orienters. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 132–142, 1986. 6

[124] A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proc. Conference on Uncertainty in Artificial Intelligence*, 2000. 5

[125] C. Ó. Dúnlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982. 7, 66

[126] J. M. O'Kane. Global localization using odometry. In *Proc. IEEE International Conference on Robotics and Automation*, 2006. 70

[127] J. M. O'Kane and S. M. LaValle. Almost-sensorless localization. In *Proc. IEEE International Conference on Robotics and Automation*, 2005. 71

[128] J. M. O'Kane and S. M. LaValle. On comparing the power of mobile robots. In *Proc. Robotics: Science and Systems*, 2006. 33

[129] J. M. O'Kane and S. M. LaValle. Localization with limited sensing. *IEEE Transactions on Robotics*, 2007. To appear. 70

[130] J. M. O'Kane and S. M. LaValle. On comparing the power of robots. *International Journal of Robotics Research*, 2007. To appear. 33

[131] J. M. O'Kane and S. M. LaValle. Sloppy motors, flaky sensors, and virtual dirt: Comparing imperfect ill-informed robots. In *Proc. IEEE International Conference on Robotics and Automation*, 2007. 33

[132] C. H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985. 50

[133] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991. 31, 66

[134] M. Rao, G. Dudek, and S. Whitesides. Randomized algorithms for minimum distance localization. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, pages 265–280, 2004. 73

[135] N. Rao, S. Kareti, W. Shi, and S. Iyenagar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, 1993. 7

[136] W. Renken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1993. 24, 72

[137] K. Romanik and S. Schuierer. Optimal robot localization in trees. In *Proc. ACM Symposium on Computational Geometry*, pages 264–273, 1996. 73

[138] S. Russell and J. Wolfe. Efficient belief-state AND-OR search, with application to kriegspiel. In *Proc. International Joint Conferences on Artificial Intelligence*, 2005. 5, 110

[139] M. Seiz, P. Jensfelt, and H. I. Christensen. Active exploration for feature based global localization. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000. 73

[140] C. E. Shannon. Presentation of a maze solving machine. In H. von Foerster, M. Mead, and H. L. Teuber, editors, *Cybernetics: Circular, Casual, and Feedback Mechanisms in Biological and Social Systems, Transactions Eighth Conference*, pages 169–181, New York, 1952. Josiah Macy Jr. Foundation. 67

[141] H. Shatkay and L. P. Kaelbling. Learning topological maps with weak local odometric information. In *Proc. International Joint Conferences on Artificial Intelligence*, pages 920–927, 1997. 104, 111

[142] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1060–1065, 1998. 69

[143] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. International Joint Conferences on Artificial Intelligence*, pages 1080–1087, 1995. 104, 111

[144] M. Sipser. *Introduction to the Theory of Computation*. PWS, Boston, MA, 1997. 30, 34

[145] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985. 31, 49, 72

[146] R. I. Soare. *Recursively enumerable sets and degrees*. Springer-Verlag, Berlin, 1987. 34

[147] K. Sugihara. Some location problems for robot navigation using a simple camera. *Computer Vision, Graphics, and Image Processing*, 42(1):112–129, 1988. 69, 71, 72

[148] S. Suri, E. Vicari, and P. Widmayer. Simple robots with minimal sensing: From local visibility to global geometry. In *Proc. National Conference on Artificial Intelligence (AAAI)*, 2007. 67

[149] K. T. Sutherland and W. B. Thompson. Inexact navigation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1–7, 1993. 72

[150] R. H. Taylor, M. T. Mason, and K. Y. Goldberg. Sensor-based manipulation planning as a game with nature. In *International Symposium of Robotics Research*, Cambridge MA, 1988. MIT Press. 50

[151] M. Thompa. Lower bounds on universal traversal sequences for cycles and other low degree graphs. *SIAM Journal on Computing*, 21(6), December 1992. 7

[152] S. Thrun. Probabilisitic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000. 24, 72

[153] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, pages 1–25, April 1998. 24, 72

[154] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005. 33, 110, 111

[155] M. Tomono and S. Yuta. Mobile robot localization based on an inaccurate map. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 399–404, 2001. 72

[156] B. Tovar, L. Guilamo, and S. M. LaValle. Gap Navigation Trees: Minimal representation for visibility-based tasks. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2004. 7, 66

[157] B. Tovar, S. M. LaValle, and R. Murrieta. Locally-optimal navigation in multiply-connected environments without geometric maps. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003. 7

[158] A. F. van der Stappen, R.-P. Berretty, K. Goldberg, and M. H. Overmars. Geometry and part feeding. In *Sensor Based Intelligent Robots*, pages 259–281, 2000. 6

[159] I. A. Wagner, M. Lindenbaum, and A.M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, October 1999. 67

[160] G. Weiss, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1994. 24, 72

[161] D. E. Whitney. Real robots don't need jigs. In *Proc. IEEE International Conference on Robotics and Automation*, 1986. 3, 6

[162] J. Wiegley, K. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. *Assembly Automation*, 17(2), August 1997. 6

[163] J. D. Wolter, T. C. Woo, and R. A. Volz. Optimal algorithms for symmetry detection in two and three dimensions. *The Visual Computer*, 1:37–48, July 1985. 85

[164] A. Yershova, B. Tovar, R. Ghrist, and S. M. LaValle. Bitbots: Simple robots solving complex tasks. In *Proc. National Conference on Artificial Intelligence (AAAI)*, 2005. 7

[165] N. Zhang and W. Lin. A model approximation scheme for planning in partially observable stochastic domains. *Journal of Artificial Intelligence Research*, 7:199–230, 1997. 110

[166] Y. Zhang and Q. Ji. Sensor selection for active information fusion. In *Proc. National Conference on Artificial Intelligence (AAAI)*, 2005. 110

[167] R. Zhou and E. A. Hansen. An improved grid-based approximation algorithm for POMDPs. In *Proc. International Joint Conferences on Artificial Intelligence*, 2001. 5

Author's Biography

Jason O'Kane was born near Pittsburgh, Pennsylvania on November 7, 1979, a day in which the temperature was a seasonable 40 degrees and the skies were partly cloudy. He graduated from McKeesport Area High School in 1997. In 2001, O'Kane earned a B.S. degree *summa cum laude* in Computer Science from Taylor University in Upland, Indiana. From the University of Illinois at Urbana-Champaign, he earned an M.S. degree in 2005 and a Ph.D. degree in 2007, both in Computer Science. O'Kane was named Taylor University's Outstanding Computer Science Graduate in 2001 and he was awarded a Roy S. Carver Fellowship by the University of Illinois for 2001-2002. He is a member of the Institute for Electrical and Electronics Engineers, the IEEE Robotics and Automation Society, the Association for the Advancement of Artificial Intelligence, and the Society for Industrial and Applied Mathematics. In August 2007, O'Kane will join the faculty of the Department of Computer Science and Engineering at the University of South Carolina.