Sensor Interpretation and Task-Directed Planning
Using Perceptual Equivalence Classes

Bruce Donald*
Jim Jennings**

TR 91-1248
December 1991

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*This paper describes research done in the Computer Science Robotics Laboratory at Cornell University. Support for our robotics research is provided in part by the National Science Foundation under grants No. IRI-8802390, IRI-9000532 and by a Presidential Young Investigator award, and in part by the Mathematical Sciences Institute and AT&T Bell laboratories.
**Supported in part by an NSF graduate fellowship and under the grants in (*).

Sensor Interpretation and Task-Directed Planning Using Perceptual Equivalence Classes

Bruce Donald* **Jim Jennings†**

Cornell Computer Science Department

Ithaca, NY

Abstract

We consider how a robot may interpret its sensors and direct its actions so as to gain more information about the world, and to accomplish manipulation tasks. The key difficulty is uncertainty, in the form of noise in sensors, error in control, and unmodelled or unknown aspects of the environment. Our research focuses on general techniques for coping with uncertainty, specifically, to sense the state of the task, adapt to changes, and reason to select actions to gain information and achieve the goal.

Sensors yield partial information about the world. When we interrogate the environment through our sensors, we in effect view a projection of the world onto the space of possible sensor values. We investigate the structure of this sensor space and its relationship to the world. We observe that sensors partition the world into perceptual equivalence classes, that can serve as natural "landmarks." By analyzing the properties of these equivalence classes we develop a "lattice" and a "bundle" structure for the information available to the robot through sensing and action. This yields a framework in which we develop and characterize algorithms for sensor-based planning and reasoning.

*This paper describes research done in the Computer Science Robotics Laboratory at Cornell University. Support for our robotics research is provided in part by the National Science Foundation under grants No. IRI-8802390, IRI-9000532 and by a Presidential Young Investigator award, and in part by the Mathematical Sciences Institute and AT&T Bell laboratories.

†Supported in part by an NSF graduate fellowship and under the grants in (*).

Contents

1 Introduction

In recent years, robotics researchers have focused considerable attention on the key problem of uncertainty (at many levels): for example, error or noise in modelling, actuation, and sensing [LMT, Mason, DW, Brooks, Mat, Brost, Buc, Caine, Can, Erd, E89, Don, LLS]. The domain of applicability for strategies for reducing uncertainty, gaining information, and directing actions to accomplish manipulation tasks is very broad. It includes (for example)

- Manipulation and mechanical assembly
- Mobile robot navigation and planning
- Design and layout.

In assembly, we may have sensing and control uncertainty, and sparse or incomplete environmental models. Work on synthesizing assembly strategies is particularly applicable during fine-motion and tight assemblies. Mobile robots may have noisy sensors and operate in dynamic, uncertain environments. In design for assembly, we wish to develop a physical device that will be functional in a variety of physical situations; this "variety" may be viewed as a parametric form of uncertainty.

A central theme to previous work has been to determine what information is required to solve a task, and to direct a robot's actions to acquire that information to solve it. In this paper, we will expand on this theme, with particular emphasis on strategies that tolerate uncertainty and recover from failure. In addition, one of our goals is to develop a framework that permits us to measure the sensitivity of plans to particular assumptions about the world, and to in fact minimize those assumptions where possible.

Our main idea is as follows. For reasons of algorithmic tractability, much of our previous work has concentrated on planning (and execution) that emphasized *reachability* (the places a robot can reach during an action) while using very simple models of *recognizability* (the places a robot can distinguish using sensors). Erdmann first introduced this mathematical distinction in [Erd]. Indeed, some foundational work has indeed assumed that recognizability (e.g., recognizing attainment of a goal during a continuous control regime) was given by some "oracle" and that only reachability need be considered [Don90,E89,Bri,FHS]. This effectively reduces planning to a backwards simulation (under control uncertainty), which is an easier (although important) problem.

However, due to the sensitivity of simulation to modelling errors, this focus on reachability tightly coupled the planning process to relatively accurate models of the environment.[1] To consider sparse or incomplete models of the environment, or mobile robots operating in very uncertain and dynamic domains, we take a different tack. In very uncertain domains, reachability computations (i.e., forward- and back-simulations) are essentially valid only locally, since they can only rely on local models of the world. We are trying to develop a framework for sensing and action in which recognizability is the "senior" partner. To say this precisely: in previous work, we attempted in effect to "reduce" recognizability computations to reachability computations. Hence, the fundamental geometric building blocks for this theory were the reachable sets (from some initial conditions, under particular controls). Since these sets had considerable algebraic and geometric structure, geometric algorithms could be obtained to generate control and sensing strategies—plans, if you will.

In this paper we propose a theory of planning, sensing and action in which the fundamental building blocks are, in effect, the "recognizable sets"—that is, the places in the world that the

[1]There is an extension to the work of [LMT,E] that includes error in the geometric models of the environent as well as in controls and sensing, (see [Don, D, Don90]). These algorithms takes as input a parametric representation of the geometric uncertainty. But this theory still requires an "exact model" of the environment by which we mean an exact parameterization of error. This implies a known (and finite) dimensionality for the shape space. Finally, the resulting preimage planning algorithms are at least doubly-exponential in the dimension of the shape space [CR, Can].

robot can recognize and distinguish between. To this end we observe that viewing the world through sensors partitions the world (locally) into "perceptual equivalence classes."[2] The more information the sensors provide, the "finer" this partition is. Various possible partitions of the world fit into a lattice structure (where one partition is "higher" than another if it is finer). This lattice structure captures the information or knowledge state about the world; the lattice is related to the "version space" of Mitchell [Mit]. Recognizability is a key theme in the preimage framework of [LMT]; see [LMT,Erd,Don,Can]. Canny [Can] described a planner for LMT strategies based on the recognizable sets. The recognizable sets are related to the "preimages" of [LMT], the "effective sensor interpretation sets" of Erdmann [Erd], the "recognizable sets"[3] of Canny [Can], and the "signature neighborhoods" of Mahadevan and Connell [MC]. They are called "sensor equivalence classes" by Christiansen, Mason, and Mitchell [CMM], "perceptual aliases" by Whitehead and Ballard [WB], and "model aliases" by Shafer [S].

Hence, a particular partition of the world into perceptual equivalence classes is an entry in the lattice. Sensing actions that "gain information" allow us to *ascend* to a "higher" (hence finer) partition in the lattice. Motion of the robot (in general) may induce a move between perceptual equivalence classes in the same partition. If the motion "gains" information, then we ascend in the lattice as well. Of course, it is possible for actions (and even sensing operations) to "lose" information (*descend* to a more ambiguous partition) as well. For example, if the robot is touching a wall and breaks contact, then the uncertainty in the robot's position may be increased.

Using this theory we develop notions of task-directed planning and show how history can be used to direct actions and gain information. We investigate the structure of the recognizability lattice and its relationship to the world. We consider how to compute perceptual equivalence classes both from a geometric world model,[4] and without a model, incrementally, from the world. Finally, we'll investigate mathematical properties that can help the robot generate disambiguating strategies to gain information about the world to accomplish a task. We believe our framework is useful in understanding, programming, and planning for other mobots, such as Mataric's [Mat] robot Toto as well. We are implementing our theories on a mobile autonomous platform equipped with sensors and several Cornell Generic Controllers in a ring-network [OdD, BSD, BD, RD]. We make connections with experimental work where appropriate. The first part of this paper develops the framework informally. A more detailed development may be found in the appendix.

2 Research Issues

2.1 Outline

The primary goal of our research is task-level planning. We approach this goal by utilizing a blend of theory, implementation, and experimentation. We wish to investigate task-level planning for autonomous agents, such as mobile robots, that function in an uncertain environment. These robots typically have very approximate, inaccurate, or minimal models of the environment. For example, although the geometry of its environment is crucial to determining its performance, a mobile robot might only have a partial, or local "map" of the world. Similarly, the expected effects of a robot's actuators critically influence its selection of actions to accomplish a goal, but a robot may have only a very approximate, or local predictive ability with regard to forward-simulation of a control strategy. While mobile robots are typically equipped with sensors in order to gain information about the world, and to compensate for errors in actuation and prediction, these sensors are noisy, and in turn provide inaccurate information. We investigate an approach whereby the robot attempts

[2] This term was suggested to us by Dan Huttenlocher.

[3] We are grateful to J. Canny for pointing out the similarity.

[4] By *world model* we mean an *a priori* geometric representation of the robot's environment; for example, a CAD solid model provided by a human. Our robots could never build such a model, since it would contain imperceptible and hence unconstructible features. In particular, our *B-encoded RR-graph* (below) is not a world model, but a knowledge-representation of sensor and control history.

to acquire the necessary information about the world by planning a series of experiments using the robot's sensors and actuators, and building data-structures based on the robot's observations of these experiments. A key feature of this approach is that the experiments the robot performs should be driven by the information demands of the task. That is, in performing some task, the robot may enter a state in which making progress towards a goal requires more information about the world (or its own state). In this case, the robot should plan experiments which can disambiguate the situation. When this process is driven by the information demands of the task, we believe it is an important algorithmic technique to effect task-directed sensing. This paper considers:

1. A theory of *sensor interpretation and task-directed planning using perceptual equivalence classes*, intended to be applicable in highly uncertain or unmodelled environments, such as for a mobile robot.

2. Algorithmic techniques for modeling geometric constraints on recognizability, and the building of internal representations (such as maps) using these constraints.

3. Explicit encoding of the information requirements of a task using a *lattice* (information hierarchy) of *recognizable sets*, which allows the robot to perform experiments to recognize a situation or a landmark.

4. The synthesis of robust mobot programs using the geometric constraints, *constructive recognizability* experiments, and uncertainty models imposed by the task.

One of our most important goals is to show how our theory can be made constructive and algorithmic. We propose a framework for mobot programming based on *constructive recognizability*, and discuss why it should be robust in uncertain environments. Our objective is to demonstrate the following: When recognizability is thusly constructive, we naturally obtain task-directed sensing strategies, driven by the information demands encoded in the structure of the recognizable sets.

A principled theory of sensing and action is crucial in developing task-level programming for autonomous mobile robots. We propose a framework for such a theory, providing both a precise vocabulary and also appropriate computational machinery for working with issues of information flow in and through a robot system equipped with various types of sensors and operating in a dynamic, unstructured environment.

Structured environments, such as those found around industrial robots, contribute towards simplifying the robot's task because a great amount of information is encoded, often *implicitly*, into the robot's control program. There remains considerable structure, however, in human-occupied environs, including offices and laboratories. Unfortunately, much of the corresponding useful information is lost when the world is projected through imprecise sensors. Furthermore, acquiring information is often mediated by imperfect controls. The key issue, then, is coping with uncertainty.

Briefly stated, our objectives include:

- development of a theory of task-level programming of autonomous agents in unstructured environments,

- construction of theoretical tools for answering such information-related questions as, e.g.:

  1. What information is needed by a particular robot to accomplish a particular task?
  2. How may the robot acquire such information?
  3. What properties of the world have a great effect on the fragility of a robot plan/program?
  4. What are the capabilities of a given robot (in a given environment or class of environments)?

- Development of algorithms to implement this theory, and experiments using physical robots to develop, refine, and test the theory.

A key issue, given these goals, is to characterize the kinds of representations and data structures a mobile robot can build in an uncertain and dynamic world—based on its observations of this world though noisy sensors. One key question is how the robot should use *a priori* information about, or "models" of the world, and concomitantly, what are the kinds of errors and mistakes it might make by basing inferences on "matches" between noisy sensor data and (possibly faulty) "expectations" about the world (i.e., models). One way to explore this issue is to investigate how the robot can function with "minimal" models of the world. We often take this route, not so much because we believe this will result in better robots, but rather, because determining what initial state a robot needs to perform a task, and analyzing how the robot should use this *a priori* information are both exceedingly difficult. Indeed, this difficulty drives our investigation. Finally, we may not even have a choice of whether or not to give the robots models of the world: they may be unavailable or highly inaccurate (for example, a mobile robot in an unknown environment), or extremely expensive to use (for example, see [CR, Can]).

2.2 A Calculus of the Observable

Hence, we wish to construct a theory of mobile robotics from a "calculus of the observable." The advantages of such a theory would be that, first, any representations or data structures that the theory requires will *a priori* be only things that it can construct from its observations. We call this property a *constructive theory of recognizability.* Second, if there is a correspondence between (parts of) the robot's data structures, and features or parts of the world, then the resolution of this correspondence will be no finer than the perceptual limit of the robot. Third, all the "places" in the world that the robot can "address" or "plan for" would be *a priori* recognizable.

We attack the research programme above as a principled exercise in spatial reasoning and experimentation. For example, one of our first tasks is to suggest precise definitions of the terms "resolution" and "perceptual limit" used in the paragraph above. We believe the benefits of our approach should be a framework for mobile robotic planning and reasoning which features *task-directed sensing*: that is, the sensing and actions that the robot undertakes are driven by the quantity and quality of its needs for information about the world. This paper concentrates on the basic concepts and constructions in our theory. We also discuss algorithms for computing these representations, both incrementally (from the world), and off-line (given a model).

2.3 An Architecture for Constructing Recognizability

The methods we propose may be described in terms of three layers, each building on the next. The first, definitional, layer consists mainly of the consequences of a principled, functional approach to examining robotic sensing. We begin with a functional definition of the action of a sensor. That is, a sensor is a projection from the world (and the robot's configuration) onto a sensor space. Issues of perceptual equivalence, perceptual aliasing, recognizability, sensor uncertainty (noise), and robot system capabilities arise here. An important feature of this approach is its close connection with a theory of Error Detection and Recovery (EDR) [Don, Don90]. In fact, robot programs developed within our framework exhibit "natural" EDR properties. For example, the goal (and subgoal) states of a mobile robot control program, when developed within our framework, are *a priori* recognizable. Thus, error detection is built-in.

When sensors are modeled functionally, one might think of ways to make use of various compositions of sensing functions. In fact, one may define "virtual sensors" in terms of existing "concrete sensors". We will describe a method for task-directed construction of such "virtual sensors". Virtual sensors are queried in robot programs much as their concrete counterparts are. In allowing the task to direct the composition of virtual sensors, we derive robot programs which are organized in such a way as to guide the robot toward acquiring the information it needs to accomplish the task; i.e. many information-acquisition and representational issues are made explicit. For example, consider a manuipulation task: the task of pushing a box stably along a wall. The mobot needs a way to sense the relative position of the wall and box, and also the dynamic relationship of the mobot to the box. Now, the motivation for constructing a "wall sensor" is simply that the task might have been described (i.e., specified) in human-oriented terms such as walls, boxes, etc. These perceptual categories may be indistinguishable to the robot with noisy sensors. In actuality, the robot will sense features of the world that are perceptually equivalent to some human-programmed notion of a wall. We can provide the robot with a virtual wall sensor, that is, a sensing and computation tree that combines current sensory information, history (retained state), and computation, and outputs a parameterized "inference" about recognizing an object in the same perceptual category as "wall." This "inference," then has two parts: a *symbolic* part, that identifies the inference with the task-level specification of a "wall," and a *parametric* part, that represents the parameters that can vary with respect to how the robot perceives a wall—eg, relative distance, orientation, extent, etc. Now we ask, how might this wall sensor work, and how would it be used in constructing a virtual sensor appropriate to this task?

First, a time-indexed history of the robot's controls may be integrated into an odometry sensor, and when odometry is in turn combined with the output of various range sensors, the robot may sense "$\mathtt{wall}(\alpha_1, \ldots, \alpha_n)$." For example, suppose the robot senses a range contact dead ahead. It can verify the hypothesis that this range contact represents a (piece of a) wall by attempting to move parallel to the hypothesized wall, and observing whether the range sensor returns the same (or nearly the same) orientation and distance readings. This is the sensing strategy that the virtual sensor for $\mathtt{wall}(\alpha_1, \ldots, \alpha_n)$ encodes, and the output of the sensor would be (in the case of success), the hypothesis of the existence of a wall, and an estimate as to its parameters. The key point is that this process of sensing, computation, retention of state, and the recursive construction and conditioning of virtual and concrete sensors can be viewed as a general geometric exercise in task-directed sensing. Our mission here is to elaborate that mechanism.

Hence, in particular, the intended "meaning" of $\mathtt{wall}(\alpha_1, \ldots, \alpha_n)$ is that the robot is near a wall with parameters $(\alpha_1, \ldots, \alpha_n)$, which describe the location, extent, and possibly other parameters of a wall. We have force and torque sensors attached to a "manipulation bumper" (for pushing) on the mobot. Next, the output these force sensors is combined with contact sensor information to report on the experimental dynamics of the robot's interaction with the environment. Suppose that a virtual box sensor has been defined in a similar manner to the wall sensor described above. A contact sensor can then detect contact with the box, and force and torque sensors report on the robot's interaction with the box. While translating along the wall in a stable way, the box stays in the same relative position to the robot. In addition, some nominal force and torque is required to continue moving the box; a large required force may mean that the box is stuck. The sensing strategy encoded by a box-pushing virtual sensor would output the relative location of the box, and also the dynamical properties associated with pushing it, such as point of contact, and applied force and torque. This sensor may also output the relative "drift" of the center of friction of the box relative to the robot. Such information could be used to alter the robot's controls such that it would continue to push the box without rotating it.

Finally, we may construct a task-directed virtual sensor that incorporates the wall sensor and the experimental dynamics of the robot's interaction with the box. This sensor is built such that it reports that the robot has achieved a goal state, has entered a terminal non-goal (failure) state, or that the robot should continue. Note the intrinsic EDR character of the resulting "program," in which goal and failure states are recognized when they are reached.

Information from physical sensors, previous state, and computation results, combine as task-directed virtual sensors are constructed. The flow of information through the hierarchy of such virtual sensors is quite explicit. Methods of construction of virtual sensors comprises the second layer of our approach.

The third layer describes how the tools we have built, such as virtual sensors, may be used to do task-level programming. The emphasis here is on how to make use of multiple internal representations of the world. We propose to support different representations using a *version-space*-like technique [Mit] to maintain data structures with respect to current and past observations.

3 Previous Work

Many of the issues in this paper have been encountered or have precursors in earlier work. The constructive theory of recognizablity—a calculus of the observable—and the idea of directing experiments based on information needs—these ideas are important themes in the classical physical sciences and in robotics. Our enterprise is not new! For example, in the context of human experimenters in quantum mechanics, R. Sorenson quotes Ernst Mach's *Contribution to the Analysis of Sensations*: [5]

> Thus, I lie upon my sofa. If I close my right eye, [this] picture is presented to my left eye[:] in a frame formed by the ridge of my eyebrow, by my nose, and by my moustache, appears a part of my body, so far as visible, with its environment. My body differs from other human bodies... by the circumstance, that it is only partially seen, and especially, is seen without a head. If observe an element A within my field of vision, and investigate the connexion with another element B within the same field, I step out of the domain of physics into that of physiology or psychology, provided B ... passes through my skin.

In short, Sorenson concludes, Mach believed:

> ... there is no such thing as a purely physical event or connection, but only physical events and connections filtered through our senses (our skin).

However, in our problem, the entities performing and analyzing the observations are machines. In robotics, strategies for coping with uncertainty, and reasoning about reachable and recognizable sets has been a theme in robotics research for many years. See, for example: [LMT, Erd, E89, Don, Brost, Can, Bri, DW]. Early work on planning in the presence of uncertainty investigated using skeleton strategies. [Lozano-Pérez 76] proposed a task-level planner called LAMA which used geometric simulation to predict the outcomes of plans, and is one of the earliest systems to address EDR planning. [Taylor] used symbolic reasoning to restrict the values of variables in skeleton plans to guarantee success. [Brooks 82] later extended this technique using a symbolic algebra system. [Dufay and Latombe] implemented a system which addresses learning in the domain of robot motion planning with uncertainty.

[LMT] proposed a formal framework for automatically synthesizing fine-motion strategies in the presence of sensing and control uncertainty. Their method is called the *preimage* framework. [Mason, 83] further developed the preimage termination predicates, addressing completeness and correctness of the resulting plans. [Erd] continued work on the preimage framework, and demonstrated how to separate the problem into questions of *reachability* and *recognizability*. He also showed how to compute preimages using backprojections, which address reachability alone, and designed and implemented the first algorithms for computing backprojections. Erdmann and Mason [EM] developed a planner which could perform sensorless manipulation of polygonal objects in a tray. Their planner makes extensive use of a representation of friction in configuration space [Erd]. See also [EVM]. [Buckley] implemented a multi-step planner for planning compliant motions with uncertainty in 3D without rotations. He also developed a variety of new theoretical tools, including a combined spring-damper dynamic model, 3D backprojection and forward projection algorithms, and a finitization technique which makes searching the space of commanded motions more tractable. Donald [Don] proposes error detection and recovery (EDR) strategies for planning and executing robot strategies in the presence of uncertainty. Canny [Can] showed the following:

[5] See "Thought Experiments", by R. Sorensen in *Amer. Scientist*, 79 (3), (1991) pp. 250-263.

when the geometric world models and also the solution trajectories to the differential inclusion governing the control system are semi-algebraic, then general LMT motion strategies can be computed using the theory of real closed fields. These assumptions cover (for example) motion planning with uncertainty under pure translation. Probabilistic planning and execution strategies are developed by [GM89, E89], and greatly enhance the LMT-family framework. [Lumelsky] considers navigation problems for 2DOF robots with perfect dead reckoning but no model of the world. Other planners, and techniques for coping with uncertainty are discussed by [Brost; Caine; Cutkosky; LLS; Laugier; PTP; Briggs; FHS].

Notions of recognizability developed in the LMT family tree are important to us even without world models and the ability to forward- and back-simulate. Erdmann formalized the "effective set of sensor interpretations" using history and sensing [Erd]. Canny called these the *recognizable sets*, and demonstrated a semi-algebraic parameterization for them [Can]. This allowed him to apply very general tools from computational algebra in order to obtain planning algorithms when one has geometric world models and algebraic forward projections. Canny's planner is based on the recognizable sets. Erdmann also made extensive use of these "sensor interpretations sets" in his work on probabilistic strategies [E89].

In this paper, we develop a family of concepts useful in mobile robotics. These include (for example): *perceptual equivalence classes,* the *lattice of recognizable sets*, the *reachability graph of recognizable sets*, the *perceptual limit*, *virtual sensors*, the *recognizability gradient*, *local sections* to encode expecations, the *reconstruction* problem, and a "measure" of the "reactivity" of a system. These ideas are connected with previous work. See the papers by Brooks, Agre, Chapman, Kaelbling, Rosenschein, Maes, Steels, Payton, Arkin, Malcolm, Smithers, Anderson, Donath, Beer, Chiel, and Sterling collected in [Maes] for a survey of some current work in autonomous agents. More details on reactive systems, behavior-based robots, and the subsumption architecture are discussed by Brooks [Brooks 1985], Mataric [Mat], and Mahadevan and Connell [MC]. The last describe the use of reinforcement learning in the mobile robot domain. In work on learning manipulation strategies, Christiansen, Mason, and Mitchell [CMM] identify perceptual equivalence classes as consequences of imperfect sensing, which may be viewed as a projection [C]. Whitehead and Ballard [WB] and also Shafer [S] discuss perceptual and model aliasing. Shafer [S] describes how model aliasing can partition the set of world states into equivalence classes with the same representation. We believe that [MC, CMM, C, WB, S, E, Can] understand and exploit notions of perceptual equivalence classes in a fundamental way. Reachability graphs have many ancestors; see even [Min]. Our lattice is reminiscent of Mitchell's *version space* [Mit]. Leonard, Cox, and Durrant-Whyte [LDWC, CL] use notions of observation and expectation that we find significant and these notions anticipate some of our constructions. Notions of intelligent sensors and processing to disambiguate perceptual aliases can be found in [KS, BaKu, KTB]. Kriegman, Triendl, and Binford, and Moravec and Elfes, explore map-making, navigation, and landmark recognition [KTB, ME, Mor], and they use geometric reasoning techniques to handle uncertainty. The work of Hutchinson on visual termination predicates [Hut] is an interesting form of task-directed sensing. In [Don90] we suggested the following approach, called *"weak error detection and recovery (EDR)."* Suppose we are given a collection of subgoals that are closed under intersection. Then by considering all unions of subcollections of these subgoals we can obtain a measure of weakest recognizability. This idea anticipates the lattice of recognizable sets, whose roots also reach to Erdmann's work on preimages [Erd]. Finally, we believe the ideas of Rosenschein [R] to be of considerable relevance. See also [Dre, Su] for related work. This paper is an expansion of our short conference paper presented in [DJ].

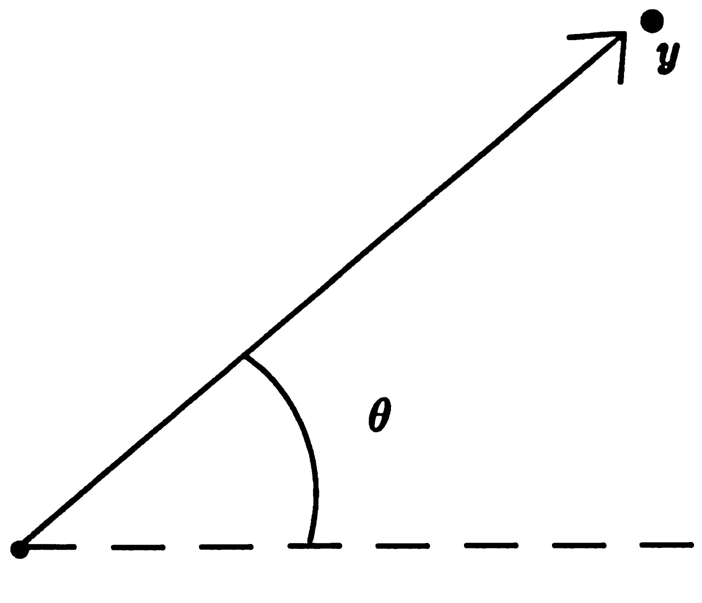

Figure 1: A mobile robot at configuration x is pointing its range finder in orientation θ to interrogate a point y in the environment.

4 Perceptual Equivalence Classes

4.1 Quotient Structure

To illustrate our ideas we will develop a general albeit somewhat simplified model of sensing. In a particular, we will develop examples of perceptual equivalence classes and their structure. On a real mobot our examples generalize, but are more complicated (see the appendix).

One key observation is that there are places in the environment that are distinct, yet appear the same when viewed through the robot's sensors. For example, consider an (idealized) mobile robot with a range sensor.[6] Suppose the range sensor only outputs discrete range values, so that objects d length units away register as value $\lfloor d \rfloor$. Now, we'll model the robot's sensing operations as follows. Suppose the robot is at a fixed configuration x, and it points the range sensor in direction θ (which it knows). See fig. 1. Consider a point y in the world. So $y \in \Re^2$, and θ is y's direction from x. We imagine the robot "interrogating" y. We can represent the sensor "value" for y as being one of the following (see fig. 2):

1. An integer $\lfloor d \rfloor$, if there is an obstacle at y and there are no obstacles on the line segment $[x, y)$.
2. The value $\emptyset$ if there are no obstacles on the line segment $[x, y]$.
3. The value ∞ if there is another obstacle on the line segment $[x, y)$.
4. The value $\perp$ if the direction (angle) of line segment $[x, y]$ is not equal to θ.[7]

If Z^+ denotes the non-negative integers, we can view the space B of sensor values as

$$B = Z^+ \cup \{\emptyset, \infty, \perp\}. \tag{1}$$

Hence we can define our sensor as a mapping $p_x(\theta, y) = b$ for b in B. It is clear that this mapping induces a partition of the world into equivalence classes of points that will appear the same under the sensor. These equivalence classes are conceptually sketched in fig. 3. If we disregard θ, two obstacles at roughly distance d will appear the same, even if they are at different angular locations.

[6] There are other ways to present this example, of course.

[7] The value $\perp$, a "sensor error," is necessary for technical reasons. We will not discuss it in any detail for fear of obscuring more important points.

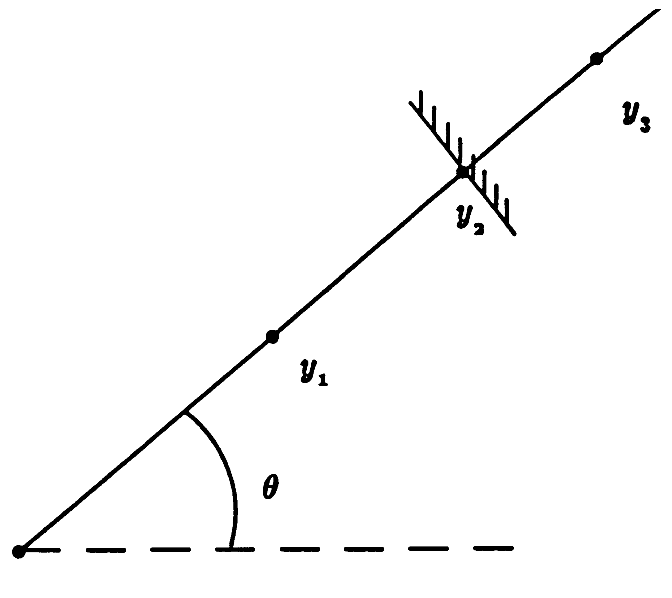

Figure 2: Suppose we have the world $M = \Re^2$. Consider points y_i in the world, which are in direction θ from x. We imagine the robot "interrogating" y_i by pointing the sensor in direction θ. In this example, there is an obstacle at y_2, which is distance d from x. We have $p_x(\theta, y_1) = \emptyset$, $p_x(\theta, y_2) = \lfloor d \rfloor$, $p_x(\theta, y_3) = \infty$.

(The reader should not be alarmed; we will take θ into account later and use it to discriminate among the equivalence classes in fig. 3.) We see this as follows. Consider fig. 3. We can construct the perceptual equivalence classes for the world (i.e., the geometry) in fig. 3 by studying the line in fig. 4, anchored at x. It suffices to consider a line, due to radial symmetry. Now, fig. 4 represents the distance of objects in the geometric environment depicted in fig. 3. In particular, the intervals W_2 and W_4 correspond to distances (from x) of the walls (surfaces) that contain points y_2 and y_4 in fig. 3. d_2 is the distance of y_2 and d_4 is the distance of y_4. Suppose that $\lfloor d_w \rfloor = d$ for all d_w in W_2 or W_4, and in particular $\lfloor d_2 \rfloor = d = \lfloor d_4 \rfloor$. Beneath the line in fig. 4 we have indicated the sensor values that are possible results when we interrogate points at various distances in the world (the distance is plotted along the line). Points $y \in \Re^2$ at distances less than W_2 from x must return sensor value $\emptyset$. If the y's distance from x lies in the interval $W_2 \cup W_4$, then the sensor value returned can be $\emptyset$, d, or ∞. If y's distance is greater than W_4, then, the the sensor must return either $\emptyset$ or ∞. The perceptual equivalence classes for this geometry may be generated as follows. All the sets on the line in fig. 4 are intervals. Each interval has a "label" (a possible sensor value). Form the arrangement of all these intervals. The label of a cell in the arrangement is a list of all possible sensor values. Sweep the resulting arrangement radially about x. This generates annular perceptual equivalence classes. (We view a disk centered on x as a kind of degenerate annulus). The width (difference of radii) of each annulus is at least 1. We see that in general: for the sensing map[8] p_x in fig. 3, the perceptual equivalence classes are always annular, and always in one-to-one correspondence with a linear order.

4.2 Modeling Sensors

Sensors are modeled, then, as functions which project the world, M, onto a sensor space, B. Note that for this example, we do not choose to model the range finder as a simple mapping from controls to (say) integers representing, eg., time-of-flight for a sonar ping. That is because we wish to define a sensing map that encodes the *information* obtained in the return ping. The information contained in a (non-infinite) return ping is not simply "There exists an obstacle at distance d." It is considerably more:

> *(*) "There exist no obstacles up to distance d; there exists an obstacle at distance d; and beyond d we have no information."*

[8]Throughout, we use *map* to mean a mathematical function, not a representation for the robot's spatial environment.

Hence, we developed this sensor map as follows. We began by asking the question:

(Q1) Given a point y in the world M, what effect does y have on the values we sense?

We call the answer to this question the *pointwise* definition of the sensing map: it gives us the rule $p_x(y) = b$. The presentation of sensing maps here is simplified to emphasize the main points. Key additional components include (a) how to model more complicated sensors and (b) the impact of sensor noise. Below, and in app. B we present more details of how more complex physical sensors may be modeled in this way. The simplified framework described here can be augmented to handle noise. See appendix C. We also note that the value b could represent a probability distribution, as in [ME, Mor]. See appendix B.

4.3 Perceptual Limits

Now, a natural question to ask is: for a given sensing map p_x, what are the finest possible perceptual equivalence classes? In this case, the answer is clear: the finest possible perceptual equivalence classes the robot could distinguish are the unit annuli centered on x with integral inner and outer radii. See fig. 3. Now, call this partition (into unit annuli) $\overline{\mathcal{U}}$. We notice that $\overline{\mathcal{U}}$ is at *least as fine* as any partition of the world into perceptual equivalence classes that we will encounter given some *particular* geometry. For example, $\overline{\mathcal{U}}$ is finer than the partition induced by the geometry in fig 3 and which is induced by sweeping the line in fig. 4 about x into annuli. Here is how one may formalize this notion: it is key to understanding our idea. Let G_α represent a world geometry, such as the walls and so forth in fig. 3. Now let $\mathcal{U}_\alpha$ denote the resulting partition of the world into perceptual equivalence classes, for example, the annuli swept out by the linear arrangement in fig. 4. For another geometry G_β we obtain a different partition $\mathcal{U}_\beta$ and so on. For any two partitions $\mathcal{U}_\alpha$ and $\mathcal{U}_\beta$, one (say, $\mathcal{U}_\beta$) may be finer than the other ($\mathcal{U}_\alpha$), or else they will be incomparable. When $\mathcal{U}_\beta$ is finer than $\mathcal{U}_\alpha$, we write $\mathcal{U}_\alpha \hookrightarrow \mathcal{U}_\beta$. To say that $\overline{\mathcal{U}}$ is the *finest* partition possible for p_x is to say that for any geometry G_α, $\mathcal{U}_\alpha \hookrightarrow \overline{\mathcal{U}}$. Of course, there always exists some such $\overline{\mathcal{U}}$ trivially (eg, consider the 1-point sets) but the interesting case is where the sets have finite extent (as in this example with p_x). Let G_α range over the set of all geometries. The "finer" relation structures the resulting perceptual equivalence class partitions into a lattice. This lattice is dominated by its direct limit, which is $\overline{\mathcal{U}}$. The map we are considering, p_x, is interesting in that despite the existence of an uncountable infinity of possible world geometries G_α, only a finite number of partitions into perceptual equivalence classes $\mathcal{U}_\alpha$ are possible. The lattice in this case simply reduces to a *finite*[9] directed graph whose nodes are the $\{\mathcal{U}_\alpha\}$ and whose edges are the $\hookrightarrow$ relationships. The graph has one "sink" which is the "minimal" partition $\overline{\mathcal{U}}$ into unit annuli. This direct limit $\overline{\mathcal{U}}$ precisely quantifies the capabilities of the sensing map p_x, and hence we call the partition $\overline{\mathcal{U}}$ the *perceptual limit of the sensing map* p_x.

The perceptual limit has the following constructive significance. For a finite lattice, any obtainable partition $\mathcal{U}_\alpha$ of the world into perceptual equivalence classes can be constructed by finite unions of the classes in the perceptual limit $\overline{\mathcal{U}}$. Hence, we think of a geometry G_α as "controlling" how the classes in $\overline{\mathcal{U}}$ are unioned to obtain $\mathcal{U}_\alpha$.

Now, it is clear how the equivalence classes in fig. 3 arise. The equivalence classes form a partition of the world (points y in M). The equivalence relation is that the points yield the same sensor reading when viewed or "projected" through p_x; compare [C]. Finally, it is clear that p_x is a kind of quotient map, and that the equivalence classes are induced by p_x. They are generated by the inverse images $\{p_x^{-1}(b)\}$ for sensor values b in B.

We now elaborate on this point. We view the mapping p_x as follows. Let S^1 denote the unit circle, so we may say $\theta \in S^1$. We view points y in the world as living in a space we denote as M. Here, M is the plane $\Re^2$.

[9]The finiteness assumption assumes the range sensor has a maximum range.

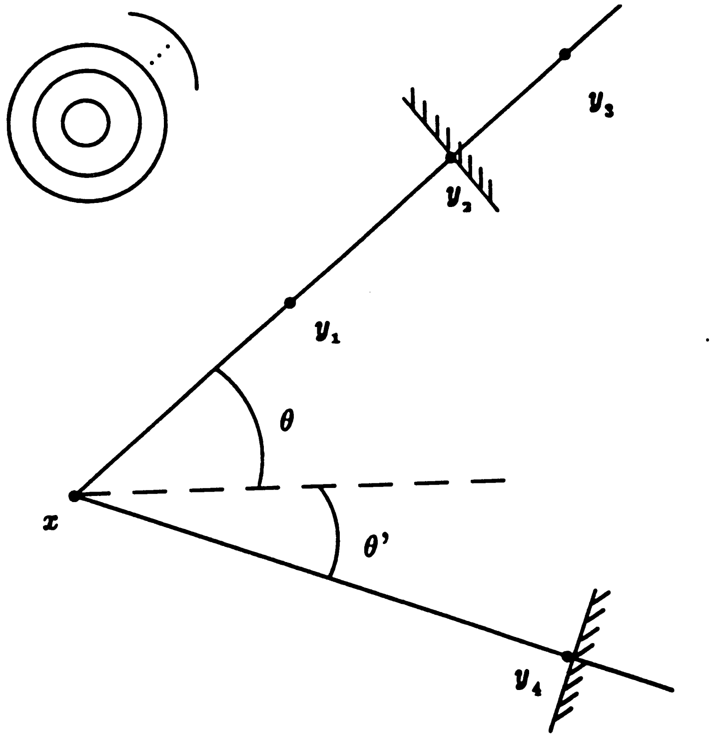

Figure 3: If we disregard θ, two obstacles at roughly distance d will appear the same, even if they are at different angular locations. Hence, since $p_x(y_2, \theta) = p_x(y_4, \theta')$, y_2 and y_4 are in the same perceptual equivalence class. The *perceptual limit* of this sensing map is given by the concentric annular sets shown here.

$$\begin{array}{cccc} p_x & : & M \times S^1 & \rightarrow & B \\ & & (y, \theta) & \mapsto & b. \end{array} \tag{2}$$

Let π_M be the natural projection of $S^1 \times M$ onto M, so $\pi_M(y, \theta) = y$. We now see the following

Definition 4.1 *At a fixed configuration x, the perceptual equivalence classes are classified by sets of the form*

$$\pi_M p_x^{-1}(b) \tag{3}$$

for a sensor value b in B. That is, each perceptual equivalence class is a cell in the arrangement generated by sets of the form (3).

Now, it is easy to extend our example to take θ, the direction of the sensor, into account. By remembering θ, the robot may distinguish points within one annulus. Consider fig. 5, which shows the resulting equivalence classes for a world with three walls. Any regions with the same "value" are perceptually indistinguishable.

If we change sensing maps, then the perceptual limit can change too. We now ask, what is the perceptual limit of this new sensing map? In this case the minimal perceptual equivalence classes will be finer than the unit annuli in $\overline{\mathcal{U}}$, and we have conceptually sketched them in fig. 6. Call this new minimal partition $\overline{\mathcal{U}}^*$. The perceptual limit $\overline{\mathcal{U}}^*$ of the world (around x) is finer than the perceptual limit $\overline{\mathcal{U}}$ (the annuli) in fig. 3. The perceptual equivalence classes in $\overline{\mathcal{U}}^*$ are isomorphic to radial line-segments of unit length at orientation θ and integral distance from x.

Now, we have noticed that the partition $\overline{\mathcal{U}}^*$ of M in fig. 6 is *finer* than the partition $\overline{\mathcal{U}}$ in fig. 3. Corresponding to this finer partition is a "finer" map p_x^* which takes θ into account in the natural way. Intuitively, p_x^* should be be a quotient map[10] $p_x \times \mathrm{id}_{S^1}$ with range $B \times S^1$ that induces the

[10]See the appendix A.1. id_{S^1} denotes the identity map on the unit circle S^1.

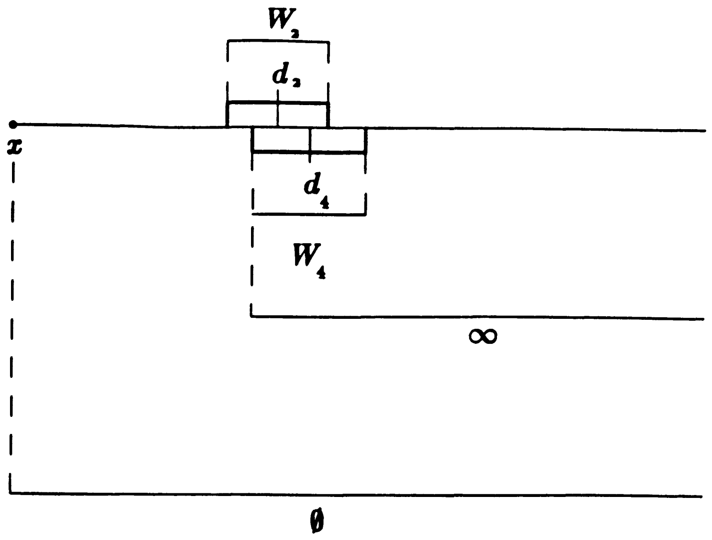

Figure 4: This line represents distances of points $y \in \Re^2$ in the world from x. It is annotated with the different sensor values possible for such points under the sensing map p_x. The intervals W_2 and W_4 denote the distances of points in the two walls depicted in fig. 3.

---

partition into perceptual equivalence classes depicted in fig. 6, by using information about θ to disambiguate objects at the same distance but at a different relative orientation.

4.4 Lattice Structure

4.4.1 Robot Capabilities

In sec. 4.1, we saw how, once we fixed a sensing map (like p_x), different environments can generate different partitions into perceptual equivalence classes. We suggested that all these partitions can be partially ordered on fineness. The resulting structure is a *lattice*, and it is dominated by its direct limit, which we termed the *perceptual limit* of p_x. The perceptual limit represents the "smallest features" the robot can interdifferentiate, in the best case. So, the perceptual limit is obtained by fixing the sensing map p_x and varying the environment.[11]

We now consider fixing the environment (world geometry), and varying the sensing map. That is, we consider varying:

(a) the sensing modalities,

(b) the computation used to interpret the sensors, and

(c) the amount of "state" or history the computation in (b) can retain.

It is important to realize that history (b) and computational power (c) are distinct concepts. *Computational power* (b) is characterized by the time and space complexity of the computation that the robot can perform. *History* (c) is characterized by storage requirements or information complexity. (Compare [E89]). To see this distinction, imagine two extremes:[12]

1. A robot with the power to perform any polynomial-time computation, but which remembers no state.

2. A robot which can remember lots of state but can only perform $O(1)$ work using that state.

[11] In our dicsussion, this is tantamount to varying the world geometry. However, more generally, one could vary other properties–eg, color, motion, or texture.

[12] These are not supposed to be realistic examples, but merely illustrative.

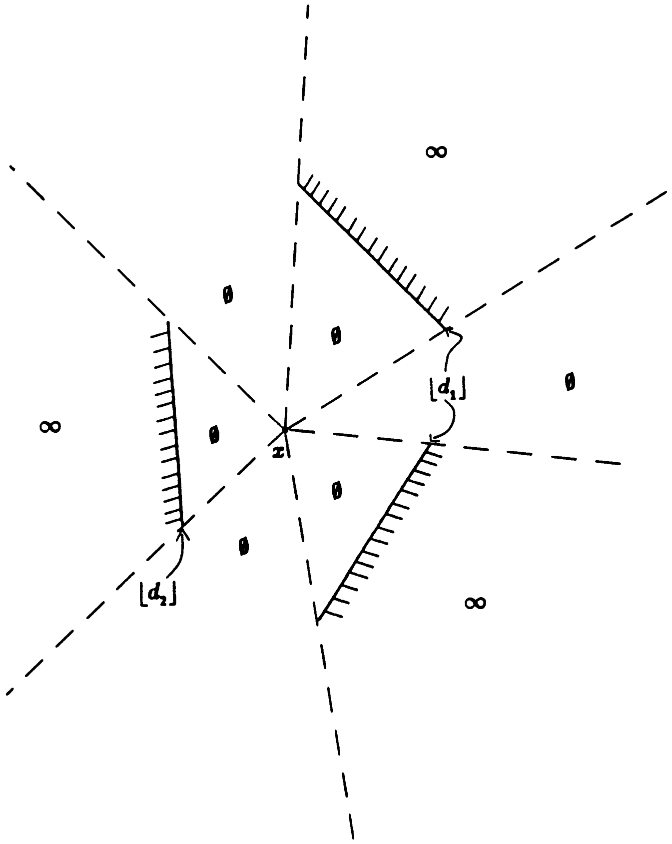

Figure 5: Suppose the robot uses θ, the direction in which the sensor is pointing, to distinguish between obstacles at the same distance, but at a different relative orientation. Then the world (around x) is divided into a finer partition than in fig. 3. Here is an example of a world with three walls. The resulting perceptual equivalence classes are 3 disconnected sets with value ∞, two disconnected linear sets with value $\lfloor d_1 \rfloor$, another linear set with value $\lfloor d_2 \rfloor$, and a large star-shaped non-convex region with value $\emptyset$.

Hence robot 1 can be booted or have its storage erased each time-step, without impeding its performance.

We call a sensing map p_x that uses no history a *raw* sensing map. Now suppose that we choose to use history in addition to interrogating the world through p_x. We will see this is equivalent to using a *finer* raw sensing map p'_x (i.e., one that results in a finer partition of the world). That is, loosely speaking, we have that

$$p_x + \text{history} \cong p'_x.$$

We call the finer map p'_x the *effective* sensing map, hoping that this term resonates with Erdmann's "effective set of sensor interpretations" [Erd], which he obtained by combining history with sensing. Each change in sensing, computational power, and state can result in a new (effective) sensing map, and hence a new partition of (a particular) world into perceptual equivalence classes. As before, in sec. 4.1, all these partitions (now for a fixed geometry!) can be arrayed in a lattice, and partially ordered on fineness.

4.4.2 Looking Ahead

In the rest of this paper, we discuss the above ideas in some detail and develop applications and algorithms. The model is augmented by considering the effect of sensor noise and error. We

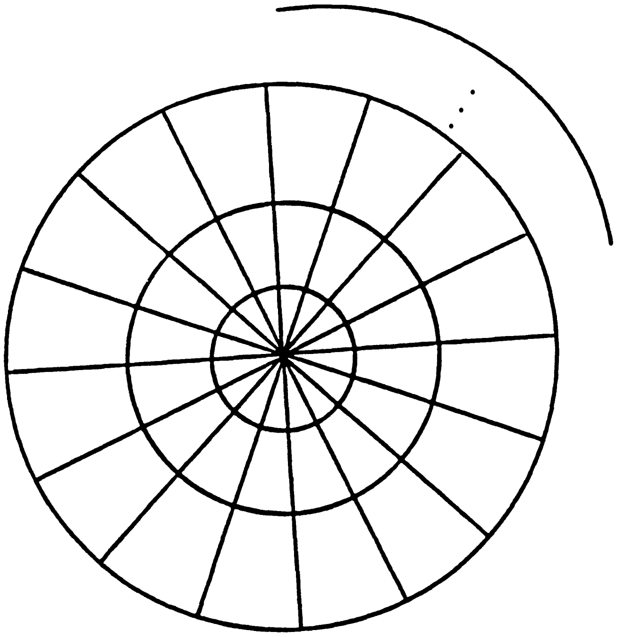

Figure 6: Consider a sensing map p_s^* that takes θ into account. The *perceptual limit* $\overline{U}^*$ of p_s^* is shown. The perceptual limit is the partition into the finest possible perceptual equivalence classes under this sensing map. Each equivalence class lies on a line through x, has unit length, and starts and ends at an integral distance from x.

identify global landmarks, which are the recognizable sets in configuration space, called the C-perceptual equivalence classes. We can compute them, given a world model, and we can give efficient algorithms for doing so. In the absence of a model, these "landmarks" can be acquired incrementally, through exploration. A graph of these explorations, called the *RR-graph*, encodes the reachability relationships between the recognizable sets. Again, it may be computed from a model, but in the absence of a model we can "encode" it using the sensor values and history alone. For a sensor space B, we call this a "B-encoding" of an RR-graph. This representation seems useful for mobots with minimal models.

RR-graphs and C-perceptual equivalence classes are simple concepts; these recognizable sets form the basic building blocks for our theory. These landmarks, or recognizable sets—that is, the places in the world that the robot can recognize and distinguish between—have structure we can exploit in developing sensor-based planning algorithms. In particular, the interaction of the lattice and history is subtle and can be used to plan or program sophisticated strategies for achieving goals and accomplishing tasks.

4.4.3 Representing Motions

A fundamental question in characterizing the utility of a robot strategy is: what information is gained (or lost) by this strategy? More generally, one wishes to represent the knowledge of the robot's state before and after an action, and to measure, or compare the knowledge states in order to determine whether uncertainty as increased or decreased.

The situation we encountered in sec. 4.1 is an example of a more general phenomenon. Let us introduce this phenomenon by contrasting our problem of robot strategy synthesis under large uncertainties with another, drastically easier problem: the gross motion planning problem with no uncertainty (perfect control, sensing, and models). In this problem, the state of the robot may be represented as a point in a configuration space. Thus moving from a start to a goal point may be viewed as finding an arc in free space connecting the two points. Since the robot is assumed to have perfect control and sensing, any such arc may be reliably executed once it is found. In particular, given a candidate arc, it may be tested. That is, motion along the arc may be simulated to see whether it is collision free. For example, an algebraic curve may be intersected with semi-algebraic sets defining the configuration space obstacles. In the presence of uncertainty, however, we cannot simply simulate a motion strategy to verify it. Instead, we need some technique for simulating *all* possible orbits, or evolutions of the robot system, under any possible choice of the uncertain

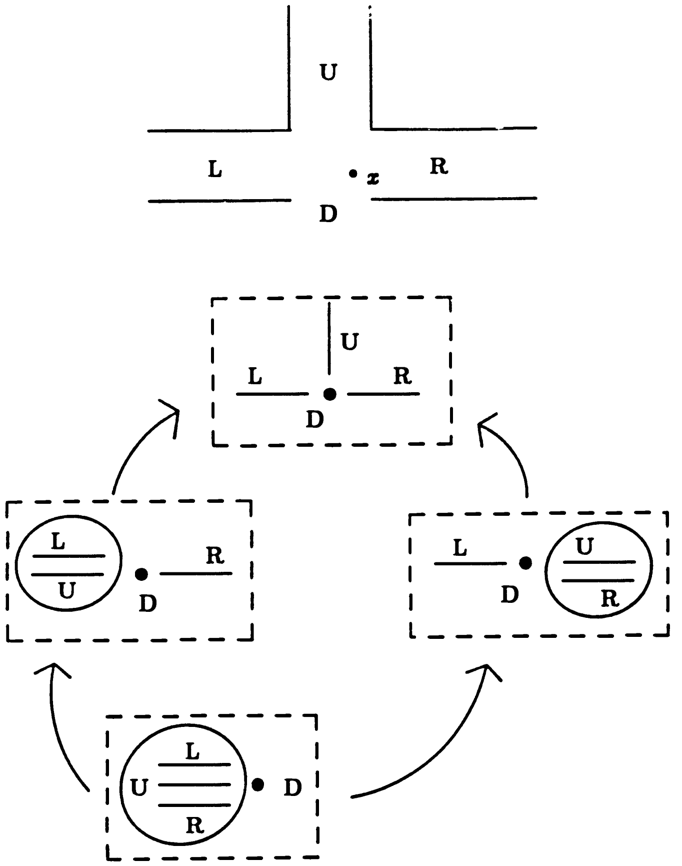

Figure 7: A sketch of the area in front of my office door, and part of the lattice of knowledge states. This lattice is (6).

---

parameters.[13] With sensing and control uncertainty, the state of the robot must be viewed as a subset of the configuration space. Motions, then, can be viewed as mappings between these subsets.

As an example, consider fig. 7, a sketch of the area in front of my office. Three corridors, labeled Left (L), Up (U), and Right (R) connect at an area near my office door (D). The top figure denotes the idealized model of the world, M. This neighborhood may be segmented into perceptually distinguishable equivalence classes as in sec. 4.1. Depending on the resolution and accuracy of the sensing, the equivalence classes might look like $\{L, U, R, D\}$ (see fig. 7). So one motion might move the robot from L to D and another might move from D to U. Hence, together with the partition of M into recognizable classes, we obtain a "reachability graph" whose vertices are perceptual equivalence classes (what we call the *recognizable* sets), and whose edges are labeled with motions or strategies. The edges may be directed if the motion is not reversible. The connectivity of this graph might be (for example):

$$\begin{array}{ccccc} & & U & & \\ & & \updownarrow & & \\ L & \longleftrightarrow & D & \longleftrightarrow & R. \end{array} \tag{4}$$

---

[13]That is, we need some method that can bound all possible behaviors of the system.

4.4.4 Reachability Graphs of Recognizable Sets

The reachability relationship is the natural analogue of path-connectivity. We call a graph such as in (4) an *RR-graph*, which stands for a *reachability graph of the recognizable sets.* Hence it is clear that motion of the robot induces a motion in the RR-graph. Viewing the motion in the RR-graph as a "mapping", we call it a *reachability map.* Each reachability map corresponds to a connected sequence of edges in the RR-graph. An example of a reachability map is: a strategy (or motion) v takes the robot from class D to class L. We denote this by $D \xrightarrow{v} L$.

However, not all motions will be as "well-behaved" as a motion such as (for example) $D \xrightarrow{v} L$. Suppose that due to uncertainty in control and sensing, we cannot guarantee that we will move to L, but that it is possible that we remain in D, or even move to U "by mistake." Suppose further that after executing the motion (call it v), we can tell (by sensing) whether or not we are still in D, but that L and U will be indistinguishable.

In this case we have the transition $D \xrightarrow{v} \{D, LU\}$, where LU denotes the union of sets L and U. In essence, this means that after executing the strategy v, we know the robot is in D, or it is in L or U (but we don't know *which* of L or U).

Now, clearly the motion v descends to a state of knowledge that is less certain. We have the following idea for how to measure this change in information of the knowledge state. We imagine another partition of M into the perceptual equivalence classes $\{LU, R, D\}$. This means that L and U are perceptually indistinguishable in the new partition. It should be clear that such a partition, in which there is less information about the world and the perceptual equivalence classes are larger could arise naturally due to less accurate sensing, and its existence is independent of strategies and motions like v.

First, we remark that each new partition such as $\{LU, R, D\}$ comes equipped with its own RR-graph. Hence, we can think of the transition $D \xrightarrow{v} \{D, LU\}$ as being the composition of two mappings: a *forgetful* map and a reachability map:

1. The first map descends *in the lattice* shown in (6). It is a map from one partition to a less fine partition. This map is specified by:

$$\begin{array}{ccc} \{L, U, R, D\} & \to & \{LU, R, D\} \\ L & \mapsto & LU \\ U & \mapsto & LU \\ R & \mapsto & R \\ D & \mapsto & D. \end{array} \tag{5}$$

   We call the map (5) the *forgetful map,* since it loses information (and "forgets" whether we were in L or U). It is clear that for any two partitions $\mathcal{U}$ and $\mathcal{V}$, if $\mathcal{U}$ is finer than $\mathcal{V}$, then there is a well-defined forgetful map from $\mathcal{U}$ to $\mathcal{V}$.

2. The second map is simply a reachability map in the less fine partition $\{LU, R, D\}$. That is, it is a self-map of $\{LU, R, D\}$. In this case it takes (non-deterministically) D to D or D to LU.

Hence, in the same way that sensing operations that provide more information will "ascend" to a finer partition, it is clear that motion strategies can "descend" to a less fine partition (with less information). Of course, motion strategies can also gain information; for example, they can use compliant motion or wall-following to reduce uncertainty [Don].

4.4.5 The Lattice of Perceptual Equivalence Classes

One might think that strategies that lose information are to be avoided; however, sometimes they are necessary since attaining some goal may require descending to a more ambiguous state that contains the goal as one possibility (e.g. as a subset). To investigate this situation, we must better understand the relationship between partitions.

Consider the new partition $\{LU, R, D\}$. Clearly, the new partition is "less fine" than $\{L, U, R, D\}$ in the same sense that the partition in fig. 3 is less fine than fig. 6. We denote this relationship of "finer partition" by placing $\{LU, R, D\}$ lower than $\{L, U, R, D\}$ in the lattice shown in fig. 7. There are other partitions we could imagine, such as $\{L, UR, D\}$ that are incomparable in fineness to $\{LU, R, D\}$ but are both finer than (say) $\{LUR, D\}$ and less fine than $\{L, U, R, D\}$:

$$\begin{tikzcd}
& M & \\
& \vdots \arrow[u] & \\
& \{L,U,R,D\} \arrow[u] & \\
\{LU,R,D\} \arrow[ur] & & \{L,UR,D\} \arrow[ul] \\
& \{LUR,D\} \arrow[ul] \arrow[ur] & \\
& \{LURD\}. \arrow[u] &
\end{tikzcd} \tag{6}$$

We think of a partition $\mathcal{V}$ as "including" in a finer one $\mathcal{U}$ and since we think of the arrow relationships in (6) as being like inclusion maps, we will call them *inclusions*.[14]

4.5 Inclusion Maps Model Sensing Strategies

Now, consider sec. 4.1, and figs. 3 and 6 in particular. Suppose that the partition shown in fig. 6 is denoted $\mathcal{U}$, and fig. 3 is denoted $\mathcal{V}$. Then by the observation above, there is an inclusion $\jmath : \mathcal{V} \hookrightarrow \mathcal{U}$. What is the nature of this inclusion $\jmath$? The map $\jmath$ takes a coarse perceptual equivalence class V (in this case, an annulus) to a "fine" equivalence class U (a line segment, as shown in fig. 8).

Now, which line segment U is the image of V? Clearly, we must have $U \subset V$. In physical terms, the answer depends on θ. Here is the key point: passing from the coarser partition to the finer one corresponds to gaining information. The information is gained by using another sensor, or a better sensor, or by interpreting the information in the sensor better. We name this process, and call it *the application of sensing operation* $\jmath$. The image U of V under $\jmath$ is the result of applying operation $\jmath$ to the known state V. That is, the image will depend on the "better" sensing strategy's outcome. The manner in which V is refined into some (sub)set U depends on the exact state of the world and the result of the sensor operation.

However, even though we cannot always predict, ahead of time, what $\jmath(V)$ will be, we may be able to characterize this set in some way. For example, suppose V is the unit annulus with integral inner radius d. U will depend on θ (which $\jmath$ "senses"). V and θ completely characterize U, and yet θ is unknown before applying $\jmath$. Let $r(x, \theta)$ be the ray from x in direction θ. Then we know that

$$\jmath(V) = U = V \cap r(x, \theta).$$

Hence, we have a complete classification of the kind of set that can result from applying operation $\jmath$. Facts deduced from this classification–e.g. size, shape, dimensions, bounds–can be used in deciding whether performing the operation and increasing the information about the state is useful.

We note that these inclusions naturally model the application of a sensing strategy that gains information. An inclusion maps from a partition to a finer one. The image of a particular knowledge state under the inclusion depends on the result of the operation and cannot necessarily be predicted

[14]In general, given two distinct partitions $\mathcal{U}$ and $\mathcal{V}$ of M into perceptual equivalence classes $\mathcal{U} = \{U_\alpha\}$ and $\mathcal{V} = \{V_\alpha\}$, either $\mathcal{U}$ or $\mathcal{V}$ will be finer, or they will be incomparable.

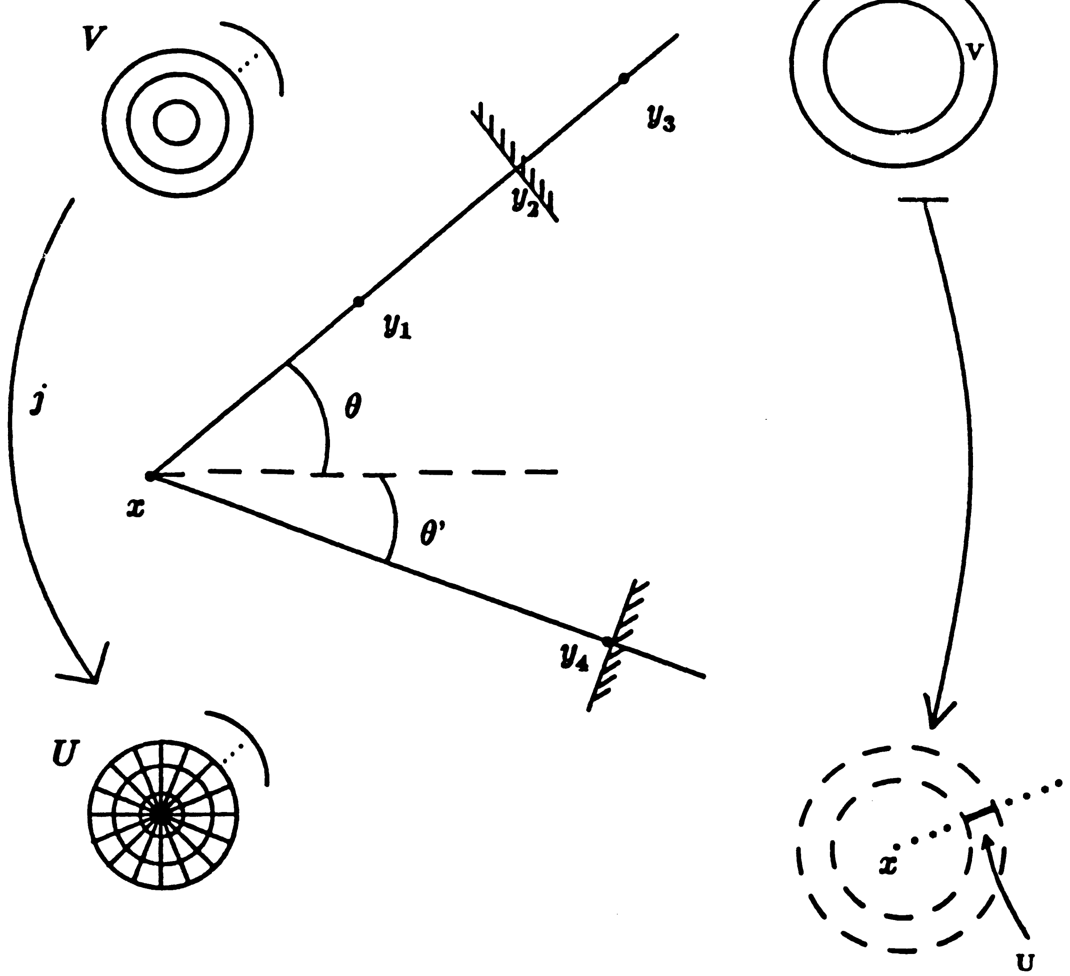

Figure 8: **Illustration of the inclusion map. $\mathcal{V}$ is the partition obtained by a sensing strategy that merely interrogates the distance of objects. j models a "better" sensing strategy that interrogates the relative orientation θ as well. The image of equivalence class $V \in \mathcal{V}$ will be a set $U \in \mathcal{U}$ of the form $U = V \cap r(x,\theta)$, where $r(x,\theta)$ is the ray from x in direction θ.**

---

in advance of trying the sensing strategy. (In this sense they are "dual" to motion strategies under uncertainty). However, using knowledge of the sensors, often the space of recognizable sets (in this case, $\mathcal{U}$) can be characterized, and this characterization can be used in planning and choosing actions.

4.6 The Local Quotient is a Bundle

A key phenomenon of sensing is that the robot's perception of the environment changes when the robot has moved. For example, suppose x and z are configurations of the robot, and z is "far" from x. Then from x, objects in the world "near" x appear finely distinguished, while objects in the world near z all blur together. When the world is viewed from z, the situation is reversed: objects near z appear finely interdifferentiated, while the environment near x is identified to a single perceptual equivalence class. We wish to understand, and to model this phenomenon; that is the the purpose of this section. The model we build will help us in characterizing planning and sensing in uncertain domains.

For example, suppose from configuration x, near my office door, the perceptual equivalence classes and RR-graph are given by this variant of (4):

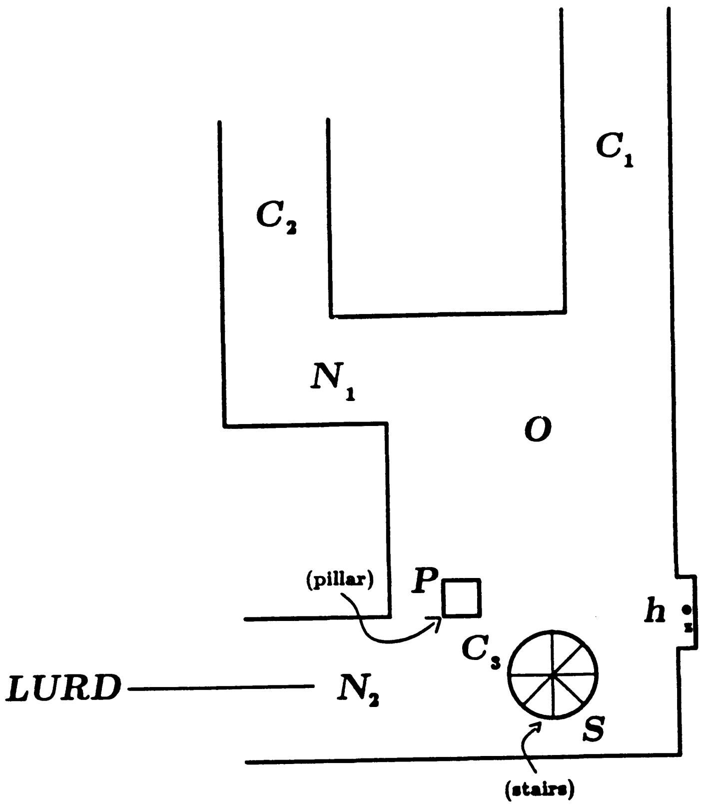

Figure 9: Diagram of the area near Hopcroft's office. Note that z is in h, the area near Hopcroft's door. S is a spiral staircase. P is a pillar.

$$\begin{array}{ccccccc} & & U & & & & \\ & & \updownarrow & & & & \\ L & \longleftrightarrow & D & \longleftrightarrow & R & \longleftrightarrow & H. \end{array} \tag{7}$$

For convenience, suppose $x \in D$. Here H denotes the area down a rather long hall, near Prof. John Hopcroft's office, which is near configuration z. (We will have $z \in H$). From "my" perspective, near x, the entire environment near z is indistinguishable, and just looks like one point, one equivalence class, which I'll call H. Now suppose we send the robot to take Hopcroft a cup of coffee, and the robot moves from x to z. This induces a reachability map (a motion in the reachability graph (7)) such as $D \longrightarrow R \longrightarrow H$. Now, the key issue is, once we are at z, the world near x no longer segments into fine perceptual equivalence classes as in (7). Instead, the environment near x (when viewed from z) becomes indistinguishable, and just looks like one point, one equivalence class, which I'll call $LURD$. The area near Hopcroft's office, where the robot is now, now reveals its finer structure and no longer looks like one undifferentiated perceptual equivalence class H, but instead breaks up into a finer partition. Suppose the neighborhood near Hopcroft's office looks like fig. 9.

Hence, when we reach z the RR-graph for the environment around z might look like this: (note that the stairs S and pillar P are obstacles. For simplicity, we've left P out of the graph):

$$
\begin{tikzcd}
 & & C_2 \arrow[d, leftrightarrow] & \\
 & & N_1 & C_1 \arrow[d, leftrightarrow] \\
 & & C_3 \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] \arrow[dr] & O \arrow[ul] \arrow[d, leftrightarrow] \\
LURD \arrow[r, leftrightarrow] & N_2 \arrow[ur] \arrow[r, leftrightarrow] & (S) \arrow[r, leftrightarrow] & h
\end{tikzcd}
\tag{8}
$$

Notice that from the vantage point of z in equivalence class h, the entire partition $\{L, U, R, D\}$ from fig. 7 has essentially shrunk to a single equivalence class $LURD$ and the entire RR-graph (4) has shrunk to a single vertex. Now, it is clear that the RR-graph from z (and associated partition) in (8) is simultaneously a refinement (of H) and a coarsening (of $\{L, U, R, D\}$), relative to the RR-graph (7) from x. While these partitions are incomparable (in the global lattice of recognizable sets), we can use a bundle structure to globally represent the natural refinement of perceptual equivalence classes that occurs as a neighborhood is entered, and the simultaneous coarsening (or identification) of classes that corresponds to moving away from a neighborhood. Mathematically speaking, we have a sensor map p_x that yields the perceptual equivalence classes in (7). Correspondingly, we have a sensor map p_z that yields the perceptual equivalence classes in (8). We consider varying the configuration ϕ from x to z along a path connecting the two points. This defines a family of local sensing maps $\{p_\phi\}$. Each map locally defines a partition of the world into perceptual equivalence classes (and an RR-graph); this "sequence" of RR-graphs "converges" to (8) as $\phi \to z$ and $p_\phi \to p_z$. We wish to paste together these local maps $\{p_\phi\}$ to get a "global" sensing map, which will be a bundle. The bundle, when suitably restricted, will yield the local quotient maps like p_x with which we are by now familiar.

We proceed as follows. If z is "far" from x, the sensing map p_x partitions the neighborhood about x into a (relatively) fine partition, and the neighborhood near z into a (relatively) coarse partition, as above. Presumably, if we consider a path from x to z the fine partition of the neighborhood near x gradually gets coarser, while the coarse partition of the neighborhood near z gradually gets finer. We may model this by globalizing the sensing map, and thereby endowing it with a "bundle" structure. Now, technically a bundle would require that the gradual change in fineness described above satisfy some smoothness condition. This may be too much to hope for in our case: for example, consider the sudden rounding of a corner, in which an entire novel diorama is revealed to the robot. Depending on the situation, it may be more convenient to model this as a kind of singularity (rapid change) in information, rather than as a smooth process in which, as the turn is made, more and more of the new scene is smoothly revealed. However, we can still use a *bundle with singularities* (in this case) in our model; we'll use the term "bundle" here to cover both situations.

4.6.1 Constructing the Bundle

To construct the bundle, we first let M be a space (eg, a manifold with boundary) denoting the world, and C be the configuration space of the robot. Let B be our sensor space as above (sec. 4.1). C can contain parameters that control sensing as well as motion; for example, C can contain as a factor the circle S^1 which controls the angle θ at which a range-finder is pointed.

For a fixed configuration ϕ, we have a sensing map such as p_ϕ described in some detail above (for example, p_ϕ is like p_x or p_x^* in (2)). We "globalize" p_ϕ as follows:

Definition 4.2 *The* world sensing bundle p *is given by*

$$
\begin{array}{rcccl}
p & : & M \times C & \to & B \\
 & & (y, \phi) & \mapsto & p_\phi(y).
\end{array}
\tag{9}
$$

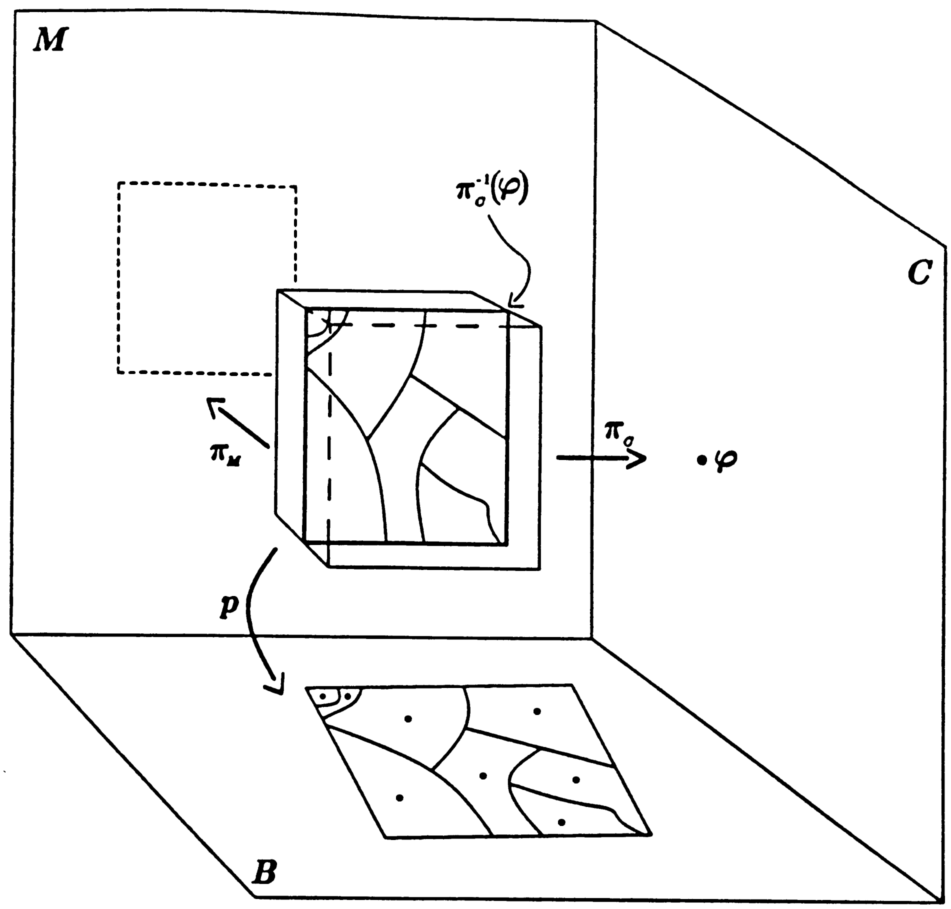

Figure 10: The bundle p and the projections π_M and π_C, drawn conceptually. For ϕ in C, the fibre $\pi_C^{-1}(\phi)$ is shown. It is broken into perceptual equivalence classes by the sensing map p_ϕ, which is a quotient on this fibre.

Now, let π_C be the natural projection of $M \times C$ onto C, so $\pi_C(y, \phi) = \phi$. Hence both p and π_C are fibrations. We construct the diagram

$$\begin{array}{ccc} & M \times C & \\ \swarrow_{\pi_C} & & \searrow_{p} \\ C & & B. \end{array} \tag{10}$$

It is not hard to see the following property, which yields the desired local quotient structure we promised:

Proposition 4.3 *The map p is locally a quotient map; i.e., it is a quotient on each fibre $\pi_C^{-1}(\phi)$ of π_C.*

These fibre maps are sketched conceptually in figs. 10, and 11.

When we have a robot fixed at one configuration, the definition of perceptual equivalence classes we adopted in 4.1 is sensible. Now, we wish somehow to generalize the kind of definition in 4.1 to the case of the bundle p. We can do this by passing to the configuration space C of the robot, and define an equivalence relation on this space that defines what it means for two places to "look the same." This essentially means that to construct global representations of perceptual equivalence classes, we pass to "robo-centric" coordinates (configuration space).

To make sense of this definition, it helps to bear in mind the following:

1. The configuration space C includes the control parameters of the sensors (e.g. aim of the rangefinder), and knowledge of these parameter values is encoded in the sensor space B.

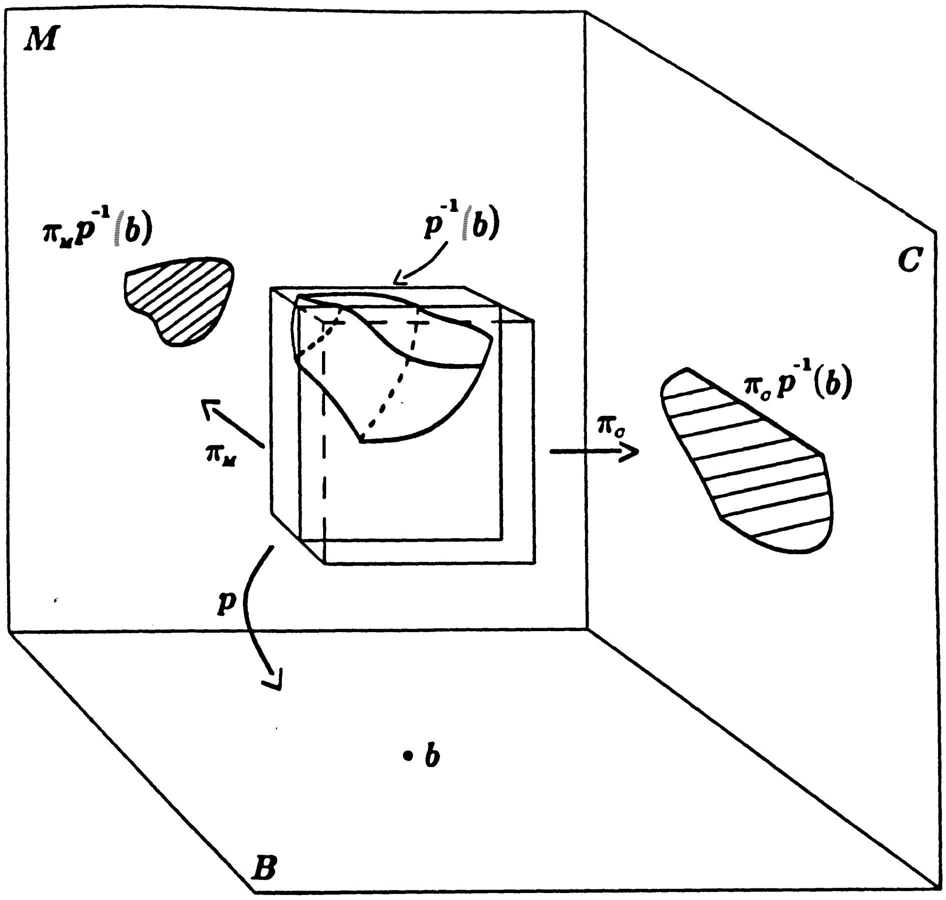

Figure 11: The configuration space equivalence classes are generated by taking fibres of the form $p^{-1}(b)$ for $b \in B$, and projecting them onto C. Hence the configuration space perceptual equivalence classes are the cells in the arrangement generated by the sets $\{ \pi_C p^{-1}(b) \}$ for $b \in B$.

---

2. Sensor noise is modelled by passing to set-valued functions. That is, we consider $p_\phi(y)$ to be a subset of B, representing a bound on all possible sensor values. In general, this can destroy the transitivity of, for example, def. 4.4 (below), but the reflexivity and symmetry are retained. Hence a definition like 4.4 becomes analogous to the "adjacency" relation, and its transitive closure analogous to "connectedness". This issue is subtle. Even though noise *can* destroy transitivity, we still use the term *perceptual equivalence class* to illuminate connections with previous work [MC, WB, S, CMM], which uses the term *sensory equivalence*. While these ideas cover the basic approach to dealing with noise and uncertainty, there is much more to be said; see the appendix C.

Definition 4.4 *Fix a sensor value $b \in B$. We call a subset S of configuration space C a* configuration space perceptual equivalence[15] class *(or simply* C-Perceptual Equivalence Class*) when for each configuration ϕ in S, there exists some point y_ϕ in the world M such that $p_\phi(y_\phi) = b$.*

If we imagine the robot living in configuration space C, the places relevant to the robot will be places in that configuration space. This definition tells us what it means for two places to appear the same; it defines what it means to be a perceptual equivalence class in configuration space.

[15] But see point 2, above, and the appendix C, below.

4.7 The Structure of Configuration Space Perceptual Equivalence Classes

It is somewhat difficult to imagine what the C-perceptual equivalence classes look like. We now work a simple example to illustrate the C-perceptual equivalence classes. We must point out that noise and dihedral reflections (for instance) would considerably alter and complicate this example.[16] However, the point of the example is to illustrate the definition, not to guide the robot. In the presence of noise, certainly *different* C-perceptual equivalence classes would occur and by constructing their reachability graph, the robot could make a map of landmarks in the world, and how to navigate between them. That is, the C-Perceptual equivalence classes form the (globally speaking) recognizable sets in configuration space. We define a reachability graph whose vertices are the C-perceptual equivalence classes, and whose edges are motions or motion strategies. The presence of a directed edge labeled (say) v indicates that motion strategy v moves from one recognizable set to another. This graph—the reachability graph of C-Perceptual equivalence classes—is the global analogue of the RR-graph discussed above. It is like a map of landmarks in configuration space the robot can construct and use to navigate. The landmarks are naturally defined using the perceptual equivalence relation.

We will develop two partitions of a configuration space C, into a fine partition $\mathcal{J}$ and a coarse partition $\mathcal{K}$. The lattice relationship that exists among partitions of C, together with inclusion maps and forgetful maps, is the generalization *mutatis mutandis* of the lattice of recognizable sets discussed above in sec. 4.4.

4.7.1 A Fine Partition $\mathcal{J}$ of Configuration Space

We suppose a point robot (or that we have passed into configuration space and grown the obstacles appropriately; see [LoP]). Suppose we have a rangefinder as described in 4.1. Suppose the the rangefinder can only sense distances less than m. Let Z_m denote the ring of integers (mod m). If a point out of range is interrogated, the sensor returns the value o. We assume the distance values sensed lie in the set

$$B_0 = Z_m \cup \{\emptyset, \infty, \circ\} \tag{11}$$

as described in sec. 4.1.[17] For convenience, we will assume that the rangefinder is not servoable, but that it sweeps around and we can approximately sense its orientation θ. Specifically, the orientation values lie in a quotient of the orientation space S^1 (a circle) into 4 quadrants: Hence, S^1 is partitioned into

$$S^1/4 = \{N, S, E, W\}$$

and the sensor space B is the product

$$B = B_0 \times S^1/4.$$

Alternatively, we could assume we have 4 broad-angle rangefinders pointed at the four points of the compass.

Assume the world M and the configuration space C are both the cartesian plane $\Re^2$. The local sensing functions have the form $p_\phi : M \to B$ and paste together to form a global world sensing bundle p as in def. 4.2.

Now, we assume the robot is in a rectangular room of width and breadth much greater than m, and that the walls of the room are fortuitously lined up North to South. The C-perceptual equivalence classes are shown in fig. 12.

Notice the following about fig. 12:

[16]For an example of realistic perceptual equivalence classes that can be recognized by a state-of-the-art mobot, see, for example, [Mat].

[17]Techincally, B_0 should also include the value $\perp$.

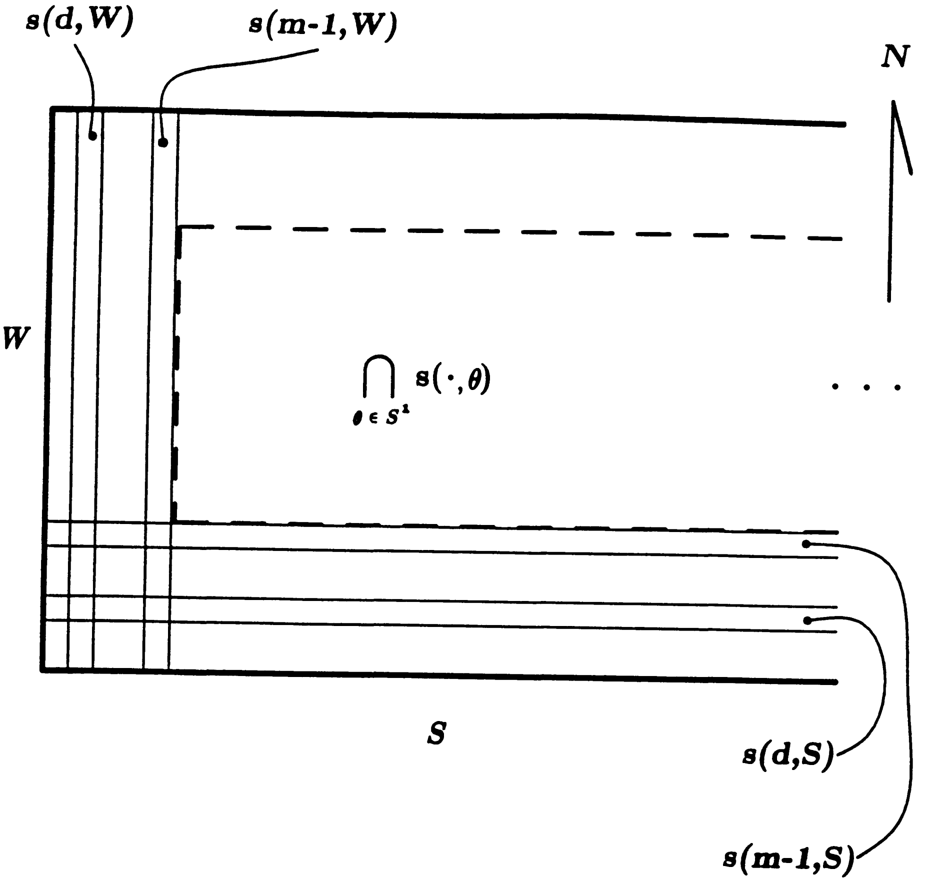

Figure 12: Example: the map of configuration space perceptual equivalence classes for a robot with a range finder, in a rectangular room. This shows the fine partition $\mathcal{J}$. Now, Let (d, θ) be a typical sensed value in $B = \left(Z_m \cup \{\emptyset, \infty, \circ\}\right) \times S^1/4$. Then the generating sets of the C-perceptual equivalence classes $s(d, \theta)$ are of the form $s(d, \theta) = \pi_C p^{-1}\left((d, \theta)\right)$. The cells in the arrangement formed by these sets form the partition of configuration space desired. We write $\theta \in \{N, S, E, W\}$ to denote angles in $S^1/4$. Hence, south strips have the form $s(d, S)$ and west strips are $s(d, W)$.

1. In the middle of the room is a blank rectangle from which no wall can be sensed. This area is further than distance m from any wall. From each of these configurations, no sensor values are possible except for $\circ$ and $\emptyset$.

2. Parallel to the south wall is a strip of points of distance $[d, d+1]$ from the wall, for each $d \in Z_m$. From each of these points, the south wall appears the same. Call these south-strips.

3. There are similar strips parallel to the North, East, and West walls. Call these North-strips, etc.

4. Near the south-west corner, (specifically, at distance less than $m-1$ from the two walls), we fall in the intersection of a south-strip and a west-strip. This means that using the sensors, we are able to localize position much better. The perceptual equivalence classes are much smaller at the corners. These perceptual equivalence classes are the intersections of the south- and west-strips.

Hence, the C-perceptual equivalence classes are formed as follows: take the generating sets of type (1)-(3) as shown above. Form the arrangement of these generating sets. The cells in this arrangements are the C-perceptual equivalence classes.

4.7.2 Classification of the C-Perceptual Equivalence Classes

We wish to formalize the construction of the C-perceptual equivalence classes, to obtain a way to generate and classify them in general. This construction generalizes the example in the previous section 4.7.

Lemma 4.5 *The maximal C-Perceptual Equivalence Classes under definition 4.4 are the cells in the arrangement*[18] *generated by the sets*

$$\{\pi_C p^{-1}(b)\}$$

for $b \in B$.

This result is completely general, and does not apply merely to our current example. We will later discuss how this classification lemma leads to an algorithm for computing the recognizable sets in configuration space.

4.7.3 A Coarse Partition $\mathcal{K}$ of Configuration Space

By analogy with section 4.1, by ignoring (or not being able to sense) the orientation θ of the rangefinder, we should obtain a coarser partition of C into "bigger" C-perceptual equivalence classes. We now see why this is so.

Suppose we can not sense the orientation θ of the distance value, i.e., that $B = B_0$ as discussed in sec. 4.1. We would expect the C-Perceptual equivalence classes in this case to be "bigger", and to form a less fine partition of C than $\mathcal{J}$ in fig. 12. Indeed, this expectation is born out. In the case where we cannot sense or know the orientation θ of the distance value, the C-perceptual equivalence classes are given as follows (see fig. 12). First, we develop the following notation.

Definition 4.6 *Let*

$$s(d,\theta) = \pi_C p^{-1}((d,\theta))$$

for $d \in B$ *and* $\theta \in S^1/4$. *We call* $s(d,\theta)$ *a "d-strip."*

Here is the coarse partition $\mathcal{K}$:

1. The class denoted "the neutral zone" O,

$$O = \bigcap_\theta s(\circ,\theta)$$

   is the same.

2. For each $d \in Z_m$, the following classes are identified into one single perceptual equivalence class s_d, where

$$s_d = \bigcup_\theta s(d,\theta). \tag{12}$$

[18] See construction A.2 in the appendix

(The union and intersection are over $\theta \in \{N, S, E, W\}$). Hence, for example, a south-strip at distance d is indistinguishable from a west-strip at distance d, etc.

Now, in the same way that we defined a lattice of recognizable sets in M (for a fixed configuration), we can define a similar lattice of recognizable sets in C. $\mathcal{J}$ is finer than $\mathcal{K}$, and there exists an inclusion map $\jmath : \mathcal{K} \hookrightarrow \mathcal{J}$ and a forgetful map $\mathcal{J} \rightarrow \mathcal{K}$. Claim A.3 generalizes straightforwardly, and we can think of the lattice of recognizable sets in configuration space in the same way that we imagined the lattice in sec. 4.4. The partition $\{\phi\}_{\phi \in C}$ of C into all its one point sets corresponds to perfect sensing (any two points are distinguishable). Similarly, the partition of C into just one class $\{C\}$ corresponds to *no* sensing (all the world looks the same); this is the least fine partition.

4.7.4 An Intermediate Partition $\mathcal{K}'$ of Configuration Space

We now consider a partition $\mathcal{K}'$ that is finer than $\mathcal{K}$ but less fine than $\mathcal{J}$:

$$\mathcal{K} \hookrightarrow \mathcal{K}' \hookrightarrow \mathcal{J}.$$

This partition is formed by the following sensor assumptions. We assume that our robot cannot sense or infer θ, and hence it cannot distinguish (say) a south-strip from a west-strip if both are at distance d. Therefore, in $\mathcal{K}'$, these strips are identified, as in sec. 4.7.3. Suppose, however, the robot *can* tell when it is in the *intersection* of a south-strip and a west-strip: for example, it could infer this by receiving two distinct rangefinder readings.

Call a South strip "pure" where it does not intersect a West or East strip. The "pure" south strips, we denote by $s^*(d, S)$. See fig. 13. For example,

$$s^*(d, S) = s(d, S) - s(Z_m, \{E, W\})$$

. We similarly define the "pure" north, east, and west strips. In $\mathcal{K}'$, any two pure d-strips are indistinguishable.

Hence the C-Perceptual equivalence classes in $\mathcal{K}'$ are

1. The neutral zone O,

2. All pure d-strips form one equivalence class, which we denote s_d^* (see fig. 13):

$$s_d^* = \bigcup_{\theta} s^*(d, \theta).$$

3. The corner intersection classes of the form (eg) $s(d, S) \cap s(n, W)$ for $d, n \in Z_m$. See fig. 13.

The following construction is somewhat technical, but we provide it for completeness. The reader may skip it at first reading:

Construction 4.7 *In $\mathcal{J}$, a corner intersection class such as $s(d, S) \cap s(n, W)$ is determined by its "distances" d and n and its "directions" S and W. Now consider $\mathcal{K}'$: in a corner intersection the directions are irrelevant (since we can't sense them). We can tell it's a corner (because, eg, we get two distinct wide-angle returns from the rangefinder) but we don't know the direction of the returns. We must therefore identify any two corner intersection classes with the same "distances" but possibly different "directions".*

Hence, the intersection classes are in bijective correspondence with the (unordered) pairs of distances (d, n). For example, consider two corner intersections $s(d, S) \cap s(n, W)$ and $s(d', N) \cap s(n', E)$, with $d \leq n$ and $d' \leq n'$. They are identified (in the same equivalence class) if and only if $d = d'$ and $n = n'$. These "folded" corner classes are denoted $f(d, n)$ for $d \leq n$:

$$f(d, n) = \bigcup_{\substack{\theta_1, \theta_2 \in S^1/4 \\ \theta_1 \neq \theta_2}} s(d, \theta_1) \cap s(n, \theta_2) \tag{13}$$

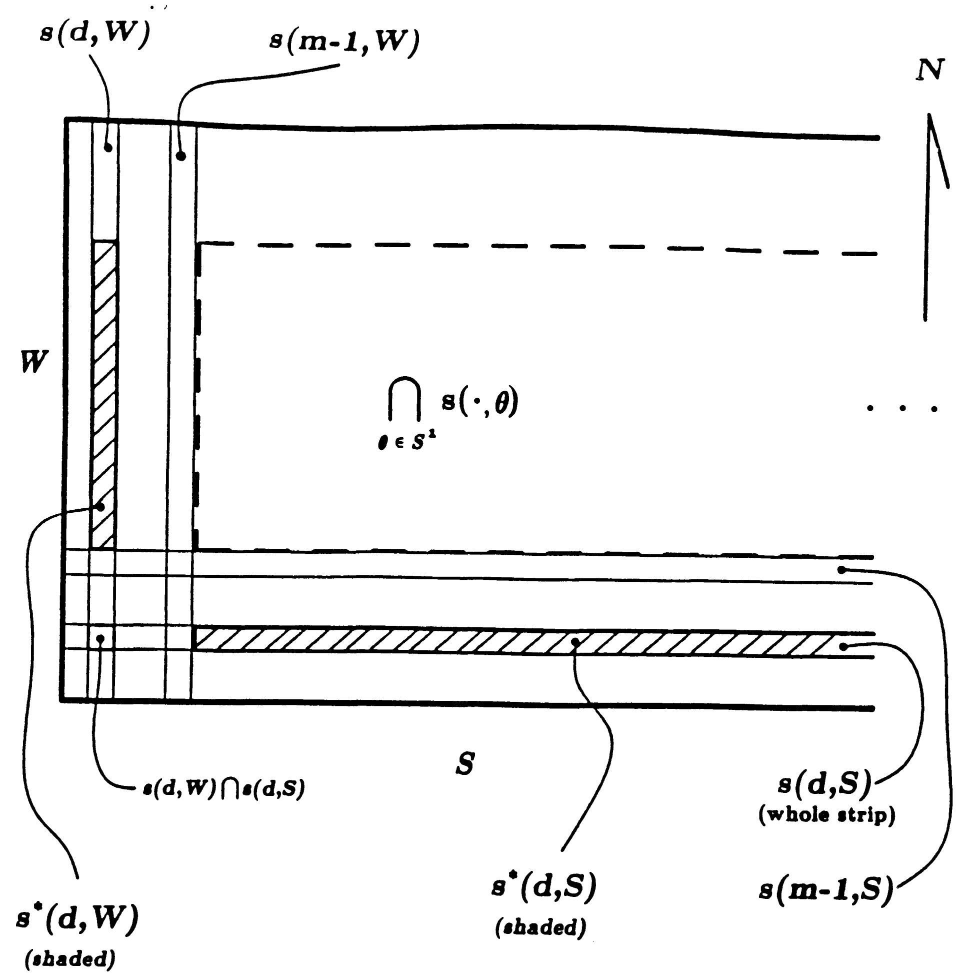

Figure 13: The "pure" south strips $s^*(d,S)$ and west pure west strips $s^*(d,S)$ are shown. A corner intersection class is also shown. It is $s(d,S) \cap s(d,W)$ in the southwest corner.

4.8 History is Important

4.8.1 A General Discussion of History

In mobile robotics, many algorithms, planners and executions systems have employed *history* to reason about geometry, goals, tasks, and uncertainty. For example, sensor history can be used as a kind of odometry. If we have bounds on control error, then control history can be used as a kind of dead reckoning, by integrating the controls from an estimate of the initial conditions. The two can be combined to give a good estimate of the state of the robot and the environment. Erdmann [Erd] gave an elegant characterization of history in his work on pre-images. In highly unstructureed environments, the precise computational role of history, and the information gained by using history, is often hard to quantify. In this section, we explore the use of history, and use our framework to illuminate the key issues.

We use the term *sensor history* to denote a mapping from time to sensor values, or any "subset" of this function, that the robot remembers. Similarly, we use the term *control history* for the mapping from time to commanded controls. Both can be immensely useful in robot strategies [Erd]. Consider an robot that must follow walls, in a typical office building. Suppose it can distinguish corners from straight hallways, but cannot tell, say, a north from a west wall. If the robot starts near wall L (e.g. as in fig. 7) and makes 1 or 2 turns (following walls), it can know that it is has not reëncountered L, even though the nearest wall now (say, U) is *locally* perceptually indistinguishable from L. This reasoning is accomplished using sensing and control history, represented by a reachability map (4.4.4). Effectively, history permits us to ascend to a finer partition, in which U and L are distinguishable.

As another example, consider fig. 8, illustrating a partition $\mathcal{V}$ and a finer partition $\mathcal{U}$ related by an inclusion map $\jmath$, which models the *a priori* availability of better sensing information (namely, orientation). Assume that sensing θ is impossible. This places us in the coarse partition $\mathcal{V}$. Now suppose control history is available. For example, coarse control history $v^*(t)$ might approximately "remember" the commanded motion direction $v(t) \pm 45°$. Our inability to sense θ makes the inclusion map $\jmath$ unavailable directly; hence, the finer partition $\mathcal{U}$ is apparently unimaginable. However, our *a priori* knowledge of $\mathcal{V}$ plus control history $v^*(t)$ permits us to "lift" to the partition $\mathcal{U}$ "manually"—that is to construct an inclusion map like $\jmath$ from the evidence $(\mathcal{V}, v^*(t))$. See the appendix for details of the construction. Hence, we can pass to a finer partition $\mathcal{U}$ and reason about sensing and action there. This example illustrates a quite general phenomenon. There is much more to be said about the "lifting" ability history admits; for a mathematical formalization see the appendix.

4.8.2 History's Effect On Partition Fineness

History can have an interesting effect on the "coarsening" obtained in passing from a fine partition $\mathcal{J}$ to a less fine partition $\mathcal{K}'$—this process is described in sec. 4.7.3, 4.7.4 above. Let us examine how the "identification" of "small" C-perceptual equivalence classes in $\mathcal{J}$ yields larger classes in $\mathcal{K}'$. The key point is that the identification described above (sec. 4.7.3, 4.7.4) is correct in the case of no history. However, with history, the situation is more complicated. This is because (in our example), it may be possible to obtain a global estimate of orientation θ by correctly interpreting the history.

We recall that the *reachability* relationship between C-perceptual equivalence classes in a single partition defines the "connectivity" of the quotient space. This "connectivity" is the analogue of path-connectivity. The connectivity relationship defines a graph (which we will continue to call the RR-graph) whose vertices are the C-perceptual equivalence classes in a given partition of C, and whose edges are motion strategies. (We should distinguish that this graph is "in" C rather than M, but that is the only difference). The exact semantics of these arcs depends on the application. For example, if we require guaranteed strategies, then the arcs should represent motion strategies that are guaranteed to succeed. If we want EDR strategies [Don89], they should represent EDR plans. These kind of reachability arcs we call *strong*, in that they come with guarantees.

On the other hand, if we are simply recording the experience of the robot as it moves—making a "map" of the C-perceptual equivalence classes—then the arrows simply denote which motion strategy was used to move from one recognizable set to another. We call this last kind of arc **weak reachability**. For $U, V \subset C$, a weak reachability arc $U \xrightarrow{v} V$ denotes the fact that the robot used motion strategy v and recognizably achieved V having started in U.[19]

It is certainly true that path-connectivity is a necessary condition for (weak) reachability. Hence, if two equivalence classes are disconnected, then one is not reachable from another. The key point is that, in our example, pure south-strips and pure west-strips are disconnected: one must first pass through the neutral zone O or through a class that is the intersection of a south-strip and a west-strip.[20] That is, (part of) the RR-graph for $\mathcal{J}$ (fig. 12) looks like this (let $m_i = m - i$):

[19] Weak reachability may also be viewed as follows: From the planning point of view, we may write $U \xrightarrow{v} V$ if it is *possible* to achieve V from U by doing v. Hence we view deciding this relationship without executing v as a form of planning.

[20] Actually, the classes $s(d, W)$ and $s(d, S)$ are connected by a "diagonal" path when $d = m - 1$, but only for this value of d. Furthermore, the intersection of their closures (when $d = m - 1$) is a single point, and so the connectivity is highly singular.

$$
\begin{array}{ccccc}
s^*(m_2, W) & \longleftrightarrow & s^*(m_1, W) & \longleftrightarrow & \bigcirc \\
\updownarrow & & \updownarrow & & \updownarrow \\
s(m_2, W) \cap s(m_1, S) & \longleftrightarrow & s(m_1, W) \cap s(m_1, S) & \longleftrightarrow & s^*(m_1, S) \\
\updownarrow & & \updownarrow & & \updownarrow \\
s(m_2, W) \cap s(m_2, S) & \longleftrightarrow & s(m_1, W) \cap s(m_2, S) & \longleftrightarrow & s^*(m_2, S) \\
\updownarrow & & \updownarrow & & \updownarrow \\
s(m_2, W) \cap s(m_3, S) & \longleftrightarrow & s(m_1, W) \cap s(m_3, S) & \longleftrightarrow & s^*(m_3, S) \\
\updownarrow & & \updownarrow & & \updownarrow \\
\vdots & & \vdots & & \vdots
\end{array}
\tag{14}
$$

(In (14), we have omitted the diagonal connections for clarity).
We propose the following:

- It is only reasonable to identify C-perceptual equivalence classes when they are connected by a (weak) reachability arc.
- Two classes that are not path-connected cannot be connected by a (weak) reachability arc.

We now justify this idea. Again, the basic point is that history can be used to give the robot a global estimate of position, hence interdifferentiating the south-strips and the west-strips. That is: if we identify pure south strips and pure west strips, then a motion that moves from a south strip $s(d, S)$ to a west strip $s(d, W)$ may *appear* to be reentering the same equivalence class s^*_d. However, with history, the robot can know that a corner equivalence class was traversed in between. This sensor history, plus control history, allows the robot to distinguish between the south strip and the west strip. We now make this point precise.

4.8.3 Example with History

In this example, we illustrate how control history plus sensing history can be used to gain information, and how this in turns enables us to plan in a less coarse partition. The sensing history required is: a history of the equivalence classes in $\mathcal{K}'$ that have been traversed. This may be modelled as a mapping from time to $\mathcal{K}'$. The control history may be viewed as (an approximate) mapping from time to the control actuation of the robot. For this example, it suffices that we suppose the robot is velocity controlled, and that we have a rough measure of the time-history of the commanded velocity. We will assume that a mapping from time to $S^1/4$ is available.

We justify this assumption as follows. Assume we have a robot that can do rather crude incremental control with bounded error. This permits a kind of primitive odometry. In particular, we assume we can command a velocity of the robot for some very short time period. The error in velocity is bounded (see below). After the time period is up, we can sense and move again.[21]

Consider fig. 14. Suppose the robot starts at configuration x in pure south-strip $s(d, S)$, where $d \neq m - 1$. The robot executes the following plan.

Motion Strategy v

v_1 Move west, staying distance d from the wall on the "left," until an obstacle straight ahead is detected. Stop at distance d from the object dead ahead.

v_2 Turn North. (90° right turn). Move ahead, staying distance d from the wall on the "left."

[21] These are rather weak assumptions.

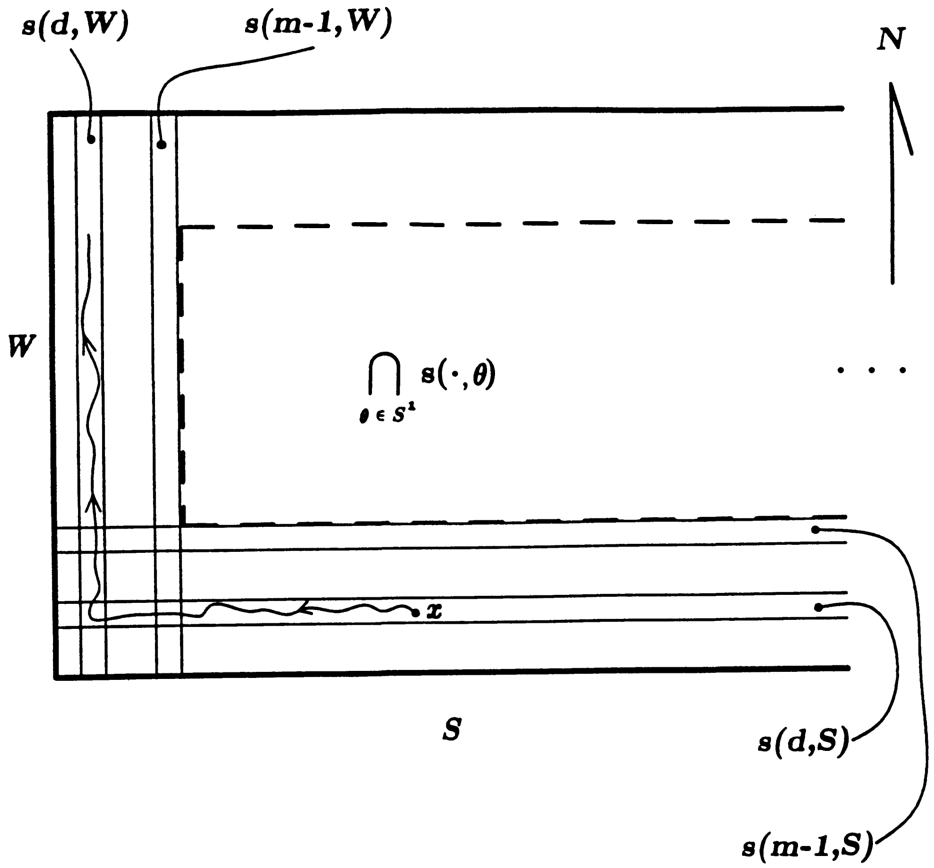

Figure 14: The robot starts at configuration x. It is commanded to move, while remaining in C-perceptual equivalence class $s(d,S)$. When $s(d,S) \cap s(d,W)$ is reached, the robot is commanded to turn 90° right. Next, the robot is commanded to move, while remaining in $s(d,W)$. When $s(m-1,N)$ is reached, the motion halts.

v_3 Stop when until an obstacle straight ahead is detected.

This plan uses orientation in sensing (and control). However, the plan may be expressed entirely without orientation in sensing (for example, without notions such as *left*) by using commands like "Move, while maintaining one rangefinder reading at distance d." This effectively means: ***Move in a straight line, while remaining within C-perceptual equivalence class*** s_d.[22] One may do this—remove all references to orientation in sensing—by expressing the motion steps using invariants. For example: the step ***Move in a straight line while maintaining sensor readings*** $s(d,S)$ corresponds exactly to wall-following, and is closely related to following a heading. Note that we still need orientation in control: the controls, and control history (Time $\rightarrow S^1/4$) do affect the connectivity (reachability) among the equivalence classes in the partition, and the resulting RR-graph, as we shall see.

Here is how the plan would be specified using C-perceptual equivalence classes. All that is required to succeed is (first) that the robot's progress be monotonic (i.e., it cannot reverse direction accidentally while remaining in one C-perceptual equivalence class). That is, the robot's global error in heading must be less than $\pm 90°$. Second, note that this plan will work so long as the robot can "turn right"—that is, so long as the robot's global error in heading is less than $\pm 45°$. The latter is a tighter constraint. Note that recognizability is automatic because we have used the recognizable sets as our basic building blocks.

[22] s_d is defined in eq. (12).

Motion Strategy v for Partition $\mathcal{J}$

v_1 Move west, while remaining in C-perceptual equivalence class $s(d, S)$. Halt when $s(d, S) \cap s(d, W)$ is reached.

v_2 Turn right 90°. Move north, while remaining in $s(d, W)$.

v_3 Halt when $s(m-1, N)$ is reached.

Now, how would the plan look in partition $\mathcal{K}'$? The answer is as follows: Again, this plan will work so long as the robot can "turn right"—that is, so long as the robot's global error in heading is less than ±45°. Recall $f(\cdot, \cdot)$ is the "folded" equivalence class of corner intersection regions in def. 4.7 above:

Motion Strategy v for Partition $\mathcal{K}'$

We use s_d as a kind of shorthand, since even though s_d is a class in $\mathcal{K}$, we can build it by unioning $\mathcal{K}'$'s recognizable sets.

v_1 Move west, while remaining in C-perceptual equivalence class s_d. Halt when $f(d, d)$ is reached.

v_2 Turn right 90°. Move north, while remaining in s_d.

v_3 Halt when $f(d, m-1)$ is reached.

Obviously, these plans can only be executed approximately. One possible resulting path is shown in fig. 14. This path induces the following (weak) reachability edges in the RR-graphs of C-Perceptual equivalence classes $\mathcal{J}$, $\mathcal{K}'$, and $\mathcal{K}$ (recall $m_1 = m - 1$):

$$\begin{array}{llllllll} [\mathcal{J}] & s^*(d,S) & \xrightarrow{v_1} & s(d,S) \cap s(d,W) & \xrightarrow{v_2} & s^*(d,W) & \xrightarrow{v_3} & s(d,W) \cap s(m_1,N) \\ [\mathcal{K}'] & s_d^* & \xrightarrow{v_1} & f(d,d) & \xrightarrow{v_2} & s_d^* & \xrightarrow{v_3} & f(d,m_1) \\ [\mathcal{K}] & s_d & \xrightarrow{v_1} & s_d & \xrightarrow{v_2} & s_d & \xrightarrow{v_3} & s_d. \end{array} \tag{15}$$

We can view (15) as a partial sensing history for the motion strategy v, shown in different partitions of C.

Construction 4.8 *Actually the sensing history will be more detailed, because we can detect all the $f(d, n)$'s we cross. The actual sensing history in $\mathcal{K}'$ would be something like this*

$$\overbrace{s_d^*, f(d,m_1), f(d,m_2), \ldots, f(d,d+1), f(d,d),}^{v_1} \underbrace{f(d,d), f(d,d+1), \ldots, f(d,m_2), f(d,m_1), s_d^*,}_{v_2} \overbrace{f(d,m_1)}^{v_3}.$$

We use (15) as a kind of shorthand to denote important (for us) sets that the motions recognizably reach or cross.

The control history may be written as follows:

$$\begin{array}{lcl} v_1 & : & W \\ v_2 & : & N \\ v_3 & : & N. \end{array} \tag{16}$$

We suppose the robot lives in the partition $\mathcal{K}'$ as described above. Now, consider in particular the reachability maps for $\mathcal{K}'$:

$$[\mathcal{K}'] \quad s_d^* \xrightarrow{v_1} f(d,d) \xrightarrow{v_2} s_d^* \xrightarrow{v_3} f(d,m_1). \tag{17}$$

Recall the definition in eq. (12) of s_d. Now, according to (17) (and especially from the full history in construction 4.8 above), the robot leaves equivalence class s_d^* during motion v_1 and reenters it during v_2. However, it is clear that using control (16) and sensing (17) history, that v_2 enters a different component of s_d^* (in this case, $s(d,W)$) from the component exited during v_1 (that component was $s(d,S)$). This is because the controls (16) and initial conditions $f(d,d)$ are incompatible with reentry of the first component. Since the two components are hence separated by classes of type $f(\cdot,\cdot)$, we can distinguish them. That is, it is possible to tell one from another since we must pass through the intermediate $f(\cdot,\cdot)$ states to get between them. Hence, the history of sensors and controls permits us to pass to a finer partition $\mathcal{J}$. The passing from $\mathcal{K}'$ to $\mathcal{J}$ precisely quantifies the amount of information gained by using this sensor and control history. Thus we have described a fairly general way to exploit history of past sensor values and actuator controls, and shown how to use this to gain information about the environment and the robot's state. The information gain from history can allow us to lift to a finer partition of C-Perceptual equivalence classes, in which there are "more" recognizable sets that can be told apart. The difference in fineness in the two partitions gives a precise measure for the amount of information gained by using history.

Finally, we should point out that in this example, it was not necessary to preserve a continuous history sensor or control values, such as $b : T \to B$ (T denotes time). Instead, we only need a mapping of time into the coarse partition $\mathcal{K}'$, and a very coarse history of control. The sensor history mapping $b_* : T \to \mathcal{K}'$ such as (17) should be available since the sets in $\mathcal{K}'$ are the *a priori* recognizable sets, without history. That is, from the mappings

$$\begin{tikzcd} T \arrow[dr, "b"'] & C \times M \arrow[d, "p"] \\ & B \end{tikzcd} \tag{18}$$

all we need construct is the mapping b_*, which is essentially the pull-back of b by the bundle p. We elaborate on the construction of b_* in sec. 9.2.

The control history (16) should be available from a trace of the plan. In general, control history will be some mapping from time into a quotient of the tangent bundle.[23] The reason we view control history this way is: control history specifies (approximately) what velocity the system has at a given time. Velocities live in the tangent bundle. By integrating this control history, one obtains an approximate "dead reckoning" map $T \to C$.

Of course, the lifting required some knowledge of $\mathcal{J}$. It would be of great interest to determine whether a finer history than b_* was ever required to obtain a lifting to a finer partition. Second, we would like to know the minimal amount of information about the finer partition $\mathcal{J}$ required to construct a lifting. In the appendix, sec. D, we formalize the sense in which history permits "lifting" to a finer partition.

5 Building Virtual Sensors

Consider the general problem of how new sensing maps can be built out of old ones. We can show that the key constructive tools are integration (of sensor histories) and union (of equivalence classes), composed with procedural abstractions similar to `map` and `Accumulate` (see app. B). The motivating reasons for building new sensing maps out of old ones are as follows:

[23] Also called the phase-space.

1. It shows how the robot can use information we are confident it can obtain (eg, raw sensor data), to make sound inferences about the existence and geometry of objects in the world that are relevant to its task.

2. If the construction is algorithmic, then we can implement sophisticated sensors using simple ones plus some computation, and be assured that the construction is principled and has some predictive power.

In addition, there is hope we can define quite general "virtual sensors." The idea is as follows. Suppose we wish to specify a robot task in terms of landmarks (perceptual equivalence classes) recognizable to humans. Examples include: the hallways, doors, corners, the atrium, and Hopcroft's office. How can we know if these features in the environment are recognizable to the robot? The key issue here is that if a task is specified in terms of "landmarks" that are not recognizable sets to the robot, then the robot may not be able to achieve the goal—in this case we say the task specification is not *robust*. This notion of robustness is analogous to the stronger notion of a *guaranteed strategy* in the [LMT] framework. Thus we ask, "How can a task be robustly specified to a robot when the robot's perceptual equivalence classes and the human's perceptual equivalence classes are different?"

Next, given the robot's "concrete" on-board sensors (eg, sonar, vision, etc), there will exist landmarks (perceptual equivalence classes) that are recognizable to the robot. But what are they, and how can humans understand them? One possibility is to let the robot rove around and then "dump" its map of perceptual equivalence classes in some human-readable form. Then, the human programmer could correlate (match) these landmarks with the features in the environment with respect to which he can specify the task. This may be a difficult matching problem; we note that tools from model-based recognition in machine vision might be used here. In this vein, we pause to note that, throughout our discussion, the term "human programmer" could be replaced by "task-level planner"—in effect, an automatic programmer. We will say more about this later.

Another approach to robust task specification is as follows. The human defines "virtual sensors"—effectively "oracles" that can recognize the features in the environment (walls, etc) that are critical to task specification. Thus, these virtual sensors are defined to return "symbolic" information like *"There is a wall of length l at coordinates (θ, d)."* The programmer then specifies the task in terms of these recognizable places. Next, the programmer tells the robot how to build or approximate these "virtual sensors" using a combination of raw sensors, computation, and retained state. This is the process we term "building new sensing maps out of old ones." Corresponding to the available effective sensing maps are what we will call the *concrete* perceptual equivalence classes. Our job is to show how to build the *virtual* perceptual equivalence classes corresponding to a virtual sensor out of the concrete perceptual equivalence classes on hand.

We believe that this process is quite general: namely that a new sensing map can be constructed by integration and union (or `map` and `Accumulate`) of existing sensor maps. Roughly speaking, the process involves first passing to the perceptual equivalence classes for the existing sensor maps. Then these classes are unioned, and the union is "controlled" by the values of some computation or other sensing maps. This corresponds to constrained integration (boundary or contour integration) of the sensing "signals." As an example, see fig. 15 and sec. B.3 below. This line of inquiry is quite general: it concerns how we can combine differential information relationships using computation to build structures with more information. That is, our research asks what are the differential relationships our sensors measure in the world, and how, over time, can we aggregate them to gain information? We hope that our answer, while still partial, can shed some light on this key robotics problem.

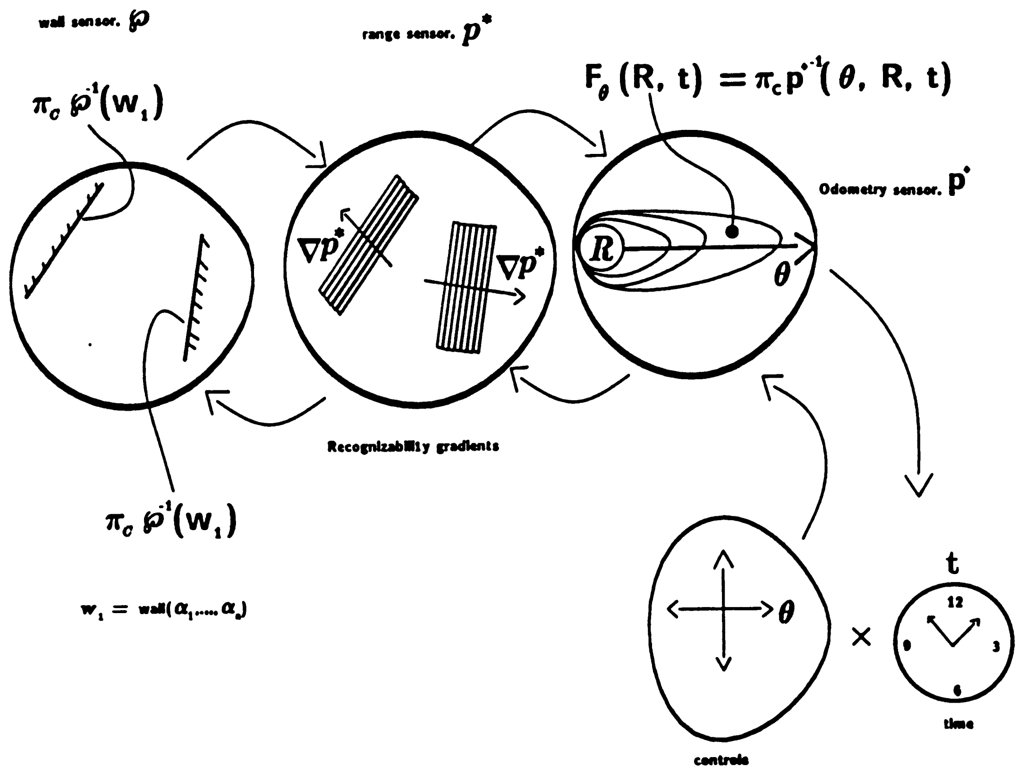

Figure 15: An illustration of how controls and time are integrated to build an odometry sensor p°. Odometry classes may be further integrated with range sensor data, and unioned to form a sensing map p^*, whose perceptual equivalence classes are strips parallel to walls. One may follow the recognizability gradient of these classes, and even define a virtual wall sensor $\wp$. Note that the odometry perceptual equivalence classes correspond to forward projections of the previous classes under the given robot controls and control uncertainties. See app. B and B.3.

6 Task-Directed Sensing and Planning

When virtual sensors can be defined constructively using concrete sensors, computation, and retained state, we enable a robot using these sensors to perform *task-directed* sensing. That is, if a task is specified in terms of the virtual sensors—more precisely, if the task is specified in terms of the "landmarks" recognizable using these sensors—then the robot may perform experiments to determine which "landmarks" it is near. This is because (i) the landmarks recognizable with the virtual sensors are virtual perceptual equivalence classes, (ii) the virtual perceptual equivalence classes are defined (for the robot) by a data structure that specifies how the concrete sensors should change, with respect to controls, over time. The notion of change here is captured in our framework using the *recognizability gradient*.

We must assume that the "lowest level" concrete perceptual equivalence classes may be reconstructed from sensor values using, for example, built-in sections of the associated sensing map (call this map p). That is, the robot must have some means of hypothesizing the existence and geometry of a perceptual equivalence class such as $\pi_C p^{-1}(b_i)$ given only a history of sensor values $\{b_i\}$. This capability may be encoded in the form of local sections[24] of the map p. But, once the robot can reconstruct *some* concrete perceptual equivalence classes, then it can begin composing

[24]Given a bundle $p : E \to B$ we call a map $s : B \to E$ a *section* of p if $p \circ s$ is the identity on B. A section is called "local" if it is only defined on a neighborhood U of B. Suppose we have a partial or incomplete model of the geometry of the world, and a partial or incomplete characterization of the sensor map p. In our view, a "small

these classes to form virtual classes. The definition of a virtual perceptual equivalence class (say, S, from appendix B), suggests, via the recognizability gradient, experiments (control changes and corresponding expected sensor readings) to identify S.

But once a virtual sensor p^* and its perceptual equivalence class are defined, we may then repeat the process, and in turn treat p^* as a *concrete* perceptual map. Using this concrete sensing map we may go on to construct another *virtual* sensing map $\wp$ and its perceptual equivalence classes, and so forth. The terms *concrete* and *virtual* are relative, and in fact, all sensing maps are partially ordered on constructibility. This order defines a hierarchy of virtual perceptual equivalence classes; from this point of view it seems clear that the job of an (offline) task-level planner is to make up a hierarchy of virtual perceptual equivalence classes. We contrast this with the job of an (online) "motion planner" whose job is

1. Using local sections, construct concrete perceptual equivalence classes.
2. Perform experiments to match the concrete perceptual equivalence classes against the virtual perceptual equivalence classes.
3. Plan (concrete) perceptual equivalence class-wise controls to accomplish (attain) the virtual perceptual equivalence classes required by the task.

7 Observations and Expectations

We now consider some key problems we must overcome. Our discussion here is considerably abreviated. Suppose we wish to design and program a mobile robot using our framework.[25] Then, we need to determine:

1. What initial state should the robot have?
2. How can the robot support multiple representations (eg, of perceptual equivalence classes from different sensors) of the same object?
3. What "program" is running on the robot? Is this just 1, above? (That is, is the initial state of the robot just the contents of its memory, both data and executable?)
4. How can we encode "expectations" about the world, in the form of 1 and 3?

These questions highlight the difference between *expectations* about the world and *observations* of the world. *Observations* are constructed solely from sensor and control history. *Expectations* are a mechanism for using initial state (1 and 3) plus history to match pre-existing data structures against the world, and constructing more state that will in turn be matched against future observations. Grasping the distinction between expectations and observations is crucial to understanding our framework. Compare the work of Cox, Durrant-Whyte, and Leonard [CL,LDWC], who make a similar distinction.

Let us illustrate why resolving these questions is somewhat tricky. Suppose we wish to write a robot *program* that uses or addresses entities in the world such as walls, doors, junctions, corridors, etc. If we pre-encode these features into the initial state (1 and 3, above), then the mobot will attempt to "match" the environment against these templates, and will hallucinate these features

number" of remembered sensor values and control values defines a kind of local section of the world sensing bundle $p: C \times M \rightarrow B$. It is possible that a "family" of these sections $\{ s_t \}$, parameterized by time, may be used to construct an approximation to the inverse map p^{-1}.

[25]This section owes much to discussions with Tomas Lozano-Pérez, Matt Mason, and Mike Erdmann, and we are grateful to them for their insight and suggestions.

even when they are inappropriate. For example, an office robot in the open desert might still build up a "map" with walls, corridors, etc.

First of all, this might not be all bad—for example, an arroyo is a bit like a corridor, and a cliff like a wall. But let us address the undesirable characteristics of matching the world's projection through our sensors, against pre-encoded expectations. How, then, shall we reconcile the two fundamentally different data structures:

A. *"Maps", constructed by a program, that consist of "expected" features matched against observations from the world.*

VERSUS

B. *RR-graphs of perceptual equivalence classes, and Lattices of RR-graphs.*

Clearly, both are useful: for example, (A) has addressable entities such as walls, corridors, etc, that can be used a "primitives" in pre-existing robot programs. However, these "features" crucially depend on the assumptions, encoded in the form of 1 and 3, above. This dependence makes the entities (eg landmarks) in (A) fragile. On the other hand, how can a human programmer know what the entities (the recognizable sets) in (B) look like? For example, we'd like to be able to tell the mobot, *Go down the hall and to the left* without first "microcoding" this high-level instruction in terms of concrete perceptual equivalence classes.

7.1 Multiple Representations

When asked how to reconcile (A) and (B), our answer, in a sense is, "why compromise?" We propose to support both representations. The idea is to keep both representations around and to use a version-space like technique [Mit] to maintain the data structures with respect to current and past observations. Structures like (A) would be "higher" in the lattice, since they are more "optimistic" —they rely on the accuracy of the pre-encoded expectations, and the accuracy of the matching algorithms. If correct, however, they also contain more information, since they encode not only sensor and control history, but also a correlation of that history with landmarks in the environment. On the other hand, structures like (B) are "safe" or "conservative"—they are based only on the fundamental laws of the calculus of the observable. While conservative, they contain less information since they do not match the history to the world. In between (A) and (B), we can imagine various other possible worlds, which form "partial" matches of RR-graphs to expectations. Indeed, one can imagine a partial order of such interpretations. The order is induced by the size of the domain of the matching function, and a measure of its monomorphism (i.e., how "one to one" it is).

Each entry in the lattice is a partition of the world, together with connectivity information. In general, the entries in the lattice correspond to possible worlds, and they are ordered from most optimistic (most information) to least. That is, instead of maintaining just one interpretation, we propose to keep a set, or *range* of interpretations, from safe to optimistic. This set is represented by an intermediate range—(a connected subgraph) in the lattice—of partitions. The top of the lattice is a complete, global world model. The connectivity information at the top corresponds to the ability to perform predictions (eg, physical simulation), with perfect accuracy. The bottom represents knowing nothing about the world. In between is a blob of feasible interpretations. As new observations are made, the mobot prunes interpretations that are inconsistent. For example, our feasible set of interpretations might contain *both* an RR-graph based solely on the history and concrete perceptual equivalence classes experienced by the robot, and *also* a "match" (or several possible matches) of this history to expected landmarks like walls, doorways, and corridors. We wish to build a system that functions as follows: In the office environment, over time, the evidence for the second interpretation (walls, etc.) would increase. In the desert, however, the weight of observation would eventually cause this interpretation to appear more unlikely, and eventually it would be pruned altogether.

To develop such a system, we must be able to show how to encode expectations about the world as perceptual equivalence class-like structures within our framework. We propose that *virtual sensors* provide such an encoding.

7.2 Encoding Expectations

To encode expectations, we define a set of *virtual perceptual equivalence classes.* When matched against observations, these encode "symbolic" information about the world, for example, *"There is a wall of length l at coordinates (θ, d)."* The precise definition is somewhat complex, but we can imagine that it consists of two parts: a "symbolic" predicate like `wall`, and a set of parameters, for example, (θ, l, d) in this case. Examples of predicates for an office robot might reference features in the world such as walls, corridors, or junctures; we may write them as, for example, `wall`, `corridor`, `juncture`, etc. The parameters for these predicates would specify the size, orientation, and so forth, of the feature in the world. Hence, we can imagine that the robot matches its observations against a set of templates (`wall`, etc) and instantiates the parameters of the template in the match. To make this notion precise we also need an *interpretation function.* In logic frameworks for AI, the interpretation function maps these predicates to possible worlds containing the features. Compare, for example, [R]. One may view our framework as specifying a geometric interpretation function. When discussing a general predicate we will imagine it has the general form `wall`(θ, l, d). We'll write a typical predicate like this: `wall`$(\alpha_1, \ldots, \alpha_n)$. So `wall` is the predicate and $(\alpha_1, \ldots, \alpha_n)$ are the parameters. We say that `wall`$(\alpha_1, \ldots, \alpha_n)$ *defines* a (parameterized) virtual perceptual equivalence class; the precise relationship will become clear below.

Our thesis is that the parameterized virtual perceptual equivalence classes encode local sections of the world sensing bundle $p : C \times M \to B$. Consider the following constructions:

History. We have *sensor history*, which may be represented by a map b^* from time T to sensed values B; hence $b^* : T \to B$. We have control history, which we might represent (for example) as a map v^* from time to commanded controls (eg, commanded velocity): $v^* : T \to TC$.

Local Sections. Given a bundle $p : E \to B$ we call a map $s : B \to E$ a *section* of p if $p \circ s$ is the identity on B. A section $s_t : U_t \to C \times M$ is called "local" if it is only defined on a neighborhood U_t of B. If there is a local section s_t for each time t in some interval of T, then we have a parametric family of local sections.

Expectations and Inference. History can define a family of local sections. The map s_t encodes our hypotheses or expectations about what features in the world correspond to the "current" sensor values.

Virtual Perceptual Equivalence Classes. The virtual perceptual equivalence classes encode local sections of p. See fig. 16.

To see the last point, consider the predicate `wall`$(\alpha_1, \ldots, \alpha_n)$ (here, we don't care about the semantics of the parameters $(\alpha_1, \ldots, \alpha_n)$). Here is how we define the virtual perceptual equivalence classes associated with `wall`. For discussion, we fix a set of parameters $(\alpha_1, \ldots, \alpha_n)$ for `wall`. Now, the information encoded in the predicate `wall`$(\alpha_1, \ldots, \alpha_n)$ is, essentially, *"There is a wall with parameters $(\alpha_1, \ldots, \alpha_n)$."* Now, we imagine the robot as comprising some collection of communicating subsystems. If one subsystem passes another a "message" saying `wall`$(\alpha_1, \ldots, \alpha_n)$, we can imagine that this message encodes either the "absolute" information *"There is a wall with parameters $(\alpha_1, \ldots, \alpha_n)$,"* or, if it is from a sensing subsystem, we may say (somewhat anthropomorphically) it connotes the sensory information *"I see a wall with parameters $(\alpha_1, \ldots, \alpha_n)$."* (This difference highlights the need for an "interpretation function" in AI logical frameworks for sensory systems).

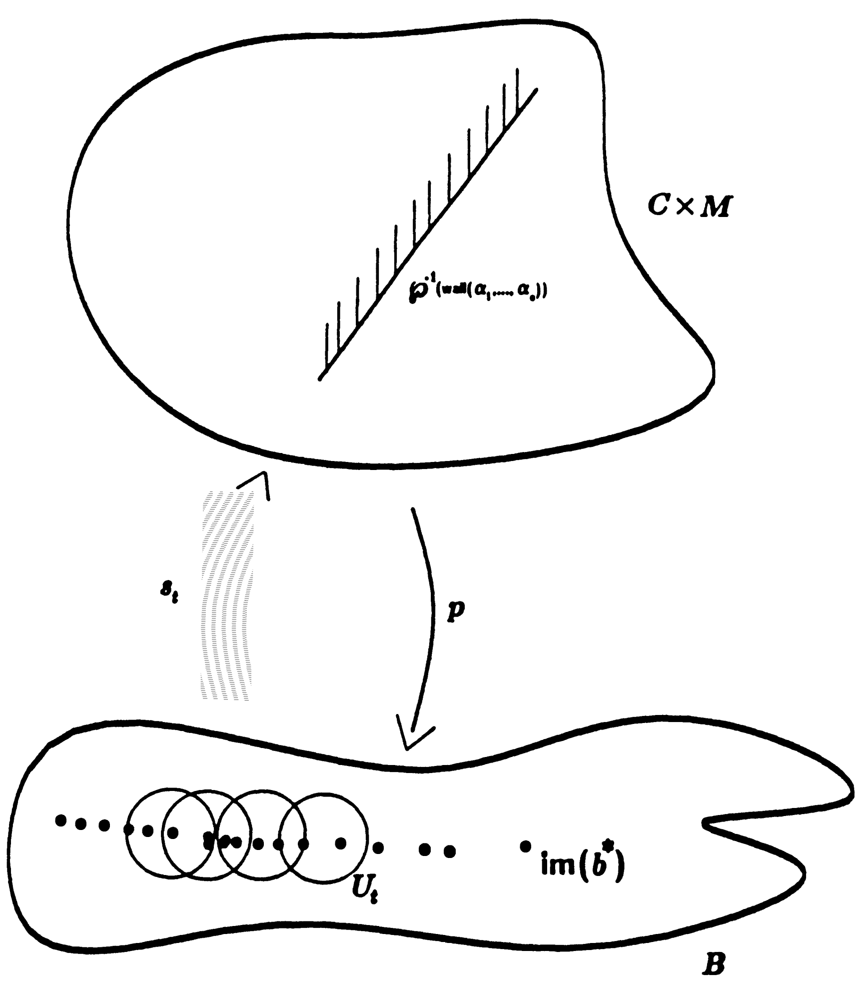

Figure 16: Shows how the perceptual equivalence class for $\mathtt{wall}(\alpha_1,\ldots,\alpha_n)$ is encoded using local sections of the world sensing bundle p, which is a concrete sensor.

The former interpretation we will call *absolute*; the latter we will call the *sensory* interpretation.[26] Note that we are ***not*** talking about the feasibility of building such a "perfect" sensing subsystem; we are discussing the meaning of the symbolic information passed between robot systems.

In light of the *sensory* interpretation above, consider the following. The sensory interpretation presumes some mechanism controlled by the robot, whose input is the world, and whose output is the symbolic piece of information $\mathtt{wall}(\alpha_1,\ldots,\alpha_n)$. Hence we may model this mechanism by some sensing map $\wp : C \times M \to \overline{B}$. We see that $\mathtt{wall}(\alpha_1,\ldots,\alpha_n) \in \overline{B}$, and that if $\wp(x,y) = \mathtt{wall}(\alpha_1,\ldots,\alpha_n)$, this means that when the robot is at configuration $x \in C$, then there is a wall with parameters $(\alpha_1,\ldots,\alpha_n)$ containing point y in the world M. The sensor space $\overline{B}$ then contains predicates such as $\mathtt{wall}(\alpha_1,\ldots,\alpha_n)$, and also some symbols like $\emptyset$ (meaning, "there is no wall at y visible from x", etc).

Now, the map $\wp$ defines a perfect virtual sensor—this sensor is analogous to an "oracle" in the theory of computation—it is a perfect information source that gives accurate "symbolic" information about the world. That is, $\wp$ takes $(x,y) \in C \times M$ as input, and returns perfect information about the existence of a wall at y. An alternative, equivalent interpretation is to say that $\wp$ accesses a perfectly accurate geometric model, and "looks up" whether or not there is a wall at y visible from x. Again, we are not claiming that we can build such a sensor or a model; this discussion of oracle-like sensors is only for illustration. Once we understand what it means for a perfect sensor to output "symbolic" information about the world, then we can generalize this notion to an imperfect computational process that interrogates the world, and matches its (imperfect) observations

[26] This distinction is temporary; in our geometric framework there is no difference.

to pre-encoded expectations about the structure of the world.

Now, consider the following alternative semantics for the statement

$$\wp(x, y) = \mathtt{wall}(\alpha_1, \ldots, \alpha_n). \tag{19}$$

We will stipulate that this statement encodes the following: *the sensor subsystem $\wp$ hypothesizes the existence of a wall containing y, with parameters $(\alpha_1, \ldots, \alpha_n)$*. That is, the subsystem has made observations of the world (through imperfect sensors), (possibly) retained state, and performed some computation. This computation relies on initial state, both in the form of a pre-existing robot program, and also in the form of pre-existing data-structures. Together, this initial state encodes expectations about the local fine-structure of the world—landmarks such as walls, corridors, hallways, etc. Let π_C be the natural projection of $C \times M$ onto C. The perceptual equivalence classes induced by $\wp$ (which we call *virtual* perceptual equivalence classes), are as follows: consider the inverse image of $\mathtt{wall}(\alpha_1, \ldots, \alpha_n)$, projected onto C—this is

$$\pi_C \wp^{-1}(\mathtt{wall}(\alpha_1, \ldots, \alpha_n)).$$

It consists of the set of configurations from which the existence of a wall with parameters $(\alpha_1, \ldots, \alpha_n)$ can be hypothesized. (Intuitively, the region from which such a wall can be "seen"). The boundary of this set contains the set of configurations touching the wall—this set is what we think of intuitively as the the "perceptual equivalence class *wall*."

Hence it is now clear how virtual sensors such as $\wp$ pre-encode expectations about the world. We model this pre-encoding as local sections of the world sensing bundle p (p corresponds to the "concrete" or actual sensors available to the robot). That is, corresponding to the value $\mathtt{wall}(\alpha_1, \ldots, \alpha_n) \in \overline{B}$, there is neighborhood U_t of B, and a local section $s_t : U \to C \times M$. This local section encodes how the sensor history (the image of b^*) is mapped into $C \times M$, to form the virtual perceptual equivalence classes of $\wp$. See fig. 16. Note that all such sections (eg., s_t) require *choices*, they are not canonical. These choices precisely encode our expectations about the local structure of the world. In mechanistic terms: "using" the sensor map p, we sense a series of values $b_1, b_2, \ldots \in B$. We find a subset U_t of B (or a series of subsets), containing these sensed values. We ask, what "features" in the environment could have caused these sensor values? This question is underdetermined, and, in any case, we usually do not have access to the inverse map p^{-1}. However, suppose we have some computer program—a computational process—that "matches" $b_1, b_2, \ldots$ against "expectations" about the world—in short, it produces a set S in $C \times M$. This set S represents a hypotheses—pairs $(x, y) \in C \times M$—that could have generated the sensor history $b_1, b_2, \ldots$ The process be be viewed as a local section of p with domain U_t. When projected onto C, the hypothesis S represents a virtual perceptual equivalence class. Note that it may be necessary for the computational process to have initial state, and to retain state (eg, previous sensor and control values) over time.

Hence, a key research challenge is to show how to construct virtual perceptual equivalence classes out of concrete perceptual equivalence classes. When recognizability is thusly constructive, we naturally obtain task-directed sensing strategies, driven by the information demands encoded in the structure of the recognizable sets. We have begun research to demonstrate this constructive program. For the interested reader, we sketch some of these constructions in appendix B.3.

8 Reconstruction

We now introduce a key concept called *reconstruction*. Since this notion is tricky, we take a first cut at a "naive" definition in idea 8.1. After some discussion, we refine this definition into def. 8.2.

8.1 What is Reconstruction?

There is much more to be said about our framework in the absence of world models. For example: Given a bundle $p : E \to B$ we call a map $s : B \to E$ a *section* of p if $p \circ s$ is the identity on B. A section is called "local" if it is only defined on a neighborhood U of B. Suppose we have a partial or incomplete model of the geometry of the world, and a partial or incomplete characterization of the sensor map p. In our view, a "small number" of remembered sensor values and control values defines a kind of local section of the world sensing bundle $p : C \times M \to B$. That is, this history and a partial model permits us to hypothesize a correspondence between current sensor values (and also "nearby" sensor values) and objects in the world. Hence history in general generates a "family" of these sections $\{ s_t \}$, parameterized by time. This allows us to define a computational problem that is related to what others have called "map-making" in mobile robotics. We call it "reconstruction." Here is an idea for how one might define it:

Idea 8.1 (∗) *Consider defining the* reconstruction problem *as follows. Given a partial world model of geometric and physical obstacles in M, that "projects" to sensor values by a world sensing bundle $p : C \times M \to B$, construct an approximation to the inverse map p^{-1} from a family of sections $\{ s_t \}$.*

This definition is approximate; in def. 8.2, below, we give a a precise characterization.

We view the value of *reconstruction* as a key topic of debate in analyzing "reactive" systems. In a way, reconstruction is a hybrid of "classical" planning (planning with world models) and reactive planning (planning with minimal models). Need a robot need reconstruct in order to be successful? Can we characterize which tasks and domains require reconstruction, and to what degree reconstruction can be ignored? These issues (since they concern the importance and utility of the map p^{-1}) are "dual" to questions of the importance of world models. That is, one may independently ask (for a task or domain): *(1) Are world models required? How much of a model is necessary?* and *(2) Is reconstruction required? How good an approximation of p^{-1} is necessary?*. Finally, we conjecture the following: *(3) Reactive systems do not reconstruct, or reconstruct minimally. A measure of "reactivity" may be how little the map p^{-1} is approximated, constructed, or used.*

8.2 Definition of Reconstruction

With the discussion of sec. 7 in mind, we are now prepared give a precise definition of reconstruction.[27] This definition refines idea 8.1. We need the following notion: we say that a section s of the world sensing bundle $p : C \times M \to B$ is *global* when the domain of s is all of B. We say a local section $s_t : U_t \to C \times M$ *can be globalized* (or *is globalizable*) when it can be extended to a global section. Of course, any global section *restricts* naturally to a local section for any subset U of B. We say two local sections are *incompatible* when there is no global section that restricts to both.

Now, consider, once again, "an office robot that is placed in the open desert." (This example may seem artificial, but in fact it is metonymous for a family of difficult sensor interpretation problems). Suppose the space of configurations C includes both of these environments, as does the world M. The robot constructs a set of sensor and control histories. It then uses a local section to match these histories against internal data structures; in effect, it hypothesizes a match between its observations and the expectation of being in an office environment. More specificially, this local section maps a set of sensor values to the region in $C \times M$ corresponding to the office environment.

On the other hand, given different initial state, the robot could match its observations to a desert environment. This corresponds to a different local section. More practically, we could imagine an office robot that performs tasks on some set of perceptually similar floors–or in perceptually similar "Pods" like at Xerox PARC. For each floor i, there could be a local section matching the

[27] We are very grateful to Prakash Panagaden for his suggestions regarding this construction.

observations of the robot to features on the i^{th} floor. Some more sophisticated query (asking a human) or sensing operation (reading a floor sign) could be used to disambiguate *which* local section is accurate. While perhaps all the local sections can be globalized, only one of the globalizations is correct. Hence, local sections represent local hypotheses (or local matches), and global sections represent global hypotheses (or global matches).

So, to recapitulate, a local section of the world sensing bundle represents a (local) hypothesis about what features in the world match against the robot's collection of sensor (and more generally, sensor and control) history.[28] Multiple simultaneous hypotheses correspond to several, incompatible local sections. The multiplicity of hypotheses, or the number of incompatible local sections for a given neighborhood U of B, corresponds to the "degree" of ambiguity in the world—the "symmetry" or the "number of similar-appearing environments" etc. For example, for the office robot in the desert, there are two incompatible local sections; one corresponds to the hypothesis that the robot is in Upson Hall at Cornell, the second to the hypothesis that it's in the Mojave desert. For an office robot in a building at 545 Technology Square (which has 8 non-Polaroidal floors these days), the incompatible local sections could be 8 in number, each corresponding to a hypothesis that the robot is on a different floor.

Now, the key point is that each of these (incompatible) local sections globalizes to a different global section. Only one of these global sections can be "correct"—that is the robot is only on one floor at MIT; the robot is either at Cornell or in the Mojave, etc. The remaining global sections represent other possible interpretations of the robot's situation.

With this intuition, we can now give the following

Definition 8.2 *We define the* reconstruction problem *as follows. Given a partial world model of geometric and physical obstacles in M, that "projects" to sensor values by a world sensing bundle $p : C \times M \to B$, approximate a family of global sections of p given a family of local sections $\{ s_t \}$.*

It is crucial to note that the two parameterizations in def. 8.2 are different. In particular, the given family of local sections $\{ s_t \}$ is typically parameterized by time (i.e., t is a time). The family of *global* sections is parameterized by the symmetries of the environments, or the different possible "worlds" or "floors" as discussed above. As a very simple example, imagine a mobile robot in a rectangular room. Suppose the robot has a range-finder but cannot sense orientation (hence, its sensing map is like p_x in eq. 2). Suppose further the robot has coarse odometry, and that it has a geometric model of the rectangular room. We model the odometry as a "relative position sensor," whose values (approximately) indicate relative motion. Now, suppose the robot is near a wall, and makes a motion (which it records using odometry) while obtaining range readings from that same wall. So these sensor readings and control history form a time-indexed sequence $b_1, b_2, \ldots$ in a sensor space that contains range values and odometry values. Suppose a set U contains these values in the sensor space B. The robot is possibly near one of the four walls. Hence there are four hypotheses about which wall the robot is near, and therefore four local sections mapping the sequence $b_1, b_2, \ldots$ into $C \times M$: one local section for each (local) hypothesis. So, each match of the sensor history to the "world" is accomplished by a local section—a pre-encoding of expectations. Each of these local sections may, of course, be globalized. Since the robot is actually near only one wall, only one globalization is correct. The globalizations are parameterized by the symmetries of the world, whereas the *local* sections are paameterized by the time-history of their acquisition. (More generally, the global sections will be indexed by the "degree of ambiguity" in the environment, or the "number of similar looking sub-environments", etc). For example, for this same robot in a room shaped like an n-gon, the world (locally) has n-fold symmetry, and hence there are n local sections. Now, of course, over time, the robot will encounter corners and be near two walls at once.

[28]The reason we blur this distinction between sensor and control history is that we can model the control history as "another sensor" and then use the product of this sensing map and our "true" sensors as an efective sensing map. The details are straightforward, but see sec. B.2.2 for more discussion.

At that point, if the robot has history, then the incorrect global sections may be pruned, since they are incompatible with a local section that is known to be correct.

> EXERCISE: *Show that, for a robot that cannot sense orientation, and has no model of the world, the global sections will be indexed by the rotation group.*

As another example, consider a robot that uses ultrasonic sonar to detect corners in the environment (see, eg., [KTB, BaKu]. Suppose a sensor history we will call b^* corresponds to recognition of a "corner" c_i. If the robot has a geometric world model containing n corners, then there are n sections, each corresponding to the hypothesis that the robot is on one of the corners. That is, we have that $p^{-1}(b^*) = \{\text{all corners in the world}\} = \{c_1, \ldots, c_n\}$. Each section $g_i : B \to C \times M$ $(i = 1, \ldots, n)$ has the property that $g_i(b^*) = c_i$. This property is quite general, and it relates the idea 8.1 above to the proper definition 8.2 we have given:

Definition 8.3 *Given a partial world model of geometric and physical obstacles in M, that "projects" to sensor values by a world sensing bundle $p : C \times M \to B$, suppose we reconstruct a family of global sections $\{g_i\}$ of p. We say the reconstruction is* complete *if, for any subset U of B,*

$$p^{-1}(U) = \bigcup_i g_i^{-1}(U).$$

8.3 Discussion: "Map-Making" and Reconstruction

We have seen that control history can be viewed as a kind of "approximate mapping" of time into the tangent bundle (specifically, a mapping into a quotient of TM). This mapping can be integrated to obtain a "dead reckoning" path $T \to C$. Sensor history $b : T \to B$ can be pulled-back by the bundle p to a mapping of time into the C-perceptual equivalence classes, eg, $b_* : T \to \mathcal{K}'$. How are these two combined?[29] One answer is as follows.

Consider a world-sensing bundle $p : E \to B$. In sections 8.1–7 we discussed how history together with a partial model permits us to hypothesize a correspondence between current sensor values (and also "nearby" sensor values) and objects in the world. The idea is that the local section $s_t : U_t \to C \times M$ sends a sensor value b to the pair (ϕ, y); this encodes the estimate "given what I know of your history and initial conditions, if you got sensor reading b, then you may be at configuration ϕ looking at point y." Naturally, if we have a geometric world model, the point y can be "looked up" to see what is there. Alternatively, the model could be updated "at y"—eg, we could add a "wall" or a "face" there, if b indicates this situation (for instance, if $b \in Z_m \subset B_0$, where B_0 is as in eq. (11)).

As time goes on, these remembered sensor and control values evolve, and may be parameterized by time. For simplicity, assume time is the unit interval $[0,1]$, which we denote I. Hence history parameterizes these sections s_t by time $t \in I$ and defines a family of sections of p. That is, if the local section at time t has the form $s_t : U_t \to C \times M$, then history defines a family of sections $\{s_t\}$, which we may write as

$$\begin{array}{rccc} s : & B \times I & \to & C \times M \\ & (b,t) & \mapsto & s_t(b). \end{array} \tag{20}$$

(The domain of this map may be some subset of B instead). Now, suppose we view the map s in (20) as being set-valued (i.e., $s_t(b)$ is a *set* in $C \times M$). The meaning of this set is, roughly, "given what I know of your history and initial conditions, if you got sensor reading b at time t, then you may be at configuration ϕ looking at point y for $(\phi, y) \in s_t(b)$."

[29] Classically, we could combine them using a kalman filter to obtain an "optimal" estimate of position. In this case, we would still ask, how does this new estimate enable us to lift (in the lattice) to a finer partition? The map s_t (below) could model a kalman filter.

From this viewpoint, it is clear how history enables us to lift to a finer partition of C as discussed in sec. D. Suppose the robot experiences a motion v while recording history. The motion moves in the RR-graph of partition $\mathcal{K}'$ as, say, $U \xrightarrow{v} V$. Suppose the detailed sensor history $I \to \mathcal{K}$ is of the form

$$U, U_{\alpha_1}, \ldots, V_{\alpha_1}, \ldots, V \tag{21}$$

as in, for example, construction 4.8 above. History during the motion v defines a family of sections $\{ s_t \}_{t \in I}$ on the union of the sets (21), as described above. We suppose the sections are set-valued. Then, having executed v, "$\mathcal{K}'$'s estimate" of the current state is V. But "history's estimate" of the state is $s(b_1, 1)$ where b_1 is the current sensor reading. The intersection of $V \cap s(b_1, 1)$ can define a smaller set, and hence (globally) a finer partition. Hence we can lift to this finer partition using history.

This allows us to define reconstruction as in def. 8.2.

Now, imagine a mobot with a geomtric world model. It is possible to characterize mobot algorithms by how "tightly" the loop *reconstruct, sense, reconstruct,*... is closed. This loop corresponds to $s_t, p, s_t, p, \ldots$ For example, a "classical" planner has a very tightly coupled "reconstruction" loop. To parody its behavior, it works as follows: (i) Sense a value (say, a feature of an obstacle). (ii) Build a polyhedral geometric model from that value and enter it into the world model. (iii) Use the new model to plan actions and execute them. (iv) Sense a value....

On the other hand, "reactive" systems[30] hardly ever close the loop; they have some hand-compiled knowledge of p built in the hardware, using little or no world model, and almost never construct or use the inverse map p^{-1}. For example, Mataric [Mat] first "pre-encoded" the map p in the architecture (in the form of landmarks, which may be viewed as C-perceptual equivalence classes); her highly successful robot computes entirely with equivalence classes of the sensor space B. The robot never reconstructs or builds a geometric world model; the model it builds is essentially a B-encoded RR-graph (as we discuss in sec. 8.3). By "never reconstructs," we mean the robot never attempts to approximate or use the inverse map p^{-1}.

There are many other interesting aspects of the work by Brooks and Mataric; it would be reductionistic to claim that the characterization above was anything near complete. Nevertheless, it does shed some light on what kind of representations and computations are emphasized in "reactive" systems. We conjecture that one way of viewing these systems is as follows:

- Reactive systems do not reconstruct, or reconstruct minimally. A measure of "reactivity" may be how little the map p^{-1} is approximated or used.

It is not clear whether a robot need reconstruct in order to be successful. It would be of great value to characterize which tasks and domains require reconstruction, and to what degree reconstruction can be ignored. These issues (since they concern the importance and utility of the map p^{-1}) are "dual" to questions of the importance of world models. That is, one may independently ask (for a task or domain):

- *Are models required? How much of a model is necessary?*

and

- *Is reconstruction required? How good an approximation of p^{-1} is necessary?*

One way of attacking these questions is to consider the extremes. In the following section we begin the attack and do so. First, in sec. 9 we ask, given good geometric models of the environment

[30] For example, consider the robots of Brooks [Brooks].

and the bundle p, can we compute the C-Perceptual equivalence classes? What are algorithms for doing so, and what is their complexity? Next, in sec. 9.2 we explore the question: given no geometric world model (or a minimal model), how can we compute the C-Perceptual equivalence classes and their RR-graph? How can the lattice of recognizable sets be computed? How can one plan using these structures?

8.4 Review of Key Concepts

1. Local sections encode hypotheses of expectations about what features in the world correspond to a history of sensor values.
2. The computational process associated with a local section is like matching in model-based recognition.
3. Multiple, incompatible local sections correspond to incompatible match hypotheses.
4. Local sections can be globalized. The reconstruction problem is to find all possible globalizations.

9 Computing C-Perceptual Equivalence Classes

We now address details of computing RR-graphs and C-Perceptual equivalence classes. We consider two different problems. The first is: given a geometric world model, and a representation of the world-sensing map p, compute the C-Perceptual equivalence classes. The second is: given no model, make a map of the C-Perceptual equivalence classes and RR-graphs.

9.1 Suppose we have a good geometric model

Perhaps the best application for our work is in the case where the environment is unknown or unmodelled. In this case, we wish to have the robot build up an RR-graph from observations about the world and monitoring of its motion. As the robot gains more information, it can construct the lattice structure as well. In the absence of reconstruction (sec. 8.3) or a built-in geometric world model, planning becomes a process of searching, and exploring, an RR-graph. This situation is discussed below in sec. 9.2. We have set up most of our machinery to make the algorithms for these computations fairly straightforward.

Nevertheless, it is clear that the C-perceptual equivalence classes have an interesting structure, and we may hope to exploit this structure in sensor-based planning. One may investigate our theory by picking a particular world model, and then asking what are the C-perceptual equivalence classes given some representation of the world sensing bundle p). This does not mean that our approach is limited to cases where a world model is available (see sec. 9.2 below). Instead, we merely suggest that the construction of the C-perceptual equivalence classes should be studied in the case where a model is known, and that these "maps" provide invaluable data on the structure of the recognizable sets. Finally, the ability to construct the C-perceptual equivalence classes and RR-graphs from a world model will indeed be useful in those structured situations where a model is available. However, the recognizable sets are fairly complicated, and somewhat tedious to construct by hand. In this section, we give a general algorithm to compute them.

Our basic result is: when (1) the sensor space B is finite, (2) the geometry of the world can be encoded as a solid model, and (3) we can "write down" the world sensing bundle p so that an algorithm can manipulate it, then the C-perceptual equivalence classes are efficiently computable. Note that the finite sensor space B could be obtained by breaking up a continuous sensor space into a finite number of equivalence classes. A comment on (3): we will assume that we can write down the geometric world model and the map p using the language of semi-algebraic sets.

More specifically, we assume that a geometric model is available, and that it may be expressed in the language of real semi-algebraic sets—i.e., using real polynomial inequalities with quantified variables and logical connectives [Tarski; Yap; Grigoryev]. We also assume that the bundle $p : M \times C \to B$ can be written down as a semi-algebraic map (recall that we say a map is (semi)-algebraic when its graph is a (semi)-algebraic set). We digress to an example, to show why it is reasonable, and not very restrictive, to assume p is semi-algebraic. We can show:

Lemma 9.1 *Suppose that B is finite and contains n values, and that M and C are algebraic and each has dimension d. Then the C-perceptual equivalence classes can be computed in time polynomial in n and exponential in d.*

Proof: See below. □

As a corollary, we see that "sensor fusion" (adding new sensors and sensor values) is hence polynomial-time, since adding sensor values only increases n.

Finally, we note that if our modelling is adequate to predict (simulate) the outcome of motion strategies (under uncertainty), then, having constructed the C-Perceptual equivalence classes using the algorithm above, then we can construct an RR-graph whose edges denote *guaranteed* or *strong* (instead of *possible* or *weak*) reachability. For example, weak reachability may also be viewed as follows: From the planning point of view, we may write $U \xrightarrow{v} V$ if it is *possible* to achieve V from U by doing v. Hence we view deciding this relationship without executing v as a form of planning. Algorithms for this problem are sometimes called *forward projection* (under uncertainty), and have been developed, for example, in [Erd,Buc,Don,Can]. The assumption that we can compute accurate forward projections is rather strong (in the mobot domain) but may be possible in semi-structured environments, such as mechanical parts assembly.

9.1.1 Example: A Semi-Algebraic World-Sensing Bundle

We show, by example, how the kind of world sensing bundles we have encountered in this paper can be modelled as semi-algebraic functions. This example is intended to convince the reader that if we assume a world model and a representation of the sensors, then it is reasonable to assume that these models are semi-algebraic. In fact, almost any kind of sensing function that we can "write down" and compute with will be semi-algebraic.

Let the world M be $\Re^2$ and the configuration space C be $\Re^2$. Then the obstacles in the world are a semi-algebraic set O. Consider the bundle $p : C \times M \to B_0$, where B_0 is defined in eq. (11).

Now, suppose we have $(x, y) \in C \times M$. Let $r(x, y)$ be the (infinite) ray from x towards y. Define the function $d(x, y)$ to be the distance from x to the nearest obstacle along ray $r(x, y)$. It is clear that d is semi-algebraic: we first intersect the ray with the obstacles O and then quantify over the intersection to get the closest point. Let l be the distance from x to y. Now, define p as follows. Recall m is the maximum range of the rangefinder. The table is ordered hierarchically:

$$p(x, y) = \begin{cases} \emptyset, & \text{if } d(x, y) > l; \\ \infty, & \text{if } d(x, y) < l; \\ \text{o}, & \text{if } d(x, y) > m; \\ \lfloor d(x, y) \rfloor, & \text{otherwise.} \end{cases} \tag{22}$$

Now, we wish to show that the map p is semi-algebraic. First we represent B_0 by the ring of integers Z_{m+3}, using m to represent $\emptyset$, $m + 1$ for ∞, and $m + 2$ for o. Second, we need to know that floor can be computed by a semi-algebraic "machine:"

Proposition 9.2 *The floor function can be computed by a machine that only branches, loops, and performs semi-algebraic computations.*

Proof: Consider the machine N that takes as input a pair of algebraic numbers (x, n). If $x < n$ then N halts and outputs $n - 1$. Otherwise, N calls itself on the input $(x, n + 1)$.

For positive x, we run N on the pair $(x, 0)$ to compute $\lfloor x \rfloor$. □

Hence we conclude that the bundle p is semi-algebraic. This concludes the example.

9.1.2 Complexity

Suppose we have a semi-algebraic model of the world, and of our sensing. That is, suppose the obstacles and the world sensing bundle are semi-algebraic as described above. We now prove lemma 9.1:

Proof (lemma 9.1): Each set $\pi_C p^{-1}(b)$ is a semi-algebraic set (for $b \in B$). There are n such sets, and the C-perceptual equivalence classes, by def. 4.4 and lemma 4.5 are the cells in the arrangement in C formed by these generating sets. There are n generators in d dimensions, so the arrangement has complexity $O(n^d)$. □

This very general algorithm is chiefly of theoretical interest. However, the C-perceptual equivalence classes have such a simple structure, that faster algorithms can almost always be obtained in particular cases.

9.2 Suppose we have no model: Incremental Computation

Suppose the robot has no model of the world, but it wishes to build up a reachability graph of the recognizable places. Incremental computation of RR-Graphs and C-Perceptual Equivalence Classes proceeds in the obvious way: the C-perceptual equivalence classes are completely classified by the sets $\{\pi_C p^{-1}(b)\}$, for $b \in B$ (lemma 4.5). Hence they are completely determined by the sensor values b. Let us imagine that B is discrete, or at least that it is the quotient of the true "continuous" sensing space—e.g. that we have broken the sensing space into regions that are close enough to "look the same."[31] Suppose the robot starts out sensing some value b_0. It moves, performing action v. The sensors now read b_1. Hence the robot can build the graph $b_0 \xrightarrow{v} b_1$, which of course is shorthand for $\pi_C p^{-1}(b_0) \xrightarrow{v} \pi_C p^{-1}(b_1)$. The latter is a (weak) reachability edge in the RR-graph. At any rate, subsequent moves add more edges to the RR-graph, which is the robot's representation of the recognizable places and the reachability relationships between them. We call the resulting structure a **B-encoded RR-graph.** "Planning" then reduces to a search of the RR-graph—without reconstruction (definition 8.3 below) or a "built-in" world model, this is the only type of planning possible. See, for example, [Mat]. Note that in general, we may consider the RR-graph to be only partially known.

9.3 The Interaction of History and Incremental Computation

Next, the robot may take different "tours" of the RR-graph, perhaps using different sensor arrays. The effect of this will be a finer distinguishing of the C-Perceptual equivalence classes. Constructing a finer partition corresponds to ascending in the lattice; "finer" (i.e., better) sensing corresponds to an inclusion map.

While the utility of such RR-graphs (i.e., maps) and the ease of their construction is clear, more work is required on their structure and properties. For example, consider a robot that makes several "circuits" of a room along approximately the same path. Eg, the robot circles around and around in a loop, revisiting old places and scanning with its sensors, recording data on the world and on its controls. Two effects may interact: (1) knowledge of the state is blurred by motion, and

[31] This is not necessary, but it's convenient for building intuition. It's also what one would do in practice.

(2) knowledge is increased by history (experience). The second effect was described in some detail above in sec. 4.8.3.

Let us be more specific. Presumably this "circuit" action—several tours of the room—will simultaneously tend to increase uncertainty (in that the odometry estimate may get worse over time), and reduce uncertainty (in that more and more accurate "maps"–RR-graphs–are being constructed). One effect of navigation with "real" maps is that recognition of "unique" landmarks can zero out uncertainty. Is there a similar effect with the RR-graphs? Another question is: the "circuit" described above will induce a path in the RR-graph (hopefully the path is a loop, i.e., it revisits familiar equivalence classes). But if in the motion it gains information (as we surely hope it will), then this loop will ascend in the lattice, and become "helix-like." Its forgetful projection will of course be the original circuit. The total amount that the loop rises will measure precisely the amount of information "learned" about the environment. Furthermore, the "distance" in the lattice that the helix rises each time measures the incremental gain information from one more circuit. One might conjecture that the helix might eventually converge to a loop higher up in the lattice; this would represent the situation in which no new information could be gained by orbiting the room. There might even be situations where continuing to move might *reduce* the amount of information! These issues require further study.

10 Conclusions

We considered how a robot may interpret its noisy, sparse sensors, and direct its actions so as to gain sufficient information about the world to achieve a task. Sensing forces us to view the world as a projection, and this projection divides the world into perceptual equivalence classes. We investigated how to model various sensors, including range-finders, cameras, and odometry. We build new sensing maps from old ones, creating a hierarchy of virtual sensors. These constructions are guided by the information needs of the task.

We are developing tools, including virtual sensors, the recognizability gradient, and the perceptual limit, which permit a precise characterization of the information flow into and through the robot. The robot's capabilities may be explored, as well as fragilities in robot plans or programs, due to problems in matching expectations against observations.

We identify global landmarks, which are the recognizable sets in configuration space, the C-perceptual equivalence classes. We can compute them, given a world model, and we developed efficient algorithms for doing so. In the absence of a model, these "landmarks" can be acquired incrementally, through exploration. A graph of these explorations, the RR-graph, encodes the reachability relationships between the recognizable sets. Again, it may be computed from a model, but in the absence of a model we can "encode" it using the sensor values and history alone. For a sensor space B, we call this a "B-encoding" of an RR-graph. This representation seems useful for mobots with minimal models.

RR-graphs and C-perceptual equivalence classes are simple concepts; these recognizable sets form the basic building blocks for our theory. These landmarks, or recognizable sets—that is, the places in the world that the robot can recognize and distinguish between—have structure we can exploit in developing sensor-based planning algorithms. In particular, the interaction of the lattice and history is subtle and can be used to plan or program sophisticated strategies for achieving goals and accomplishing tasks.

Our goal is to develop a framework for reasoning and planning in the presence of uncertainty, and, in particular, with minimal models. We believe our ideas can help a robot to determine what information is required to solve a task, and what actions help acquire that information to solve it. It is also our hope that this framework is useful in a broad class of domains, and can be thought of as a programming and planning paradigm for reasoning about about uncertain and unknown environments. We are currently implementing our ideas on mobile autonomous platforms, and developing the capability to construct, compute, and recognize perceptual equivalence classes, to plan and reason using them, and in particular, to exploit the lattice, bundle, RR-graph, and history structures and algorithms described here. For a description of our architecture and hardware, see [OdD, BSD]. For more on our experimental work, see [BD, RD]. In the appendix to this paper, we provide details of the constructions and arguments above.

10.1 Task-Directed Sensing

A critical property of our framework is that the data structures and representations employed are *a priori* recognizable features, or landmarks, in the world. Our theory is derived from a "calculus of the observable", and when recognizability is thusly constructive, we naturally obtain task-directed sensing strategies, driven by the information demands encoded in the structure of the recognizable sets.

One advantage of this formulation may be that it defines a set of experiments that the robot may perform to try to recognize a virtual perceptual equivalence class. That is, this construction of virtual perceptual equivalence classes explains how the observed sensor values should change with respect to strategies the robot can perform. By performing these actions and observing its sensors, the robot may verify hypotheses about the existence and geometry of the virtual perceptual equivalence classes it must recognize in order to perform a task. Specifically, we may regard certain sensors as "concrete" sensors, relative to a high-level "virtual" sensor. The resulting "concrete" perceptual equivalence classes are not likely to be human-readable in general. More likely, a robot task might be specified in terms of walls, corridors, and hallways (for example: "Go down the hall, take the second right, and follow the wall as it curves around until you reach the atrium.") These are the virtual perceptual equivalence classes with respect to which a human might wish to express a task. But how can the robot recognize these classes, using the sensors and computation it carries on-board? Instead of trying to provide the robot with "perfect sensors" such as a "wall oracle", a "hallway oracle", etc., instead, we can tell it how to recognize virtual perceptual equivalence classes from concrete ones. Using this method, the robot can then perform experiments to verify which virtual perceptual equivalence classes it is in. This is the process we call *task-directed sensing.*

10.2 Open Questions

There are many theoretical and experimental problems left that demand exploration. Much of our framework is preliminary and requires elaboration and experimental verification. In addition, there are open theoretical questions. We close with some examples.

The transformation that maps a lattice with inclusion maps into the same lattice with forgetful maps map be regarded as a functor to the opposite category. Since our sensor values so often have additional structure (eg, as a ring), this leads one to believe that the transformation is actually a *pre-sheaf.* Viewing our lattice as a pre-sheaf yields a nice explanation of the contravariance and "lifting" evident in the commutative diagrams (48) and (49) from lemma D.1. Furthermore, one is naturally led to ask whether we indeed have a *sheaf.* If we do, the "pull-back" property of sheaves yields a family of disambiguating strategies using local sensing. We hope to investigate this avenue in future work; for some ideas, see appendix B.2.

We hope our methods could be used to develop "lower bounds" analyses for mobile autonomous agents. For example, it would be interesting to prove that for robots which do not reconstruct, RR-graphs and lattices of RR-graphs are "universal internal data structures." This would mean that for such robots, the most detailed model they could build of the environment would be such a lattice of RR-graphs. The "detail" or information complexity of this data structure would in turn, be bounded by the perceptual limit. For robots with initial models or state, in turn, we could hope to characterize their additional power by using local sections to encode matches of observations against expectations about the world. In turn, such a theory could provide a way to say what robots are capable of what tasks.

Finally, we are implementing and testing the theories above on physical mobile robots [BD,RD]. Much more research is needed. Work is underway.

Appendix

We now undertake a more detailed analysis of some ideas and results described above. Some of these sections formalize the discussion in the main paper. Others cover advanced topics. Of these sections, perhaps the most important are:

B — *Modeling Sensors*, particularly B.3 (*Virtual Sensors*).

C — *Effects of Sensor Noise,* particularly C.3 *(Perceptual Aliases)*.

A Perceptual Equivalence Classes

A.1 Quotient Structure

Recall our definition 4.1 of the perceptual equivalence classes at a fixed configuration x. In sec. 4.1 we gave an intuitive description of the finar sensing map p_x^* which uses θ, the orientation of the sonar return, to distinguish between finer perceputal equivalence classes. We now formalize the structure of this map.

Recall from sec. 4.1 that we wished to extend our example to take θ, the direction of the sensor, into account. By remembering θ, the robot may distinguish points within one annulus. In this case the perceptual equivalence classes will be finer, and are conceptually sketched in fig. 6. Here partition of the world (around x) is finer than in fig. 3, and the perceptual equivalence classes isomorphic to radial line-segments of unit length at orientation θ and integral distance from x.

We noted that the partition of M in fig. 6 is *finer* than the partition in fig. 3. Corresponding to this finer partition is a "finer" map p_x^* which takes θ into account in the natural way. Intuitively, p_x^* should be be a quotient map that induces the partition into perceptual equivalence classes depicted in fig. 6, by using information about θ to disambiguate objects at the same distance but at a different relative orientation.

Construction A.1 *We construct this map as follows. Let B^* be the larger sensor space $B^* = S^1 \times B$ where B is as in eq. (1). Then we define p_x^* to be "$\mathrm{id}_{S^1} \times p_x$" where* id *is the identity on S^1. That is*

$$p_x^* \;:\; \begin{array}{ccc} S^1 \times M & \rightarrow & B^* \\ (\theta, y) & \mapsto & (\theta, p_x(\theta, y)). \end{array} \tag{23}$$

It is easy to see that each perceptual equivalence class in fig. 6 is of the form $\pi_M p_x^{*-1}(b)$ for $b \in B^*$.

Do not be disturbed that the robot both controls and "senses" θ in example A.1 (above). We could modify the example to model a "rotating rangefinder" like a sweep sonar. Such a device consists of a linear rangefinder on a constantly rotating platform. In such a device, θ is not servoable, but it may be sensed. The device returns a value of the form $(\theta, p_x(\theta, y))$. Hence, it can be modelled by a sensor map of the form

$$p_x^+ \;:\; \begin{array}{ccc} M & \rightarrow & B^* \\ y & \mapsto & (\theta, p_x(\theta, y)). \end{array} \tag{24}$$

An alternative to a "sweep" model is to assume a ring of sensors giving nearly continuous coverage, as often sported by mobile robots.

Later, we will incorporate the sensor controls (in the case of p_x, S^1) into the configuration x of the robot. In that case, the sensor "aiming" can be viewed as a kind of control parameter that we command and sense. In this case, control and sensing of θ are perfect, but this need not be the case.

We could, if we wished, encode sensor values in a continuous range of values. One way to do this would be to use -1 to represent $\emptyset$ and $d_{max}+1$ to represent ∞, where d_{max} is the maximum range of the range-finder.

Construction A.2 *Suppose we had an array of sensors, modelled by mappings $p_{x,1}, p_{x,2}, \ldots$, where each map $p_{x,i}$ has a sensor domain B_i. We could generate perceptual equivalence classes by defining a new, combined sensor space that is the product of the B_i; then the perceptual equivalence classes would be classified by the projection onto M of the inverse images of sensor values in $\prod_i B_i$.*

Alternatively, the equivalence classes are the cells in the arrangement *generated by the sets*

$$\{\, \pi_M p_{x,i}(b_i) \,\}$$

for $b_i \in B_i$.

If the sensors are noisy it is convenient to view the sensor values as subsets of B. This corresponds to using the powerset 2^B as the space of sensor values. Compare the comments in appendix A.2.1 and C below.

A.1.1 Perceptual Limits and Generalized Characteristic Maps

In sec. 4.1, we saw how, once we fixed a sensing map (like p_x), different environments can generate different partitions into perceptual equivalence classes. We suggested that all these partitions can be partially ordered on fineness. The resulting structure is a *lattice*, and it is dominated by its direct limit, which we termed the *perceptual limit* of p_x. The perceptual limit represents the "smallest features" the robot can interdifferentiate, in the best case. So, the perceptual limit is obtained by fixing the sensing map p_x and varying the environment. Recall that the perceptual limit has the following constructive significance. For a finite lattice, any obtainable partition $\mathcal{U}_\alpha$ of the world into perceptual equivalence classes can be constructed by finite unions of the classes in the perceptual limit $\overline{\mathcal{U}}$. Hence, we think of a geometry G_α as "controlling" how the classes in $\overline{\mathcal{U}}$ are unioned to obtain $\mathcal{U}_\alpha$.

We now make this construction precise. In particular, we formalize the precise manner in which a geometry G_α and a perceptual limit $\overline{\mathcal{U}}$ generate a partition $\mathcal{U}_\alpha$ of the world into perceptual equivalence classes.

Consider a world sensing map p_x centered at x. The effect of a geometry G_α is to assign "meaning", or "features" to space. Consider a function χ_α that maps subsets of M to a feature space W. We could define χ_α by the following rule. The domain of χ_α is the set of finite rays anchored at x. Assume these rays are closed, i.e., they contain their endpoints. Any such ray may be identified with its polar coordinates (r,θ). Define feature space to be $W = \{\,\emptyset, \infty\,\} \cup \Re$. Let F_α denote the complement of the obstacles (walls, etc) in the geometry G_α. F_α is open; $\overline{F_\alpha}$ denotes its closure.

$$\begin{array}{rcl} \chi_\alpha \; : \; M & \rightarrow & W \\ (r,\theta) & \mapsto & \begin{cases} \emptyset & \text{if the ray } (r,\theta) \subset F_\alpha \,; \\ r & \text{if the ray } (r,\theta) \subset \overline{F_\alpha} \,; \\ \infty & \text{otherwise.} \end{cases} \end{array} \tag{25}$$

We call χ_α a *generalized characteristic function*; we see the a *geometry* G_α is defined to be a pair, (M, χ_α). So, a geometry is a set of spatial coordinates, and a function that tells us what features are found at those coordinates.

Next, we define a *pure* sensing function to be a map ω from the feature space W to our sensor space B. In our example we have:

$$\begin{array}{rcl} \omega \; : \; W & \rightarrow & B \\ f & \mapsto & \begin{cases} \lfloor f \rfloor & \text{if } f \in \Re; \\ f & \text{otherwise.} \end{cases} \end{array} \tag{26}$$

We now see that we have

$$M \xrightarrow{\chi_\alpha} W \xrightarrow{\omega} B$$

so that $p_x = \omega \circ \chi_\alpha$ is the composition of the generalized characteristic function and the pure sensing function. So, varying the geometry G_α (while retaining the same set of features and the same pure sensing function) consists of varying the generalized characteristic function χ_α. Hence there is a deep sense in which the feature space W determines the perceptual limit.

Our construction is localized at x; to pass to the configuration space perceptual equivalence classes, we must subscript the local constructions above by the configuration x and then globalize them into a bundle as we did in the main paper. This globalization follows *mutatis mutandis* from the main paper.

A.2 Lattice Structure

A.2.1 Reachability Graphs of recognizable Sets

The following notation is helpful in analysing motion strategies under uncertainty, and in developing their relation to the lattice of recognizable sets. Let G and H be subsets of M. We denote $G \cup H$ by GH. Consider the difference between the motions notated by

$$D \xrightarrow{v_1} GH$$

and

$$D \xrightarrow{v_2} \{ G, H \}.$$

If in a motion strategy v_1 the robot recognizably achieves GH this means that the run-time executor can determine that G or H has been achieved, but cannot necessarily tell which of G or H has been entered. If the robot recognizably achieves $\{ G, H \}$, then it can further distinguish *which* of G or H it has reached. GH is called the *union* while the set notation is called the *distinguishable union.*

Consider the non-deterministic self-map of $\{ LU, R, D \}$ discussed above in 2. If we wished to consider deterministic maps, we could pass to the powerset $2^{\{ L,U,R,D \}}$; this construction is explicit in, for example, [Erd, Don89, E89]. In this case, $\{ D, LU \}$ would be a knowledge state, and when we denote the motion v by $D \xrightarrow{v} \{ D, LU \}$ we mean that v achieves the distinguishable union of D and LU from D.

A.2.2 General Properties of the Recognizability Lattice

In general, given two distinct partitions $\mathcal{U}$ and $\mathcal{V}$ of M into perceptual equivalence classes $\mathcal{U} = \{ U_\alpha \}$ and $\mathcal{V} = \{ V_\alpha \}$, either $\mathcal{U}$ or $\mathcal{V}$ will be finer, or they will be incomparable. If $\mathcal{U}$ is finer than $\mathcal{V}$ we write $\mathcal{V} \hookrightarrow \mathcal{U}$. This relationship $\hookrightarrow$ defines a complete partial order on partitions, and the set of all partitions of M hence forms a lattice. We note that the partition $\{ y \}_{y \in M}$ of M into all its one point sets corresponds to perfect sensing (any two points are distinguishable). This partition is finer than all others; we simply denote it by M in (6). Similarly, the partition of M into just one class $\{ M \}$ corresponds to *no* sensing (all the world looks the same); this is the least fine partition. For example, it is $\{ LURD \}$ in (6). We now see the following contravariant relationship between inclusions and forgetful maps:

Claim A.3 *Every inclusion map $\mathcal{V} \hookrightarrow \mathcal{U}$ induces a forgetful map $\mathcal{U} \to \mathcal{V}$ as shown in (5).*

Corollary A.4 *Every motion strategy (or sensing strategy) that loses information induces a motion in the lattice that can be modelled as the composition of a forgetful map and a reachability map.*

Every motion strategy (or sensing strategy) that gains information induces a motion in the lattice that can be modelled as the composition of an inclusion map and a reachability map.

We note an important one structural comment:

Proposition A.5 *Denote the partition* $\{y\}_{y \in M}$ *of* M *into all its one point sets by* M. *Let a sensing function* $q_x : M \to B$ *act as a quotient map and induce some partition* $\mathcal{U}$ *of* M. *Then the forgetful map* $M \to \mathcal{U}$ *is the same as the quotient map* q_x.

A.3 Example: Uncertainty in Sensor Control

It is useful to give an example in which control uncertainty is significant. Consider the example of an "aimable" rangefinder once again. We regard the "aim" as commanding some configuration of the rangefinder θ_0 in S^1. However, suppose that due to imperfect control, the rangefinder might actually rotate to some actual orientation θ instead, although we may know that $|\theta - \theta_0| < \epsilon$. We might have feedback (eg, shaft encoders) to approximately sense the new position, that is, to return a sensed value of orientation θ^*. The values θ_0, ϵ, and θ_* may be combined into some global estimate of the orientation. This orientation estimate–an approximation to the true heading of the rangefinder–will then be a factor S^1 of the sensor space B.

B Example: Modeling Sensors

We now proceed to explore just how physical sensors are modeled, and we introduce the concept of "virtual sensors".

Consider a range-finding sensor such as an ultrasonic sonar transducer (this discussion would also apply to a laser range finder). The hardware that supports these transducers has been designed to provide time-of-flight information necessary to determine how long it took a sonar ping to travel to an object and back again. If we make an assumption about the speed of sound in air, we an obtain approximate distance information given the time of flight.

Hence the physical range-finding sensor returns a value b_p which is an integer—say, the time of flight measurement to the resolution permitted by the hardware timer on a mobile robot. It is also possible to get no return. Hence, we can model the hardware as returning a value b_p in a physical sensor space B_p, where $B_p = Z \cup \{\infty\}$, where ∞ means "no return."

We do not, however, choose to model the range finder as a mapping from controls to B_p. That is because we wish to define a sensing map that encodes the *information* obtained in the return ping. The information contained in a (non-infinite) return ping is not simply "There exists an obstacle at distance d." It is much more:

> *(*) "There exist no obstacles up to distance d; there exists an obstacle at distance d; and beyond d we have no information."*

Consider our simplified model of a range finder, $p : C \times M \to B$. Let us include the sensor controls $\theta \in S_1$ as a factor of C so that the local sensor map is simply $p_x : M \to B$. We developed this sensor as follows. First we asked the question:

> *(Q1) Given a point y in the world M, what effect does y have on the values we sense?*

We call the answer to this question the *pointwise* definition of the sensing map: it gives us the rule $p_x(y) = b$. Now, using direct sums it is easy to say what sets of values we get when we apply p_x to a *set* S of points in the world:

$$p_x(S) = \bigcup_{y \in S} \{ p_x(y) \}. \tag{27}$$

For example, suppose we apply p_x to a ray $S_{x,\theta}$ originating at x and pointing in direction θ. Suppose the first object that the ray intersects is at distance d from x. Then clearly we have

$$p_x(S_{x,\theta}) = \{\, \emptyset, \lfloor d \rfloor, \infty \,\}. \tag{28}$$

Note that eq. (28) precisely encodes the information in statement ($*$) above. To summarize: we have a physical sensor whose domain is B_p. It encodes the actual values returned as signals or symbols from the hardware. We model the information contained in these signals by a pointwise map such as p_x (the range sensor) whose domain is B. The behavior of this map on sets defines the information content of the sensor. Note that this information is captured by (a) the set-valued map $p_x : 2^M \to 2^B$, (b), an "accumulation" or "interpretation" function on sets of values, that tells us that eq. (28) *means* the paragraph ($*$). It may be useful to think in terms of programming language concepts: (a) corresponds to a `map`[32] of p_x over S; (b) corresponds to `accumulate` applied to the result of `map`.

Finally, we note that B_p may be viewed as a *projection* of B. In essence, the sensor map p_x may be viewed as an *virtual* sensor that encodes the information available in the physical sensor whose range is B_p. The process of interpreting the physical sensor may be viewed then as a lifting of the physical sensor to p_x:

$$\begin{tikzcd} & B \arrow[d, "\pi"] \\ M \arrow[ur, "p_x"] \arrow[r] & B_p. \end{tikzcd} \tag{29}$$

We present models for other types of sensors, such as cameras, in section B.2. In that section we also explore odometry as a sensor, and work through a more detailed virtual sensor example.

B.1 Example: Probability Distributions

We note in closing that the sensor "value" b could represent a probability distribution as in [ME, Mor]. In this case, the information content of the sensor would still be ($*$). We construct a virtual sensor $\mathcal{P}_x$ that takes the output of p_x as its input. The virtual sensor $\mathcal{P}_x$ works as follows. Let S be a subset of M, as above. Given the information ($*$), which is encoded as the set $p_x(S)$, $\mathcal{P}_x$ treats this information as its only prior, and constructs a distribution. Hence, $\mathcal{P}_x(S)$ will be a probability distribution over S that represents the probability, given ($*$), of the existence of obstacles at any point y in S. That is, the distribution $\mathcal{P}_x(S)$ maps points y in S to the probability of y being inside an obstacle. Now, in general, this model requires the domain of $\mathcal{P}_x$ to be a function space—all possibile probability distributions over all subsets of M. However, in practice, B would be a simple parametric family of distributions. The distributions used in practice are very simple, and are parameterized by the values of p_x. This example illustrates the power of virtual sensors. See sec. B.3 for more details on virtual sensors.

B.2 Example: Modeling Complex Sensors

In this section we discuss how to model some types of sensors, particularly, a camera, odometry, and "virtual sensors". The last can be useful in the high-level programming of autonomous agents.

B.2.1 Example: Modeling a Camera

Suppose we have a 1 pixel greylevel camera. Suppose there are 8 bits of greylevel and so we may imagine the "value" of this sensor as lying in Z_7, the integers 0 through 7. Suppose the camera is

[32] `map` in SCHEME and ML; `mapcar` in COMMONLISP.

[32] This example arose in discussions with Matt Mason, and we are very grateful to him for his suggestions and insight in developing it.

situated and pointed as determined by its configuration x in sensor configuration space C, we now ask question (Q1) above. The answer to (Q1) is: the camera essentially measures luminosity, and hence a point y in the world can cause the camera to "read" between 0 and 7. ("Transparent" and occluded points, of course, will read zero also).

We now set about to model the information available in this sensor. First we define the following function on any set S in the world M:

$$\mathtt{map}(p_x)(S) = \bigcup_{y \in S} \{ y \} \times \{ p_x(y) \}. \tag{30}$$

Thus, the result of (30) on a set is a "table" of what luminosity values are assigned to each point y in S. A little thought shows that this "table" is exactly a *function*, namely the restriction of p_x to S:

$$\mathtt{map}(p_x)(S) = p_{x|_S}. \tag{31}$$

Next, we have to add up all these luminosity values to obtain the value representing the camera signal. That is, the value of the sensor on a set S is not $p_{x|_S}$, but rather the sum[33] $\sum_{y \in S} p_x(s)$. More generally the sum is infinite and of course we write this as an integral:

$$\mathtt{Accumulate}(p_{x|_S}) = \int_S p_{x|_S}. \tag{32}$$

We summarize as follows. To define the sensor map p_x that models the information content of a 1-pixel camera sensor, we first define p_x pointwise, as in (Q1) above. In this case, the physical sensor is identical to the virtual sensor corresponding to the information content of the camera "signal." Next, we wish to define the behavior of the virtual sensor on sets S in the world. To model the information obtained in "applying" p_x to a set, we define a sensor map P_x on sets of M. P_x is the composition of map and Accumulate. More precisely:

$$P_x = \mathtt{map}(p_x) \circ \mathtt{Accumulate}.$$

We observe that map is essentially the restriction map, that restricts p_x to a subset S. The set S is analogous to the notion of *support* that is common in mathematics and computer vision. Accumulate, we have seen, is simply integration, and hence $P_x(S)$ is simply integration of the function p_x over its region of support:

$$P_x(S) = \int_S p_x.$$

If we restrict the sets S to be open, then all the open sets of M form a lattice. In mathematics, a function such as $\mathtt{map}(p_x)$ that assigns, to each open set S, the restriction map $p_{x|_S}$, is called a *sheaf*. The Accumulate function is a kind of *evaluation* on the sheaf. Hence, cameras and similar sensors are naturally modeled as sheaves on manifolds. There is a wealth of literature and results on sheaves.

Finally, note that a multi-pixel camera can be modeled as product of many p_x-type maps, i.e., as a product of sheaves.

[33] One technical point is that this sum saturates and Z_7 is not really the ring of integers mod 7.

B.2.2 Example: Odometry Sensor

As a further example, we explore how to model an odometry sensor. Formalizing this example appears somewhat technical, and we present the details in sec. B.2.2. Now, we just give the basic idea. Consider the notion of *forward-projection* as formalized in, for example, [Erd]. In a configuration space C, let R be a set of initial conditions; that is, the initial state of the robot system is known to be some point in R. let us say this initial state takes place at time $t = 0$. A choice of actions, or controls, which we will call θ, determines the evolution of the system for times $t > 0$. θ is a quite arbitrary index and could range over feedback loops and complex strategies (for example); in [LMT, Erd] it is intended to index the "angular direction" of a commanded generalized damper motion (which of course, is a feedback loop). Erdmann defines the *forward projection* $F_\theta(R, t)$ to be the subset of configuration space that is reachable at time t from initial conditions R and under controls θ [Erd]. That is, the forward projection is a bound on the predictive forward-simulation accuracy of the robot planner; it encodes the robot's knowledge of the effect of executing action θ from initial conditions R. If we view the robot as a dynamical system subject to uncertainty, then the forward projection is the set resulting from integrating the differential inclusion that models the system; see [AC] for example.

It is clear that the forward projection (and its suitable restrictions) model the information available in an odometry sensor or by dead-reckoning computation on control history (integration of commanded controls with error bounds). The way a forward projection is "constructed" via the operations of integration and union is fairly clear: first, we integrate for all valid controls in the differential inclusion to obtain a family of trajectories. Next, we take the union of the states in the image of these trajectories. Indeed, this integration and union construction is explicit in [Erd]. As above (Sec. B.2.1), it is possible to represent this operation as a `map` operation composed with an `Accumulate` operation. See below, sec. B.2.2.

We now describe briefly how the odometry sensor may be viewed in our framework. The knowledge available to the system consists in knowing R, t, and θ. Hence, we may view the tuple (R, t, θ) as a "value" $b^\diamond$ in a sensor space $B^\diamond$. $B^\diamond$ in turn is the domain of a sensing map $p^\diamond$, and we define

$$\pi_C p^{\diamond-1}(b^\diamond) = \pi_C p^{\diamond-1}((R, t, \theta)) = F_\theta(R, t). \tag{33}$$

Details of the Odometry Sensor

Suppose we have a robot with perfect velocity control, and we have perfect knowledge of the commanded velocity $\dot{x}(t)$ as a function of time. In short, our system is described by a differential equation

$$\dot{x} = f(x, t); \tag{34}$$

the state x can be found by integrating this system. Now, we suppose that "knowing" our controls can be modeled by a "sensor" v_x^* whose value is $\dot{x}$. Then, a perfect odometry sensor x^* can be constructed by integration. That is, we have

$$p_x(t) = \int_0^t v_x^* = \int_0^t \dot{x}(t)dt.$$

More formally: let T denote time (the non-negative real numbers), and $\Phi : C \times T \to C$ be the flow on C generated by f, i.e., by eq. (34). Let $\phi_{x_0} : T \to C$ be an integral curve of f with initial conditions x_0, so $\phi_{x_0}(t) = \Phi(x_0, t)$. We define $x^*(t) = \phi_{x_0}(t)$.

If we have imperfect control we must replace the differential equation $\dot{x} = f(x, t)$ by a differential inclusion $\dot{x} \in V_\theta(x)$, where $V_\theta(x, t)$ is a set of tangent vectors; for example, it might be a cone of vectors in direction θ. In general, V_θ is the set-valued map that associates to the state x of the system the set of feasible velocities (see [AC]). In this case we can define an approximate odometry sensor by integration and union:

$$x^*(t) \in F_\theta(t) = \bigcup_{v* \in V_\theta} \int_0^t v^*(t)dt.$$

More generally, following [Erd], we define the forward projection $F_\theta(R,t)$ as above (sec. B.2.2). Then we parameterize the flow Φ by the controls v^* to obtain $\Phi_{v^*} : C \times T \to C$. Then the construction of the forward projection via integration and union is clear:

$$F_\theta(R,t) = \bigcup_{\substack{v* \in V_\theta \\ x_0 \in R}} \Phi_{v^*}(x_0, t).$$

Using eq. (33) to define the sensing map $p^\diamond$ for odometry or dead-reckoning, it is then clear how the sensing map $p^\diamond$ can be constructed via an appropriate `map` and `Accumulate` of the control and position sensor maps v^* and x^*.

B.3 Example: Virtual Sensors

We now work a brief example on the construction of virtual sensors. See fig. 15. We consider three sensing maps: the first, $p^\diamond$, is the sensing map for odometry described above (eq. (33)). The second is the range-finder sensing map p_x from eq. (2) above. As usual we allow $x \in C$ to vary and globalize p_x. Let us call the resulting globalization p^* (instead of p). We do this to avoid name confusion, but it will also serve to remind us that we need to generalize our analysis to similar, yet more sophisticated range sensor p^* described above in construction A.1 above. The third, *virtual* sensor, $\wp$, we will define using $p^\diamond$ and p^*.

We demonstrate how the perceptual equivalence classes for p^* may be constructed using

1. The raw sensor values of the odometry sensor $p^\diamond$,

2. The raw sensor values of the range sensor p^*, and

3. Integration and union of the perceptual equivalence classes from $p^\diamond$.

That is, without knowing an explicit inverse for p^*, but knowing (i) the perceptual equivalence classes induced by $p^\diamond$, and (ii) the values of both $p^\diamond$ and p^*, how can we hypothesize the existence and geometry of the perceptual equivalence classes induced by p^*? Here is our general method of attack. Since we know the perceptual equivalence classes induced by $p^\diamond$, let is call it a *concrete* sensor, and its perceptual equivalence classes *concrete* perceptual equivalence classes. Since we do not know (*a priori*) the perceptual equivalence classes induced by p^*, let us call p^* a *virtual* sensor, and the perceptual equivalence classes induced by p^* *virtual* perceptual equivalence classes. Our question may then be concisely phrased as:

> *How can we hypothesize the existence and geometry of the virtual perceptual equivalence classes from the concrete perceptual equivalence classes we know?*

In general, this problem, which is key to constructing one sensing map from another, can be rather difficult. It is necessary to specify how virtual perceptual equivalence classes are constructed out of (or from) concrete perceptual equivalence classes. That is, it seems necessary for us to tell the robot "What would virtual class $\pi_C p^*(b^*)$ 'look like' through $p^\diamond$?" There is a general answer:

$$p^\diamond(p^{*-1}(b^*));$$

however, this characterization does not directly suggest a computational solution.

However, here are some operational ways of defining virtual perceptual equivalence classes. These techniques require the ability to measure change in the sensing map p^*, and to union "according" to that change. It helps to think of this "controlled union" as a kind of "integration."

Now, for any sensing map, we define its "recognizability gradient:"

Definition B.1 *Let $p : C \times M \to B$ be a differentiable sensing map. Define $\nabla_C p$, the* recognizability gradient *of p to be the projection of the gradient ∇p of p onto TC. Consider the natural tangent space projection $T\pi_C : T(C \times M) \to TC$. Then we have*

$$\nabla_C p = T\pi_C \nabla p.$$

In the case where p is discrete or non-smooth, we will approximate the recognizability gradient by finite differences; we assume this is possible for the maps $p^\diamond$ and p^*. Now, consider the following definitions for virtual perceptual equivalence classes:

1. A virtual perceptual equivalence class can be defined as the union of a set of contiguous concrete perceptual equivalence classes (of $p^\diamond$) with the same recognizability gradient $\nabla_C p^*$.

   (a) More specifically, a virtual perceptual equivalence class can be defined as the union of a set of contiguous concrete perceptual equivalence classes (of $p^\diamond$) with the *zero* recognizability gradient ($\nabla_C p^* = 0$). This implies that all these (concrete) classes have the same value with respect to p^*.
   This construction allows us to use odometry to "trace out" regions of constant range-finder values. These regions will be (eg) strips parallel to linear walls at a constant distance. The width of the strips will be determined by the resolution of the range-finder's distance measurement.

2. Consider a "sequence" of concrete perceptual equivalence classes (of $p^\diamond$)—eg, a linear ordering along some path in C. Suppose as we traverse this sequence, we move in the direction of the recognizability gradient $\nabla_C p^*$. We can define a virtual perceptual equivalence class that is the *limit* of this process.

   (a) More specifically, we have:

   Construction B.2 (*) *Given a sequence of concrete perceptual equivalence classes $S = \{S_1, \ldots, S_i, \ldots\}$ where $S_i = \pi_C p^{\diamond -1}(b_i^\diamond)$ for some $b_i^\diamond \in B^\diamond$. Let I^- denote the half-open unit interval $[0, 1)$, and consider any smooth path $\phi : I^- \to C$ such that $p^\diamond(\phi(I^-)) = S$. Suppose further that at all times t in I^- we have the path ϕ pointing in the direction of the recognizability gradient $\nabla_C p^*$:*

   $$\nabla_C \, p^*(\phi(t)) = \dot{\phi}(t). \tag{35}$$

   Then we may define a virtual *perceptual equivalence class $\overline{S}$ which is the* limit *of the sequence S:*

   $$\overline{S} = \lim_{\nabla_C p^*} S_i, \tag{36}$$

   and $\overline{S}$ contains $\phi(1)$.

Now, let $b^* \in B^*$, the sensor space of p^* (recall section 2 and construction A.1, above). It is now clear that the perceptual equivalence classes of p^* can be defined using the "zero recognizability gradient" construction (1a, above). Specifically:

- A virtual perceptual equivalence class $\pi_C p^*(b^*)$ can be defined as the union of a set of contiguous concrete perceptual equivalence classes $\{ \pi_C p^{\diamond -1}(b_i^\diamond) \}$ of $p^\diamond$, such that all these (concrete) classes have the same value b^* under p^*.

Perhaps the most interesting virtual sensors we can construct entail no additional raw sensors, but only computation and history. Consider a "virtual wall sensor." Suppose that x is a configuration in C, and a point y on M lies on a wall that has absolute[34] orientation θ, length l, and distance d. It might be convenient to have a "wall recognition sensor" that would tell us about the wall at y when we are at x. We imagine this special *virtual* sensor as a new sensing function $\wp$. In the case of the particular x and y above, we want $\wp$ to return the "value" $\mathtt{wall}(\theta, l, d)$. Think of this "value" as a "symbolic expression" whose "meaning" is that there is a wall, containing y, with absolute orientation θ, length l, and distance d. We might imagine defining $\wp$ as follows.

$$\begin{aligned} \wp \ : \ C \times M \ &\rightarrow \ B_w \\ (x, y) \ &\mapsto \ \begin{cases} \mathtt{wall}(\theta, l, d) & \text{if there is a wall at } y\text{, and } y \text{ is visible from } x; \\ \emptyset & \text{otherwise.} \end{cases} \end{aligned} \tag{37}$$

Note that all we have done is specify what $\wp$ should do; that is, we have written down a specification for a sensor we might like to have. Eq. (37) gives us no clue how to implement this specification using computations on our other sensors. What to do in general with such a specification eludes us. In this case, however, the answer is clear:

Consider p^*, the range sensor, as a concrete sensor, and view the perceptual equivalence classes induced by p^* as the concrete perceptual equivalence classes. We will define the virtual sensor $\wp$ relative to the concrete sensor p^*.

The non-trivial perceptual equivalence classes induced by $\wp$ are generated by sets of pairs (x, y) in $C \times M$, such that there is a wall at y and it is visible from x. Each of these virtual perceptual equivalence classes is simply constructed as the union of a set of contiguous concrete perceptual equivalence classes (of p^*) with the same recognizability gradient $\nabla_C p^*$. That is, each concrete perceptual equivalence class $\pi_C p^{*-1}(b^*)$ may be envisioned as a "strip" parallel to a linear wall. A set of these strips with the same recognizability gradient is a set of perceptual equivalence classes all "caused" by sensing the same wall. These strips have the same recognizability gradient $\nabla_C p^* = const$, since, in any one strip, moving along, away, or toward the wall will cause the same changes (respectively) in the sensing map p^*. This is just a way of saying that making incremental motions in C results in the same observed changes in the sensing map p^*; i.e., these "strips" $\pi_C p^{*-1}(b^*)$ have the same recognizability gradient. Hence, they may be unioned into a larger set, namely the class of configurations that can "see" the same wall. For $b_w \in B_w$, this set $\pi_C \wp^{-1}(b_w)$ is defined to be the union of all contiguous strips of the form $\pi_C p^{*-1}(b_i^*)$ with a fixed, constant recognizability gradient $\nabla_C p^*$.

One advantage of this formulation may be that it defines a set of experiments—incremental motions in C that the robot may perform to try to recognize a virtual perceptual equivalence class such as $\pi_C \wp^{-1}(b_w)$. That is, this construction of virtual perceptual equivalence classes explains how the observed sensor values should change with respect to incremental motions the robot can perform. By performing these actions and observing its sensors, the robot may verify hypotheses about the existence and geometry of the virtual perceptual equivalence classes it need recognize in order to perform a task. Specifically, we may regard the sensors $p^\diamond$ and p^* as "concrete" sensors, relative to the wall sensor $\wp$. The resulting "concrete" perceptual equivalence classes are not likely to be human-readable in general. More likely, a robot task might be specified in terms of walls, corridors, and hallways (for example: "Go down the hall, take the second right, and follow the

[34] We could also define the orientation and distance to be relative to x, but the construction is a bit more complicated.

wall as it curves around until you reach the atrium.") These are the virtual perceptual equivalence classes with respect to which a human might wish to express a task. But how can the robot recognize these classes, using the sensors and computation it carries on-board? Instead of trying to provide the robot with "perfect sensors" such as a "wall oracle", a "hallway oracle", etc., instead, we can tell it how to recognize virtual perceptual equivalence classes from concrete ones. Using this method, the robot can then perform experiments to verify which virtual perceptual equivalence classes it is in. This is the process we call ***task-directed sensing.***

The construction above defines a somewhat peculiar virtual perceptual equivalence class–the set of configurations that can see a wall. Note that we can also define the more intuitive set, the class of configurations that are "against the wall." This may be done as follows. We use construction B.2 above. We define the virtual perceptual equivalence class

$$\overline{S} = \pi_C \wp^{-1}\Big(\texttt{on-wall}(\theta, l, d)\Big) \tag{38}$$

as the *limit* of a sequence of strips $\pi_C p^{*-1}(b_i^*)$, such that the recognizability gradient $\nabla_C p^*$ decreases as the sequence is traversed. This decrease, of course, corresponds to the range-finder's distance readings going down as we get close to the wall. We simply mimic the construction B.2 above. The basic idea of this construction is easy. Recall that, for $b_w \in B_w$, the set $\pi_C \wp^{-1}(b_w)$ is defined to be the union of all contiguous strips $\pi_C p^{*-1}(b_i^*)$ with constant recognizability gradient $\nabla_C p^*$. $\pi_C \wp^{-1}(b_w)$ is the set of configurations that can see the wall specified by $b_w = \texttt{wall}(\theta, l, d)$. This entire region has a non-zero constant recognizability gradient $\nabla_C p^*$. We imagine this constant gradient inducing a flow on the concrete perceptual equivalence classes $\{\, \pi_C p^{*-1}(b_i^*) \,\}$ constituting the virtual perceptual equivalence class $\pi_C \wp^{-1}(b_w)$. We follow this flow to its limit, and that limit is the set of configurations that touch the wall b_w.

We now give the formal construction. One minor point is we must flip the sign of the recognizability gradient, since we want to converge *to* the wall, not away from it, and the distance values, of course, decrease to zero as we approach the wall.

Construction B.3 *Suppose we are given a sequence of concrete perceptual equivalence classes $S = \{\, S_1, \ldots, S_i, \ldots \}$ where $S_i = \pi_C p^{*-1}(b_i^*)$ for some $b_i^* \in B^*$. Let I^- denote the half-open unit interval $[0, 1)$, and consider any smooth path $\phi : I^- \to C$ such that $p^*(\phi(I^-)) = S$. Suppose further that at all times t in I^- we have the path ϕ pointing in the direction of the recognizability gradient $-\nabla_C\, p^*$:*

$$-\nabla_C p^*(\phi(t)) = \dot{\phi}(t). \tag{39}$$

Then we may define a virtual *perceptual equivalence class $\overline{S}$, corresponding to eq. (38) which is the* limit *of the sequence S:*

$$\overline{S} = \lim_{-\nabla_C p^*} S_i, \tag{40}$$

and $\overline{S}$ contains $\phi(1)$.

Finally, we note that if we replace our range sensor with the finer sensing map described above in construction A.1, that our analysis and definitions may be suitably generalized. For reasons of space, we feel we must leave details to the interested reader.

C Discussion: Effects of Sensor Noise

We now take a step back, and address a concern that arose in section 4.1 when we declared our example range sensor to be a perfect one; i.e. no noise or uncertainty was modeled. It is clear that coping with uncertainty is critical, and in fact all of the constructions we describe work for sensor models with uncertainty. We explore how this is done here.

Sensor noise can be modeled by passing to set-valued functions. That is, we consider $p_\phi(y)$ to be a subset of B, representing a bound on all possible sensor values. In general, this can destroy the transitivity of perceptual equivalence classes which are defined in terms of inverse images of points in the sensor space B under sensing maps as in, for example, def. 4.2 (above). However, the reflexivity and symmetry are retained. Hence a definition of a perceptual equivalence class becomes analogous to an "adjacency" relation, and its transitive closure analogous to "connectedness". This issue is subtle. We now investigate it carefully.

C.1 Noise as a Set-valued World-Sensing Map

First, we show how sensor noise can destroy transitivity. To do so we develop a world-sensing bundle which models an LMT-like position sensor (with noise) (see [LMT]).[35] Most of the points in this section (C.1) are either definitional or quite clear from the development in [LMT,Erd], but since they are important, we include the discussion for completeness.

Let C, and M both be the plane $\Re^2$. Define a perfect position sensor to be the raw sensing map $p^* : C \times M \to \Re^2$ that sends (ϕ, y) to ϕ. We think of this map as "sensing" the configuration ϕ of the system.

Now, let us introduce noise into the model. Under an LMT-like model of error, we are only guaranteed that the image of ϕ under p^* will lie "nearby", eg in some ball of radius ϵ about ϕ. Hence we have $p^*(\phi) \in B_\epsilon(\phi)$. We can chose to model the noise sensor p^* as a nondeterministic sensor, or as a set-valued function. Let us choose the latter. We make the image of p^* be the powerset of $\Re^2$. The value of p^* represents a bound on all possible sensor values. We know by inspection that the value of p^* will in fact always be a disc in the plane. Suppose we try to generalize our notion of perceptual equivalence to set-valued maps. Consider the following

Idea C.1 ($*$) *We could define the two configurations ϕ and $\phi' \in C$ to be* perceptually equivalent *if there exist y and $y' \in M$ such that*

$$p^*_\phi(y) = p^*_{\phi'}(y'). \tag{41}$$

This definition is clearly inadequate, since the value of p^* is always a disc of radius ϵ, and these two discs are identical only when $\phi = \phi'$. Suppose we try:

Idea C.2 ($*$) *Modify eq. (41) in idea C.1 to be:*

$$p^*_\phi(y) \cap p^*_{\phi'}(y') \neq \emptyset. \tag{42}$$

This is better, since it means that ϕ can "look like" ϕ' when they are confusable[36]–that is, when there is some possibility that the sensor p^* will return the same position value from ϕ as from ϕ'. The problem with declaring ϕ and ϕ' to be "perceptually equivalent" is that the relationship we have defined is not an equivalence relation, because it it not transitive: see fig. 17.

The relationship we obtain is reflexive and symmetric, but not transitive. Hence it is analogous to the "adjacency" relationship in (eg) graph theory or topology. One idea is to construct an equivalence relation by taking the transitive closure. The motivation for this is that the transitive closure of "adjacent" is "connected," which is a useful notion. However, in our example of p^*, this results in identifying the entire plane into one equivalence class.

[35] This example arose in discussions with M. Erdmann, and we are grateful to him for his suggestions.

[36] See [Buc, LMT, Erd, Don], where the concept of "confusability" was formalized in an LMT context.

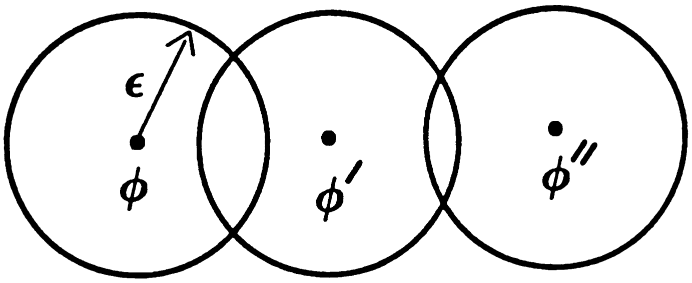

Figure 17: Three configurations with ϵ-balls around them. Under the definition in idea C.2, ϕ is "perceptually equivalent" to ϕ' and ϕ' is "perceptually equivalent" to ϕ'', but ϕ is *not* "perceptually equivalent" to ϕ''. Hence the relationship "perceptually equivalence' is reflexive and symmetric, but not transitive.

One could try to get around this problem by modeling sensor noise by a non-deterministic sensing map. However, the two constructions are basically equivalent. We can see this as follows. We notice that we can restrict to domain of p^* to obtain

$$p^* : C \times M \to \{\, \epsilon\text{-balls in the plane}\,\}.$$

The set of all ϵ-balls is parameterized by their centers, hence we can use $\Re^2$ as the domain of p^*. Now, there are two interpretations possible for p^*. The first is the same as the perfect sensor, since it sends ϕ to the ball centered on ϕ. A second interpretation is as a non-deterministic map that takes ϕ to some sensed value $s \in B_\epsilon(\phi)$. Now, the set of configurations consistent with s (i.e., that could have given rise to the sensed value s) is the ball $B_\epsilon(s)$ about s. Hence a non-deterministic sensor is a map $\phi \mapsto s$; but the set of consistent interpretations of s is a ϵ-ball about s. Hence the non-deterministic sensor may be naturally viewed as a set-valued map $\phi \mapsto B_\epsilon(s)$.

C.2 Even Perfect Sensors Can Break Transitivity

We have seen above how the introduction of sensor noise into our model can deprive the relation that the literature terms "perceptual equivalence" of its transitivity. Hence, one may conclude than in the presence of noise, this relation need not, in fact, be an equivalence relation at all.

It may seem that this effect is due to noise. But, as we shall now show, even perfect, noise-free, deterministic sensors can result in non-transitivity. We now construct such an example. To begin, we define a relation on configurations, denoted $\underset{p}{\sim}$. The relation $\underset{p}{\sim}$ captures the intuitive notion of "confusability", which may be a better name than the "perceptual equivalence" whose transitivity we are about to break. Compare Buckley's work [Buc] on the "confusable" relation, which he explored in the LMT framework.

Definition C.3 *Suppose we have a world-sensing bundle, $p : C \times M \to B$. For two configurations ϕ and $\phi' \in C$, we say ϕ is* confusable *with ϕ' (written $\phi \underset{p}{\sim} \phi'$) when there exist two points y and $y' \in M$ such that*

$$p_\phi(y) = p_{\phi'}(y'). \tag{43}$$

Claim C.4 *The "confusable" relation $\underset{p}{\sim}$ is reflexive and symmetric.* $\Box$

Now, our problems start to arise with the following:

Idea C.5 (*) *Let us define the "perceptual equivalence" using the "confusable" relation* $\underset{p}{\sim}$.

The problem being that $\underset{p}{\sim}$ need not be transitive. In fact:

Proposition C.6 *There exists a world-sensing bundle,* $p : C \times M \to B$ *such that the relation* $\underset{p}{\sim}$ *is not transitive. Furthermore,* p *is noise-free, i.e., deterministic and point-valued.*

Proof: We construct p. Let C, M, and B be $\Re^2$. Let $\phi \in C$ be arbitrary, and B_ϕ be a ball of radius ϵ centered at ϕ. Define $f_\phi : \Re^2 \to B_\phi$ to be a diffeomorphism (the obvious contraction) of the plane onto B_ϕ. Now, the ball B_ϕ includes naturally in the plane as $B_\phi \xrightarrow{\iota} \Re^2$. So we define $p_\phi = \iota \circ f_\phi$. That is,

$$\begin{array}{rccl} p_\phi & : & M & \to & B \\ & & y & \mapsto & f_\phi(y). \end{array} \tag{44}$$

In particular, for $\phi' \in C$, we have the analogous $p_{\phi'} : M \to B$, and obviously $p_\phi(M) = B_\phi$ and $p_{\phi'}(M) = B_{\phi'}$.

We now ask, when does eq. (43) hold? Exactly when $B_\phi \cap B_{\phi'} \neq \emptyset$, i.e., when the distance between ϕ and ϕ' is less than 2ϵ. So in this case, the relation $\underset{p}{\sim}$ is not transitive. □

We notice that the key to this construction is the fact that we could encode "hidden state" into the sensor–that it, the "interrogation" point y that "causes" the sensor value. It is the ability to encode hidden state that makes our sensor model powerful enough to encode non-transitivity. Essentially, we used the hidden state (y) to encode the sensor non-determinism within a deterministic model.

C.3 Perceptual Aliases

Given the discussion above, the situation may look a bit grim for a calculus of the observable based on perceptual equivalence. However, there is a solution, which we give now. We show how noise can be modeled within our framework, while preserving the structure and flavor of all we have developed. The basic idea behind this construction is to turn a weakness into an advantage, and exploit the ability of our sensor model to encode non-determinism (via hidden state) within a deterministic model.

As an example, consider once more our range finding sensor, as described in section 4.1 above. A "picture" of this sensor model is shown in fig. 18. The sensor configuration is the point $\phi \in C$. In particular, $\phi = (x, \theta)$ as shown. The point $y \in M$ is on a wall. For a perfect sensor, we have $p(\phi, y) = \lfloor d \rfloor$ where d is the distance from x to y. However, for a noisy sensor, we might have the following sensor model. There is some error, say ϵ, which lies in an interval U. This error varies non-deterministically. The error causes the sensor to return only approximate information, and sometimes to return $\emptyset$. We can think of this error "signal" ϵ as fluctuating non-deterministically, and "corrupting" the perfect sensor's readings. This results in an imperfect, or noisy sensor, which we might call p^*. See fig. 18. The effect of p^*, when applied to the point y shown, is to return some distance in the interval $[d - \epsilon_{max}, d + \epsilon_{max}]$ (or $\emptyset$). Now, for a sensor with infinite resolution (no discretization), we would have $p(\phi, y) = d$ and

$$p^*(\phi, y) = p(\phi, y) + e(\epsilon) = d + e(\epsilon).$$

However, for a sensor with limited resolution, the situation is more complicated. For such a sensor p^*, (for the same point y as above):

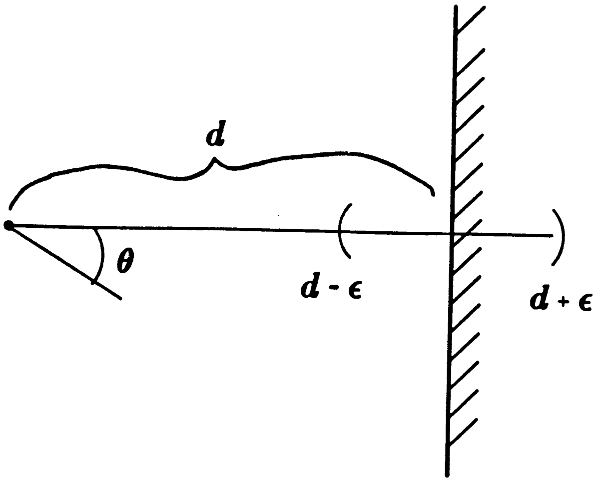

Figure 18: **The robot is at point x, with the range-finder aimed in direction θ. A wall at point y is at distance d from the robot; due to uncertainty in range-finding sensor, the range value sensed is only guaranteed to be inside $(d-\epsilon, d+\epsilon)$.**

$$p^*(\phi, y) = \begin{cases} \emptyset & \text{if } |\epsilon| > \epsilon_{max}\ ; \\ \lfloor d + e(\epsilon) \rfloor & \text{otherwise.} \end{cases} \tag{45}$$

In eq. (45), we assume that $e(\epsilon)$ is some simple function of the error ϵ. In this example, we will assume e is the identity. All our results go through when e is merely injective.

Our idea is to model the "error space" U as part of the world. We regard the "new world" M' as incorporating both M and the error space U, so $M' = M \times U$. We define a new sensing map $\mathbf{p} : C \times M' \to B$, that is,

$$\mathbf{p} \;:\; C \times M \times U \;\to\; B \tag{46}$$

where $\mathbf{p}(\phi, y, \epsilon)$ is given by eq. (45). That is, the state of the world contains U, which determines the error in the sensor. This process is quite general, and extends to almost any sensor and any model of noise. We justify it as follows. In the real world, the sensor values are subject to a host of unmodeled effects. Imagine we now choose to model all these effects. For example, in our range-finder example, we now choose to model the color of the surface at y, the material, the temperature of the air, and so forth. All of these parameters form a state space which we may term U. To model them, we extend our state space from $C \times M$ to $C \times M \times U$—to include all the error parameters. Hence, a geometric point $y \in M$ together with the "physical" parameter $\epsilon \in U$—which encodes the physical characteristics of the world that cause variation in the sensor—together, y and ϵ determine the effect of the world at point y on the sensor value. This simply says that we regard ϵ as part of the state of the world. That is, we construct the sensing map $\mathbf{p}$ to answer question (Q1) (sec. B), with M' substituted for M. By viewing U as "hidden state," we can hence hide a non-deterministic sensing model inside a deterministic sensor map.

Now of course, in general, this kind of modeling of "physical" error parameters will not be possible—we do not, in general, have adequate knowledge of the error characteristics to construct an accurate model of $\mathbf{p}$ and U. But, in these cases, we *can* make a worst-case assumption about the error. We note that for a purely non-deterministic, adversarial error model (see [E90] for a characterization of different error types, including non-deterministic, adversarial, probabilistic, etc), it suffices to have the dimension of U and B be the same. We call U the *hidden state* of the world. We assume that the error parameter $\epsilon \in U$ varies in some uncontrolled manner, and perturbs the sensor readings we obtain. We now see the following:

Claim C.7 *Sensor noise that can be modeled by a non-deterministic sensor map $p^* : C \times M \to B$*

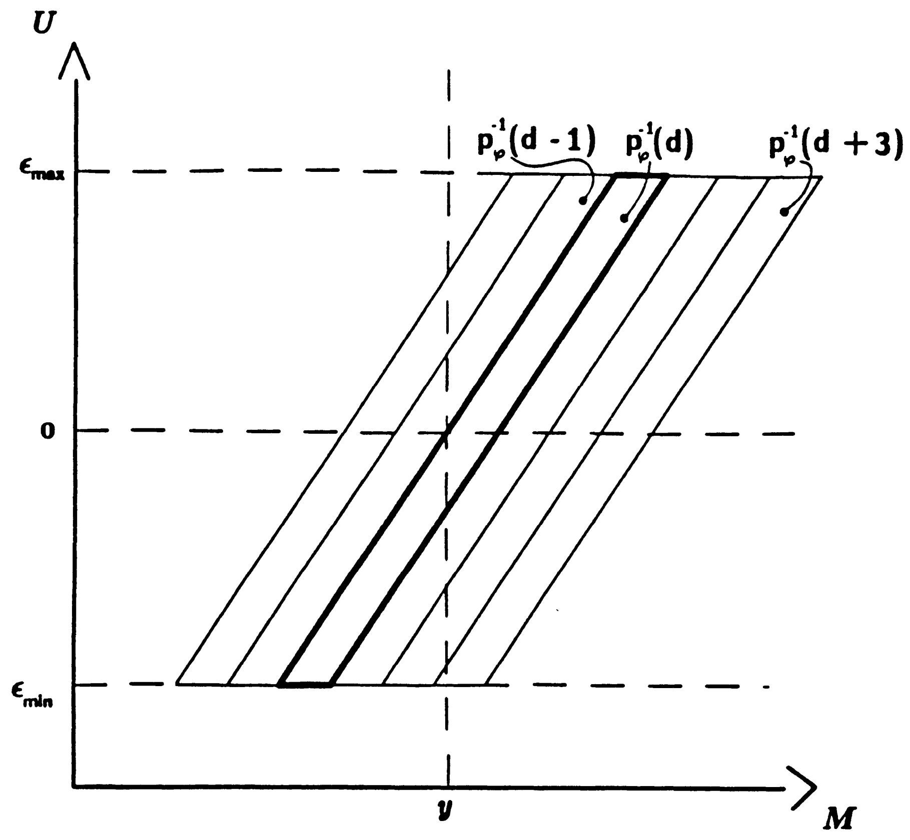

Figure 19: Uncertainty, U, is graphed against the world, M, and the ϵ-perceptual equivalence classes for a simple range-finder are shown. See def. 47.

---

with geometric error bounds, can be converted into a deterministic sensor map $\mathbf{p}$ *with hidden state* U. *It suffices to have* $\dim U = \dim B$.

Now let us return to our example of p^* and $\mathbf{p}$ in eqs. (46) and (45).

Definition C.8 *Suppose we curry a sensing map with hidden state* $\mathbf{p}$ *(eq. (46)) with* $\phi \in C$, *to obtain*

$$\mathbf{p}_\phi : M \times U \rightarrow B$$

so that $\mathbf{p}_\phi(y, \epsilon) = \mathbf{p}(\phi, y, \epsilon)$. *Then at the fixed configuration* ϕ, *the* ϵ-perceptual equivalence classes *are classified by sets of the form*

$$\mathbf{p}_\phi^{-1}(b), \tag{47}$$

for a sensor value b *in* B. *That is, each* ϵ*-perceptual equivalence class is a cell in the arrangement generated by sets of the form (47).*

Once we introduce sensor noise, we obtain an augmented framework in which the sensing maps have hidden state. This hidden state models the non-determinism of sensor noise and error. Essentially the ϵ-perceptual equivalence classes now serve in place of the perceptual equivalence classes we had before. To get some intuition for the structure of an ϵ-perceptual equivalence class, consider our simple example of a range-finder. Here, U would simply be some real interval. Suppose y is a point at distance d from x, as shown in fig. 18. The ϵ-perceptual equivalence class for sensor value

$\lfloor d \rfloor$ would contain the point $(y, 0) \in M \times U$. This is simply because $\mathbf{p}_\phi(y, 0) = \lfloor d + e(0) \rfloor = \lfloor d \rfloor$. But the ϵ-perceptual equivalence class $\mathbf{p}_\phi^{-1}(\lfloor d \rfloor)$ would also contain all points (y', ϵ), such that $\mathbf{p}_\phi(y', \epsilon) = \lfloor d \rfloor$, i.e., $\lfloor \|y' - x\| + \epsilon \rfloor = \lfloor d \rfloor$.

Now, in our new model, the world is $M \times U$, i.e., it contains the hidden state U. The ϵ-perceptual equivalence class $\mathbf{p}_\phi^{-1}(b)$ consists of all "points" (y, ϵ) in the world that can result in the sensor value b. For example, see fig. 19. In this case, the ϵ-perceptual equivalence class $\mathbf{p}_\phi^{-1}(b)$ (for an integer b) is a diagonal strip in $M \times U$. The width of the strip is the sensor resolution.

On the other hand, given a point y in M (or more generally, a feature at y), we may wish to ask, *What sensor values can y generate?* The answer is as follows; see fig. 19. We define the *perceptual alias set of y* to be all ϵ-perceptual equivalence classes that "project to cover y" in M. These are the (y', ϵ)-"points" in the world that are "confusable" with y. We define the *semblance* of a point y in M to be all the ways that y can appear as projected through the sensor, when noise is taken into account. More formally: we begin with the "local" definitions (for a fixed configuration in C). By a slight abuse of notation, we overload the projection operator π_M to denote projection from any product of M onto M: below, $\pi_M : M \times U \to M$.

Definition C.9 *For a sensing function* $\mathbf{p}$, *a configuration ϕ in C, and a point y in M, the* semblance of y *is the set of the sensor values whose ϵ-perceptual equivalence classes project to cover y. The semblance of y is denoted* $\mathrm{sem}(y)$, *and*

$$\mathrm{sem}(y) = \{\, b \in B \;\big|\; \mathbf{p}_\phi^{-1}(b) \cap \pi_M^{-1}(y) \neq \emptyset \,\}.$$

Note the lower case "s" in $\mathrm{sem}(\cdot)$. Later, $\mathrm{Sem}(\cdot)$ will be used to denote the global version of semblance. Next, we can give a rigorous characterization of perceptual aliasing:

Definition C.10 *For a sensing function* $\mathbf{p}$, *a configuration ϕ in C, and a point y in M, the* perceptual alias set of y *consists of all ϵ-perceptual equivalence classes whose projection onto M covers y. That is:*

$$\{\, \mathbf{p}_\phi^{-1}(b) \;\big|\; b \in \mathrm{sem}(y) \,\}.$$

Hence, the perceptual alias set of y consists of the ϵ-perceptual equivalence classes of the semblance of y.

The characterization of ϵ-perceptual equivalence classes in fig. 19 is in fact representative for higher dimensions as well. Thus, the ϵ-perceptual equivalence class are "well-behaved." More specifically, It is simple to show:

Claim C.11 *Suppose the error function e is injective. Then for two sensor values b_1 and b_2 in B, $\mathbf{p}_\phi^{-1}(b_1) \cap \mathbf{p}_\phi^{-1}(b_2) \neq \emptyset$ if, and only if, $b_1 = b_2$.* □

Corollary C.12 *The ϵ-perceptual equivalence classes are disjoint if the error function is injective.*

To give some intuition about the structure of the ϵ-perceptual equivalence classes, consider the following. Imagine an LMT-style position sensor, with error. We convert this sensor to a deterministic sensor (such as $\mathbf{p}$) with hidden state; in this case, U will be a ball (representing an error ball). The error value $\epsilon \in U$ is used to perturb the position $\phi \in C$ to obtain the sensor value $\phi + e(\epsilon)$. Let us assume for a moment that $M = C = B = \Re^n$, so of course, U is the n-dimensional ball, and the error function is injective. Then, the ϵ-perceptual equivalence classes are disjoint, each ϵ-perceptual equivalence class is an n-dimensional sheet in $M \times U$. (If e is linear, the sheets are planes, etc).

C.3.1 Globalizing Semblances and Perceptual Aliases

Semblences and perceptual aliases may be globalized in a straightforward manner to a world sensing bundle, $\mathbf{p}$, with domain $C \times M \times U$. We now show how.

To extend our framework to model noisy sensors, we paralleled our initial development of perceptual equivalence: in this we defined semblance and perceptual aliasing for a *fixed* configuration ϕ in C, and in terms of projections onto M. This, we did for the sake of intuition. We now globalize our constructions by allowing ϕ to vary along C; that is, we allow the robot to move and aim its sensors. We obtain the following. We first extend definition 4.4 from C to $C \times U$. Following [Don], we will call $C \times U$ *generalized configuration space.*

As usual suppose $\mathbf{p}$ is the world sensing bundle with domain $C \times M \times U$. Then:

Definition C.13 *Fix a sensor value $b \in B$. We call a subset S of the generalized configuration space $C \times U$ a* configuration space ϵ-perceptual equivalence class *(or simply* C-Perceptual ϵ-Equivalence Class*), when, for each generalized configuration ϕ in S, there exists some point y_ϕ in the world M such that $\mathbf{p}_\phi(y_\phi) = b$.*

Now, let π_{CU} denote the natural projection of $C \times M \times U$ onto generalized configuration space $C \times U$. Then it can be shown that:

Lemma C.14 *The maximal C-Perceptual ϵ-Equivalence Classes under definition C.13 are the cells in the arrangement generated by the sets*

$$\{\, \pi_{CU}\mathbf{p}^{-1}(b) \,\}$$

for $b \in B$.

We now globalize the concepts of semblance and perceptual alias analogously: below, $\pi_M : C \times M \times U \to M$.

Definition C.15 *For a sensing function* $\mathbf{p}$ *and a point y in M, the* Global Semblance of y *is the set of the sensor values whose configuration space ϵ-perceptual equivalence classes project to cover y. The global semblance of y is denoted* $\mathrm{Sem}(y)$*, and*

$$\mathrm{Sem}(y) = \{\, b \in B \;\Big|\; \mathbf{p}^{-1}(b) \cap \pi_M^{-1}(y) \neq \emptyset \,\}.$$

Definition C.16 *For a sensing function* $\mathbf{p}$ *and a point y in M, the* configuration space perceptual alias set of y *consists of all configuration space ϵ-perceptual equivalence classes whose projection onto M covers y. That is:*

$$\{\, \pi_{CU}\mathbf{p}^{-1}(b) \;\Big|\; b \in \mathrm{Sem}(y) \,\}.$$

C.4 Discussion on Noise, Error, and Perceptual Aliasing

What we have essentially done in our construction is graph the state space (M, above) against the error U. We then proceeded to reason about the resulting enlarged state space $M \times U$. It is not surprising that perceptual equivalence classes formed in the product state space were the natural generalizations of perceptual equivalence to the case with noise. Our construction, of course is not new. In particular, we recall the work by Brooks [Brooks 82], in which the uncertainty is graphed against state. Recall also the work by Donald [Don, D, Don90] on compliant motion planning in the presence of model error. In this work, we introduced an "error" space like U and formed a "generalized configuration space" which was the product of the state space and the error space. Many others have used this construction.

Here are some significant aspects of our use of the construction:

1. In most previous work, the uncertainty space U is known ahead of time. In our case it is completely unknown.

2. Suppose we let U represent the actual unknown parameters in the world that corrupt and influence our sensor values (see sec. C.3). We will call this the "concrete" error space in that parameters in this space will determine adumbrations of the sensors' interaction with the physical world, and thereby the sensor values that result from interrogating features in that world. So, for a sonar sensor, U might include temperature, surface material, windspeed, etc. As another example, consider a flux-gate compass. Suppose the robot trusts its odometry,[37] and detects the presence of some magnetic disturbance in the room. But, the disturbance might vary in intensity and direction as the robot moves. In this case U could encode the flux of an unknown, interfering magnetic field. This field would vary spatially. As the robot moved, it could, perhaps, deduce the characteristics of the interfering field–that is, the structure of U. For example, the field might be stronger near the Lisp-Machine in the corner of the lab. By discovering the correlation in field intensity change with spatial locality through experiments (varying its controls) the robot could determine the sensor flow[38] on $C \times U$. At any rate, in the same way that the robots in sec. 7 could reconstruct to infer the structure of $C \times M$, a robot could B-encode the RR-graph of the world sensing bundle, and then use or hypothesize local sections to reconstruct the structure of U or the structure of $C \times U$.

3. The "concrete" physical error space may be hard to model or estimate. But for the purposes of this framework, we can approximate it by using an error space that merely has the same dimension as B. This models a pure, non-deterministic sensor noise, with geometric bounds. We call it the "non-deterministic error space U."

   Note however, that the "concrete" error space will typically have a much higher dimension that the non-deterministic error space. However, the non-deterministic error space is always a projection of the concrete error space. By this, we mean that every point in the concrete error space determines a point in the non-deterministic error space, and that it is possible to calculate the sensor value (with noise) using only the latter.

   The problem of reconstructing the concrete error space from the non-deterministic error space is very similar to the problem of reconstructing $C \times M$ from values in B.

For situations where the intransitivity of perceptual equivalence without U makes working in C too difficult, the problem can always be lifted to the enlarged state space $C \times U$, as described in previous sections. However there exists a cost to this solution—the higher dimensional state space—and so, where possible, we prefer to work in C. The research directions highlighted in this section deserve further study.

D History Permits Lifting

We now formalize the sense in which history permits "lifting" to a finer partition. We must stress that the claim D.1 in this section is a claim about a specific example, not a general claim. However, similar constructions are possible in a host of examples. A key research issue is to discover how this claim generalizes and is applicable in arbitrary circumstances.

Assume the robot, environment, $\mathcal{J}$, and $\mathcal{K}'$ are as described above. Let T denote time (we can think of time as a bounded interval of the real line). We've shown the following:

[37] Of course, this might just be one of a number of multiple hypotheses. For more on support of multiple hypotheses, see sec. 7.

[38] *Sensor flow* is defined to be the integral of the recognizability gradient. See sec. B.3.

Suppose that our robot sensing is orientation-free and yields the partition of configuration space C into the recognizable sets $\mathcal{K}'$ as described above. Suppose a motion strategy (such as v) resulting in an (unknown) path

$$\phi : T \to C$$

is executed. Suppose further that we know a corresponding sensor history

$$b_* : T \to \mathcal{K}'$$

and control history

$$v : T \to S^1/4.$$

Then we can construct a lifting

$$\tilde{b} : T \to \mathcal{J}$$

with the following properties:

1. $\tilde{b}$ "projects" to b_* in this sense: Let $\xi : \mathcal{J} \to \mathcal{K}'$ be the forgetful map. Then $b_* = \xi \circ \tilde{b}$.
2. $\tilde{b}$ "lifts" to ϕ in this sense: Let $q : C \to \mathcal{J}$ be the forgetful map. So q is the "global" quotient map induced by the world sensing bundle p (see prop. A.5). Then $\tilde{b} = q \circ \phi$.

Let TM be the phase space (tangent bundle) of M, and let TM' be some quotient space of M. In other words:

Claim D.1 *Suppose, as described above, we are given the mesh of partitions $\mathcal{K}' \hookrightarrow \mathcal{J} \hookrightarrow M$, sensor history $b_* : T \to \mathcal{K}'$, control history $v : T \to TM'$ corresponding to a path $\phi : T \to C$:*

$$\begin{tikzcd}
 & C \\
T \arrow[ur, "\phi"] \arrow[dr, "b_*"'] & \mathcal{J} \arrow[u, "j"'] \\
 & \mathcal{K}' \arrow[u, "j"']
\end{tikzcd} \tag{48}$$

Then without using knowledge of ϕ, we can construct a lifting $\tilde{b} : T \to \mathcal{J}$, which makes the diagram (49) commute:

$$\begin{tikzcd}
 & C \arrow[d, "q"] \\
T \arrow[ur, "\phi"] \arrow[r, "\tilde{b}"] \arrow[dr, "b_*"'] & \mathcal{J} \arrow[d, "\xi"] \\
 & \mathcal{K}'
\end{tikzcd} \tag{49}$$

E Acknowledgements

Many key ideas in this paper arose in discussions with Tomás Lozano-Pérez, Mike Erdmann, and Dan Huttenlocher, and we are very grateful to them for discussions and suggestions, and for their generosity with their time and ideas. We would like to thank Peter Kahn for many valuable discussions, and for helping us to domesticate the methematics. Many thanks to Prakash Panagaden and Allen Back for discussions and advice. Thanks to Maja Mataric for discussing her work and robotics in general with us. Thanks also to John Canny, Devika Subramanian, Alan Christiansen, David Chapman, and Phil Agre for insight and comments on this work.

References

[AC] **Aubin, J.P. and Cellina, A.** *Differential Inclusions,* Springer-Verlag: New York (1984).

[BBD+] **Baillieul, Brockett, Donald,** ET AL. *Robotics,* Symposia in Applied Mathematics, vol 41. American Mathematical Society Press, Providence (1990).

[BaKu] **Barshan, B. and R. Kuc** *Differentiating Sonar Reflections from Corners and Planes by Employing an Intelligent Sensor,* IEEE PAMI 12 (**6**), pp 560-569 (1990).

[BSD] **Blondell, B., Shollenberger, S. and Donald, B.** *Design of a microprocessor driven generic controller for a mobile robot base,* Cornell CS Dept., Ithaca, NY CU-CS-TR-90-1140 (1990).

[Bri] **Briggs, Amy.** (1989) *An Efficient Algorithm for One-Step Compliant Motion Planning with Uncertainty,* 5^{th} ACM Symp. Comp. Geom., Saarbrucken, BRD

Brooks, R. A. *Symbolic Error Analysis and Robot Planning,* IJRR Vol 1, no. 4, Dec., 29–68 (1982).

Brooks, R. A. *A Robust Layered Control System for a Mobile Robot,* IEEE JRA **RA-2** (1): 14–23. (1986).

Brost, R., "Automatic Grasp Planning in the Presence of Uncertainty", *IEEE ICRA*, San Francisco, 1986.

[BD] **Brown, R. G. and Donald, B.** *A System for Robotic Navigation, Task-Directed Sensing, and Active Segmentation,* Submitted to *IEEE ICRA*, Nice, 1992 (1991).

[Buc] **Buckley, S. J.** *Planning and Teaching Compliant Motion Strategies,* Ph.D. Thesis. MIT, Dept EECS, 1987. Also MIT-AI-TR-936 (1987).

Caine, M., "Chamferless Assembly of Rectangular Parts in Two and Three Dimensions", S.M. thesis, MIT Dept. of Mech. Eng. 1985.

[Can] **Canny, J.F.** *On The Computability of Fine-Motion Plans,* Proc. IEEE ICRA Scottsdale, AZ (1989).

[C] **Christiansen, A.** *Manipulation Planning from Empirical Backprojections,* CMU Dept. C.S. (1991).

[CMM] **Christiansen, A., Mason, M., and Mitchell, T.** *Learning Reliable Manipulation Strategies without Initial Physical Models,* IEEE ICRA (1990).

[CL] **Cox, I., and Leonard, J.** *Temporal Integration of Multiple Sensor Observations for Dynamic World Modeling: A Multiple Hypothesis Approach,* NEC Research Institute; Princeton, NJ (1990).

[CR] **Canny, J., and J. Reif,** "New Lower Bound Techniques for Robot Motion Planning Problems", *FOCS* (1987).

[D] **Donald, B. R.** *A Geometric Approach to Error Detection and Recovery for Robot Motion Planning with Uncertainty,* Artificial Intelligence, **37** (1-3), Dec. pp. 223-271. (1988).

[Don90] **Donald, B. R.**, "Planning Multi-Step Error Detection and Recovery Strategies", *IJRR*, **9** (1), Feb. **1990. pp. 3-60.**

[Don] **Donald, B. R.** *Error Detection and Recovery in Robotics,* Springer-Verlag (New York): Lecture Notes in CS, vol. 336 (1989).

[DJ] **Donald, B. and Jennings, J.** *Sensor Interpretation and Task-Directed Planning Using Perceptual Equivalence Classes,* IEEE ICRA, Sacramento, CA (1991).

[DW] **Durrant-Whyte, H.** *Concerning Uncertain Geometry in Robotics,* Artificial Intelligence, **37** (1-3), Dec. (1986).

[Erd] **Erdmann, M.** *Using Backprojections for Fine Motion Planning with Uncertainty,* IJRR Vol. 5 no. 1 (1986).

[E89] **Erdmann, M.** *Probabilistic Strategies for Robot Tasks,* MIT-AI Lab PhD Thesis, Dept. EECS (1989).

[EM] **Erdmann, M., and M. Mason**, "An Exploration of Sensorless Manipulation", *IEEE International Conference on Robotics and Automation*, San Francisco, April, 1986.

[EMV] **Erdmann, M., Mason, M, and Vanacek G.** *Mechanical Parts Orienting: The Case of a Polyhedron on a Table,* IEEE ICRA (1991).

[FHS] **Friedman, J., J. Hershberger and J. Snoeyink** *Compliant Motion in a Simple Polygon,* 5^{th} ACM Symp. Comp. Geom., Saarbrucken, BRD. (1989).

[GM89] **Goldberg, K., and Mason, M.** *Bayesian Grasping,* IEEE ICRA, 1990 (1989).

[GME] **Goldberg, K., Mason, M., and Erdmann M.** *Generating Stochastic Plans for a Programmable Parts Feeder,* IEEE ICRA (1991).

Grigoryev D. Y., "Complexity of Deciding Tarski Algebra" Jour. Symbolic Computation, special issue on decision algorithms for the theory of real closed fields, (1987).

[HM] **Hager, G., and Mints, M.** *Computational Methods for Task-Directed Sensor Data Fusion and Sensor Planning,* IJRR **(10)** 4 pp. 285-313 (1991).

[Hut] **Hutchinson, S.** *Exploiting Visual Constraints in Robot Motion Planning,* IEEE ICRA (1991).

[JDC] **J. Jennings, B. Donald, and D. Campbell** *Towards Experimental Verification of an Automated Compliant Motion Planner based on a Geometric Theory of Error Detection and Recovery,* Proc. IEEE ICRA, Scotsdale, Arizona (1989).

[KTB] **Kriegman, D., E. Triendl, and T. Binford** *Stereo Vision and Navigation in Buildings for Mobile Robots,* IEEE TRA 5 **(6)** pp 792-803 (1989).

[KS] **Kuc, R. and M. Siegel** *Physically Based Simulation Model for Acoustic Sensor Robot Navigation,* IEEE PAMI 9 **(6)** pp. 766-778 (1987).

[LDWC] **Leonard, J., Durrant-Whyte, H., and Cox, I.** *Dynamic Map Building for an Autonomous Mobile Robot,* Preprint: To appear in IEEE/RSJ Intl. Conf on Intelligent Robot Systems (IROS) (1990).

[LoP] **Losano–Péres, T.** *Spatial Planning: A Configuration Space Approach,* IEEE Trans. on Computers (C–32):108–120 (1983a).

Losano-Péres T., Jones J., Maser, E., O'Donnell, P., Grimson, E., "Handey: A Robot System that Recognizes, Plans, and Manipulates", Proc. IEEE Int. Conf. on Robotics and Automation, 1987.

[Lat] **Latombe, J.-C.** *Robot Motion Planning,* Kluwer (1990).

[LLS] **Latombe, J.C., A. Lasanas, and S. Shekhar** *Motion Planning with Uncertainty: Practical Computation of Non-Maximal Preimages,* IEEE/RSJ Int'l. Workshop on Intelligent Robots and Systems, Sept. 4-6 (1989).

[LMT] **Losano-Péres, T., Mason, M. T., and Taylor, R. H.** *Automatic Synthesis of Fine-Motion Strategies for Robots,* IJRR, Vol 3, no. 1 (1984).

[Maes] **Maes, P.** *Designing Autonomous Agents,* MIT/Elsevier : Amsterdam (1990).

[MC] **Mahadevan, S. and Connell, J.** *Automatic Programming of Behavior-based Robots using Reinforcement Learning.,* IBM T.J.Watson Res. Rept. (1990).

[Ma] **Mason, M.T.** *Compliance and force control for computer controlled manipulators,* IEEE Trans. on Systems, Man and Cybernetics (SMC-11):418-432 (1981).

[Ma2] **Mason, M. T.** *Automatic Planning of Fine Motions: Correctness and Completeness,* 1984 IEEE ICRA, Atlanta Ga. (1984).

[Mat] **Mataric, M.** *A model for distributed mobile robot environment learning and navigation,* S.M. Thesis, MIT AI Lab, Dept. EECS (1990).

[Min] **Minsky, M.** *A framework for representing knowledge,* The Psychology of Computer Vision, Ed. P. Winston, McGraw-Hill, pp 211-77 (1975).

[Mit] **Mitchell, T. M.** *An Approach to Concept Learning,* Stanford Univ., Palo Alto, CA TR STAN-CS–78-711 (1978).

[ME] **Moravec, H., and A. Elfes** *High resolution maps from wide angle sonar,* Proc. IEEE ICRA. pp 116-121 (1985).

[Mor] **Moravec, H.** *Sensor fusion in certainty grids for mobile robots,* in A. Casals, ed., *Sensor Devices and Systems for Robotics.* Springer-Verlag (1989).

[Nat] **Natarajan, B. K.** 1986 (Oct. 27–29, Toronto, Ontario). An Algorithmic Approach to the Automated Design of Parts Orienters. *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pp. 132–142.

[OdD] **O'Donnell, M. and Donald, B.** *On Large Scale Planar Manipulation,* Cornell CS Dept., Ithaca, NY CU-CS-TR 90-1127 (1990).

[RC] **Raibert, M., and J. Craig**, "Hybrid Position/Force Control of Manipulators", *Journal of Dynamic Systems, Measurement, and Control*, No. 102, June, 1981, pp. 126-133.

[RD] **Rees, J. and Donald, B.** *Program Mobile Robots in Scheme,* Submitted to *IEEE ICRA*, Nice, 1992 (1991).

[R] **Rosenschein, S.J.** *Synthesizing Information-Tracking Automata from Environment Descriptions,* Teleos Research TR No. 2 (1989).

Schwarts J. and Yap C. K., "Advances in Robotics," Lawrence Erlbaum associates, Hillside New Jersey, (1986).

[S] **Shafer, S.** *Why Can't We Model the Physical World?,* for the CMU CS Dept. 25^{th} Anniversary (1990).

Simunovic, S. N., "An Information Approach to Parts Mating", Ph.D. thesis, Dept. EE, MIT, 1979.

[Su] **Sutton, R.** *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,* Proc. 7^{th} Intl. Conf. on Machine Learning (1990).

Tarski A., "A Decision Method for Elementary Algebra and Geometry" Univ. of Calif. Press, Berkeley, (1948), second ed. 1951.

Taylor, R. H. *The Synthesis of Manipulator Control Programs from Task-level Specifications,* Stanford A.I. Lab., AIM-282, July (1976).

Ward, B. and McCalla, G. *Error Detection and Recovery in a Dynamic Planning Environment,* AAAI (1983).

[WB] **Whitehead, S. and Ballard, D.** *Active Perception and Reinforcement Learning,* 7^{th} Int'l. Conf. on Machine Learning (1990).

Yap, C., "Algorithmic Motion Planning", in *Advances in Robotics: Volume 1*, edited by J. Schwartz and C. Yap, Lawrence Erlbaum Associates, 1986.