Action Subservient Sensing and Design

Michael Erdmann

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Abstract

This paper outlines a method for automatically designing sensors from the specification of a robot's task, its actions, and its uncertainty in control. The sensors provide precisely the information required by the robot to perform its task, despite uncertainty in sensing and control. The key idea is to generate a strategy for a robot task by using a backchaining planner that assumes perfect sensing while taking careful account of control uncertainty. The resulting plan indirectly specifies a sensor that tells the robot when to execute which action. Although the planner assumes perfect sensing information, the sensor need not actually provide perfect information. Instead, the sensor provides only the information required for the plan to function correctly.

1 Introduction

The goal of robotics is to develop autonomous systems. In order to achieve this goal, we must understand the relationship between action and sensing, and we must be able to convert that understanding into robots capable of operating productively and successfully in an uncertain world.

The nature of the relationship between action and sensing raises some fundamental questions:

- What is the information needed to solve a given manipulation task?
- What tasks can be solved by a given repertoire of operations?
- How sensitive are solutions of tasks to particular assumptions about the world?

Obtaining answers to these questions forms the central research agenda in robotics. We will here focus on the first question. It is a most important question in robotics. Within it are contained the basic issues of task definition, action specification, environment design, and sensing capabilities.

1.1 Repositories of Information

When we ask "what information is required?" some answers seem to appear immediately. Yet, when we look more closely these answers become fuzzy. The answers that appear immediately are of the form: some information arises from sensors, other information is encoded in the mechanism performing the task, some information is encoded in the mechanism's interpretation of the world, some in its predictive ability, and still other information acts subtly through the environment. It is obvious why these answers are fuzzy. They are fuzzy because there is no clear notion of how the information is distributed between these different repositories of information.

1.2 A Bewildering Array of Strategies

As an example, imagine the task of placing a pencil into an electric pencil sharpener in order to sharpen the pencil. There are numerous methods for accomplishing this task. One is to grasp the pencil then move one's hand slowly while carefully looking at the pencil and the sharpener, continuously readjusting one's aim as the pencil comes close to the sharpener's hole. Another is to close one's eyes, place one hand on the pencil sharpener, then with the other hand bring the pencil to the sharpener, and guide it manually into the hole using tactile feedback. A third strategy is to visually memorize the state of the world then move the pencil into the sharpener with no feedback whatsoever. A fourth strategy consists of randomly stabbing at the sharpener with the pencil until the pencil enters the sharpener. A fifth strategy consists of building a linkage of some sort that can insert the pencil into the sharpener from a large set of initial states. The linkage is specially designed as a function of the sharpener's geometry and the possible starting locations of the pencil. A sixth strategy is to dispense with the notions of pencil and sharpener, and instead build a series of tightly coupled sensory feedback loops, implemented on top of hardware that can grasp and move long thin objects. The feedback loops might implement such operations as pencil-seeking, pencil-

1050-4729/93 $3.00 © 1993 IEEE

grasping, and hole-seeking. Numerous other strategies may be constructed as hybrids of these six strategies.

This enormous array of strategies leaves one with a distinct feeling of unease. In particular, the strategies seem to gain their information very differently. Some rely heavily on sensors, others on mechanical devices, and yet others on prediction. This makes it very difficult to answer the question "what information is required to solve the task of putting a pencil into a sharpener?" Indeed, we could complicate the question even more by varying the sizes and shapes of the pencil and the sharpener. One is tempted to dismiss the question as non-sensical, based on the apparent realization that there is no intrinsic measure of required information and even no intrinsic task of the form PUT THE PENCIL IN THE SHARPENER.

1.3 Designing Sensors for Tasks

We will resist this temptation, and instead introduce time as a measure relative to which one can study information requirements. Specifically, one can first seek the information required to accomplish a task as quickly as possible. Relative to this base standard one can then compare other strategies and their information requirements.

2 Related Work

This work is motivated by a long history of work on uncertainty. We will give a brief account here. Early work on uncertainty sought to improve the performance of robot planning and execution systems by developing primitive operations specifically designed to overcome uncertainty. The intent was to retain the structure of high-level planning systems, while using the primitive operations to encapsulate the low-level activity needed to overcome uncertainty. *Guarded moves* consisted of motions that terminated execution when some sensory condition was satisfied. Similarly, *compliant motions* were developed to maintain contact with surfaces. See [Whitney 1977], [Mason 1981], and [Raibert and Craig 1981], among others.

The natural next step was to incorporate these local strategies into global motion planners. Early work considered parameterizing strategies in terms of quantities that could vary with particular problem instantiations. The skeleton strategies of [Lozano-Pérez 1976] and [Taylor 1976] offered a means of relating error estimates to strategy specifications in detail. [Brooks 1982] developed a symbolic algebra system that could be used both to provide error estimates for given operations, as well as to constrain task variables or add new sensing operations in order to guarantee task success.

Later, [Lozano-Pérez, Mason, and Taylor 1984] proposed a planning framework for synthesizing fine-motion strategies in the presence of uncertainty. This framework generates plans by recursively backchaining from the goal. Each backchaining step generates a collection of sets, known as *preimages*, from which entry into the goal is guaranteed, despite sensing and control uncertainty. The preimage framework directly incorporates uncertainty into the planning process. See [Mason 1984], [Erdmann 1986], [Buckley 1987], [Donald 1989], [Canny 1989] and [Latombe 1990] for some further work on preimages.

An important offspring of the LMT preimage planning methodology is Donald's work on Error Detection and Recovery [Donald 1989]. Donald's work moved away from the requirement that a strategy, in order to be considered a legitimate strategy, actually be guaranteed to solve a task in a fixed predetermined number of steps. Building on top of Donald's ideas, [Erdmann 1990] studied active randomization as a primitive strategy for accomplishing robot tasks. [Goldberg 1990] investigated probabilistic strategies for grasping objects and developed a framework for planning optimal orienting strategies relative to various cost functions.

An important idea lies hidden in the work cited above. If we look at the skeleton strategies of Taylor and Lozano-Pérez, and the plan checker of Brooks, we see an underlying design philosophy. Specifically, strategies and environments are seen as interwoven. Given a robot task there are two ways to proceed. One is to develop a strategy for accomplishing the task in the specified environment. Another direction is to redesign the environment so that some simple off-the-shelf strategy is guaranteed to succeed. For instance, one might develop a nominal plan for accomplishing the task under the assumption of no uncertainty. In order to ensure the plan's success in an uncertain world, one introduces a new sensor not originally postulated in the environment. This second direction for dealing with uncertainty has not received as much attention in recent years. An important research question is to determine the conditions under which redesigning the environment is a possible solution. Studying the information requirements of manipulation tasks is one approach to answering this question. In related recent work, [Lazanas and Latombe 1992] explore the idea of designing an environment in order to simplify sensing and planning. They use a preimage planner to navigate a robot between local landmark regions within which sensing is perfect and

outside of which sensing is non-existent. [Canny and Goldberg 1991] and [Donald and Jennings 1992] study the design of task-directed sensing to simplify robot programming. Canny and Goldberg are developing a *RISC* approach to programming robotics, and advocate using simple sensors, such as special light beam sensors. Donald and Jennings study the lattice of sensor values as a function of robot actions. Their robots perform ***constructive recognizability*** experiments to extract relevant information from the environment.

3 Complexity

This section argues for sensor design based on an examination of the complexity of several different approaches to motion planning. We draw our complexity estimates from two domains. The actions in these domains have non-deterministic or probabilistic transitions. One domain consists of discrete state spaces with discretely specified actions. See [Papadimitriou and Tsitsiklis 1987], and [Erdmann 1990] for further details. The other domain consists of continuous spaces with continuous actions, analogous to those in the LMT [Lozano-Pérez, Mason, and Taylor 1984] preimage methodology. See [Natarajan 1986, 1988], [Canny and Reif 1987], [Canny 1988, 1989], and [Donald 1988] for further details.

Table 1 summarizes the basic results. We omit many details. One notices the following pattern: for perfect sensing, planning seems to be easy; for general sensing, planning seems to be intractable; and for sensorless strategies, planning seems to be hard, but not as hard as in the general case.

The results of Table 1 suggest that it makes sense to look at problems in which sensing is perfect. One is reluctant to do so, since perfect sensing is impossible. There is a much more attractive view, however. This view envisions sensing as a design problem. For any task, one can assume perfect sensing in determining the control strategy for accomplishing that task. Having done so, one then builds an imperfect sensor that nonetheless provides the information required for this control strategy to operate.

Sensorless	P	NP	$2^{(nk)^{O(1)}}$
Imperfect Sensing	PSPACE	PSPACE	$2^{2^{O(nk)}}$
Perfect Sensing	P	P	P
	Perfect Discrete; Optimal	Probabilistic Discrete; Optimal	Adversarial Continuous; Guaranteed
	Control Uncertainty		
	Task Space; Strategy Type		

Table 1: A rough complexity picture. In each case the problem is to decide the existence of a k-step strategy for attaining the goal from some initial region, given n obstacle constraints. The right-most column concerns tasks on continuous spaces, with the objective of finding a guaranteed strategy that attains the goal. The other two columns concern tasks on discrete spaces, with the objective of obtaining an optimal strategy. The complexity classes are classifications, except for the two exponentials, which are known upper bounds.

4 A Family of Sensors for the Point-Into-Disk Problem

In this section we illustrate our approach with a simple task. The task is to move a point in the plane into a circular disk centered at the origin of the plane. We will look at the design of a sensor for this task, that is specifically structured to guide the point into the disk. In the process, we will see that it is possible to design a family of sensors. The family of sensors defines a tradeoff between the optimal approach strategy that is possible with perfect sensing and the motions possible given the actual information available from an imperfect sensor.

4.1 Actions and Uncertainty

We take the motion system to be a simple first-order system, in which the point is moved by commanding velocities that are subject to unknown but bounded errors. Specifically, if the execution system commands velocity $\mathbf{v}_0$, then the range of possible velocities is given by the ball $B_{\epsilon_v|\mathbf{v}_0|}(\mathbf{v}_0) = \left\{\mathbf{v}^* \;\middle|\; |\mathbf{v}_0 - \mathbf{v}^*| \leq \epsilon_v |\mathbf{v}_0|\right\}$. A basic action consists of commanding a velocity $\mathbf{v}_0$ for some duration of time Δt, where Δt can be arbitrarily small. This models a simple feedback loop.

4.2 An Ideal Sensor

Let us ask ourselves what the ideal sensor for the point-into-disk task might look like. A sensor that reports precisely the current position of the point would

certainly be nice. In particular, given a position reported by this sensor, the execution system can calculate a velocity vector that aims directly towards the goal. We will see presently that this perfect position sensor is actually superideal, in that it provides more information than is needed.

4.2.1 A World-Centered Sensor: Poor Degradation with Noise

One issue to keep in mind is how one might implement a perfect position sensor. One possibility is to estimate the robot's position by observing it, say with a camera. Another implementation might use a series of joint encoders. A third implementation might consist of a grid of rectangular wires implanted in the plane, that report back the (x, y) position of the robot.

Observe that all of these sensors are world-centered. Specifically, they report back the configuration of the robot in some world coordinate system. Given this configuration, the run-time executive then calculates the difference in position between the robot and the goal in order to suggest a commanded velocity. This procedure is fine for a truly perfect sensor, but degrades poorly as the sensor becomes noisy or as the position of the goal becomes uncertain. In particular, suppose we model the uncertainty of such a world-centered sensor as an error ball with a fixed error radius. If the error radius is larger than the goal, then as the robot approaches close to its goal, the system will be unable to decide on which side of the goal the robot is located. This makes it impossible to execute a velocity guaranteed to attain the goal.

While this example is simple, the problem is fundamental. It exists in even greater magnitude for general manipulation tasks.

4.2.2 A Task-Centered Sensor: Respects Uncertainty

A second approach is to build a task-centered sensor. What might such a sensor for the point-into-disk task look like? A good representation for a perfect sensor is not a cartesian grid but a polar coordinate grid that is centered at the goal. As with the perfect cartesian sensor, the perfect polar coordinate sensor reports back the configuration of the robot relative to the hole, but in polar coordinates. The coordinates are then again used to compute a velocity vector along which the robot should move.

Fortunately, we can go yet a step further. The polar coordinate sensor reports back both the distance of the robot from the goal and the angle that measures the relative direction between the robot and the goal. Only the angle is used to calculate a motion command; the distance is ignored. This observation immediately suggests that the ideal sensor need only report the angular coordinate of the polar coordinate representation. In short, the ideal perfect sensor is a directional beacon situated at the goal, that reports the direction from the goal to the robot. This sensor senses one degree of informational freedom.

Let us observe that this task-centered sensor degrades nicely with noise. In particular, an error of a few degrees in the directional beacon simply means that the robot will move slightly in the wrong direction. Near the goal, the sensor can quickly detect and correct such inaccurate motions.

There are two principles at play here. First, the sensor is task-centered. Second, the sensor error mirrors the control error.

4.3 Designing a Sensor with Time–Indexed Backprojections

The basic procedure for designing both the sensor and the strategy that together accomplish the task is to backchain from the goal under the assumption of a perfect sensor. The reason for using a perfect sensor is that it considerably simplifies the planning problem, without forcing us into a particular implementation of the sensor. The backchaining process tells us what information is required, how it is used, and how to design a sensor so that the information degrades nicely in the presence of uncertainty.

The backchaining process is inductive. Initially, the planner constructs a collection of backchaining regions from the goal disk. At each subsequent stage, the planner constructs a collection of backchaining regions from some other circular region.

Recall that our basic model of an action is a pair: a velocity $\mathbf{v}_0$ and a time Δt. This model of action leads to time-indexed backprojections. See Figure 1. The semantics are as follows. If the robot starts off in the crescent-shaped region depicted in the figure, then upon execution of velocity $\mathbf{v}_0$ for time Δt, the robot is guaranteed to be somewhere inside the disk of radius r_k, despite control uncertainty.

4.4 Planning a Strategy

During a given stage of the backchaining process, the planner expands the current disk of radius r_k to a new disk of radius $r_k + (1 - \epsilon_v)\,|\mathbf{v}_0|\,\Delta t$. Specifically, the planner computes time-indexed backprojections for all possible velocity directions. The union of the resulting crescent-shaped regions forms a band of

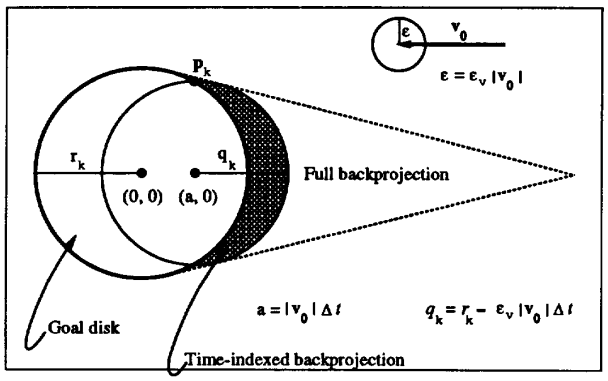

Figure 1: This figure shows a typical time-indexed backprojection of a two-dimensional disk. The goal disk has radius r_k and is centered at the origin. The action is to execute the nominal velocity $\mathbf{v}_0$ for time Δt. The velocity uncertainty is given by an error ball with radius $\epsilon_v |\mathbf{v}_0|$. The resulting time-indexed backprojection is a disk with radius $q_k = r_k - \epsilon_v |\mathbf{v}_0| \Delta t$ centered at $(a, 0) = -\mathbf{v}_0 \Delta t$. The shaded crescent-shaped area is the portion of the backprojection that lies outside of the goal disk. The point $\mathbf{p}_k$ is one of the intersection points of the circles that circumscribe the two disks. For comparison, the dashed lines outline the full backprojection (see [Erdmann 1986]).

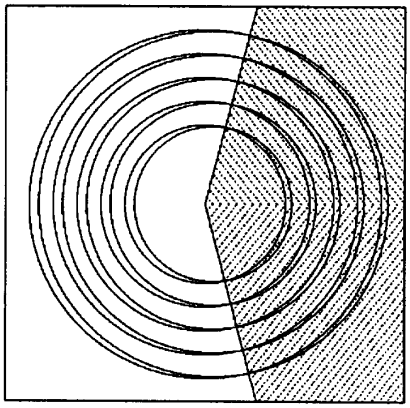

Figure 2: The two solid lines bound a cone of progress, which is indicated by the shaded region. The nominal commanded velocity is $\mathbf{v}_0 = (-1, 0)$, with uncertainty $\epsilon_v = 0.25$. If the robot starts within the cone of progress, then commanding velocity $\mathbf{v}_0$ for a differential amount of time is guaranteed to move the robot closer to the origin, despite the uncertainty in control. Each pair of circles comprises a composite subgoal disk and its time-indexed backprojection relative to the velocity $\mathbf{v}_0$ the time duration $\Delta t = 0.1$. Observe that the solid lines nearly trace out the intersection points of each subgoal disk with its time-indexed backprojection. As Δt becomes smaller, this match becomes better.

width $(1 - \epsilon_v) |\mathbf{v}_0| \Delta t$ about the old composite subgoal disk of radius r_k.

4.5 Differential Backchaining

Suppose that one repeatedly computes time-indexed backprojections, starting with the goal disk of radius r, and backchaining over and over. At the $k+1^{st}$ stage one has a disk of radius r_k centered at the origin. Here k runs from 0 on upward, with $r_0 = r$. Given r_k and $\mathbf{v}_0 \Delta t$, one computes a time-indexed backprojection, which is a disk of radius q_k centered at $-\mathbf{v}_0 \Delta t$. The circles bounding these two disks intersect at the points $\mathbf{p}_k$ and its mirror image (relative to the line given by $\mathbf{v}_0$).

For simplicity, let us assume that $\mathbf{v}_0$ is parallel to the x-axis and that it is pointing in the negative x direction. Let us define $a = |\mathbf{v}_0 \Delta t|$, and write $\mathbf{p}_k = (x_k, y_k)$. Then we observe that $q_k = r_k - \epsilon_v a$ and that $r_k = r + k(1 - \epsilon_v) a$. By solving for the intersection of two circles, it follows that

$$x_k = r\,\epsilon_v + k(1 - \epsilon_v)\epsilon_v\, a + \frac{1}{2}(1 - \epsilon_v^2)\, a.$$

Furthermore, y_k is implicitly defined by the equation $x_k^2 + y_k^2 = r_k^2$.

The interesting question is what happens as we make the time constant Δt of the feedback loop very small. Suppose we let $a \to 0$. In order to backchain out to some location $\mathbf{p}$, we must therefore let k approach infinity in such a way that the product $k\,a$ remains constant, say at some value c that depends on $\mathbf{p}$. Thus we have that

$$\begin{aligned} x_k &\to \epsilon_v \Big(r + c(1 - \epsilon_v)\Big), \\ y_k &\to \sqrt{1 - \epsilon_v^2}\,\Big(r + c(1 - \epsilon_v)\Big). \end{aligned}$$

We therefore see that

$$\frac{y_k}{x_k} \to \frac{\sqrt{1 - \epsilon_v^2}}{\epsilon_v}.$$

In other words, the intersection points $\{\mathbf{p}_k\}$ line up in a straight line with slope $\sqrt{1 - \epsilon_v^2}\big/\epsilon_v$. A similar line is formed by the mirror points $\{-\mathbf{p}_k\}$. Together these two lines form a cone.

What does this cone mean? The interior of the cone consists of all those locations at which the nominal velocity $\mathbf{v}_0$ is guaranteed to make progress towards the goal for at least a differential amount of time. We refer to such a cone as a *cone of progress.*

4.6 Cones of Progress

The previous construction should come as no surprise. The cone appeared in a different form in [Erdmann

1990] in the context of randomization. That work discusses a simple feedback loop for accomplishing the point-into-disk task, despite uncertainty in sensing and control. The feedback loop operated as follows. The robot would sense its current position. If all position interpretations consistent with the observed sensor value lay wholly within some cone of progress, then the robot would determine a velocity vector and a maximum execution time that would permit the robot to move closer to the goal. On the other hand, if no single cone of progress contained all the possible locations of the robot, then no action was guaranteed to move the robot closer to the goal. In that case, the robot would execute a random motion.

There are two important ideas contained in the last paragraph. The first concerns speed of progress. The second is a description of a better sensor. First, in the randomization example of [Erdmann 1990], the sensor was a cartesian sensor whose uncertainty was modelled as an error ball. This meant that for observed sensor values close to the origin, the system could not decide that its position lay in any particular cone of progress. Conversely, for observed sensor values far away from the origin, the location of the sensor interpretation set inside a cone of progress determined the amount of progress that the system could make. In other words, we see that it is the relationship of a particular sensor and sensed value to all the cones of progress that determines whether progress is possible and if so, how much progress. This is the sensing-speed tradeoff that we mentioned earlier.

The second idea concerns the design of a task-specific sensor. We argued earlier based on simple symmetry that the sensor should be an angular sensor that is centered at the goal and that looks out from the goal toward the robot (or vice-versa). That argument finds further support in the current geometric reasoning. In order for the robot to be certain that a particular action will move it closer to the goal, the robot needs to know that its current position lies within the cone of progress associated with that action. This requirement points towards the design of a sensor that recognizes cones of progress. For the example above, this means that we need a sensor that can decide whether the robot is located in some conical region with half-angle $\cos^{-1}(\epsilon_v)$ that is centered at the goal.

In summary:

- **Sensors should not recognize states.**
- **Sensors should recognize applicable actions, by recognizing cones of progress.**

4.7 Coverings by Cones of Progress

Designing a sensor to recognize a single cone of progress is not enough. The entire state space need not be covered by a single cone of progress. This leads us to a family of sensors. At one extreme we have a sensor that is capable of recognizing the robot's presence or absence in each and every cone of progress. For the example above this amounts to a perfect angular sensor. Given such a sensor we would at run-time see the robot executing a nearly straight-line trajectory from its initial location to the goal.

At the other extreme we have a sensor that is capable at run-time of recognizing the robot's presence in at least one cone of progress, but not necessarily more. One view of such a sensor is as follows. We, as sensor designers, decide on a covering of the state space by some small collection of cones of progress. (There may be many such minimal coverings.) We then design a sensor that can recognize this covering. In other words, at run-time the sensor accurately reports the robot's position in at least one cone of progress. Note that the speed of progress is slower than for the perfect angular sensor. Instead of heading straight for the goal, the robot moves along a path of directions aligned with the axes of those cones that comprise the covering.

4.8 Implementation

We implemented a radial sensor for inserting a cylindrical peg into a hole, using a Zebra Zero robot. The peg's diameter was about 6mm, and the hole's diameter was about 7mm. We built a small gripper attachment with four IR detectors in it. When the robot grasps the peg, the peg centers itself in the gripper so that it is surrounded by the IR detectors. We placed an IR source inside the hole. Near the hole the sensors detect the direction in which the robot must move the peg. For each execution, the robot would initially place the peg within a 3 to 7mm radius of the hole. The robot would then consult the IR detectors, move the peg about 1.5mm in a radial direction toward the hole, and repeat until it placed the peg over the hole. Once over the hole, the robot would simply open the gripper and let the peg fall in. We did not keep experimental data. In viewing old videotape it seems that the strategy places the peg into the hole reliably about 75% of the time. This is fine for our purposes. It is not great, and of course, the clearance ratio 1.17 is huge, but the setup serves as a feasibility test.

We note that placing an IR source in the hole is not practical in realworld settings. Instead, one might build a special gripper with proximity sensors instead

of IR detectors. These sensors should differentiate between solid metal and hole. Such a gripper cum proximity detector provides information much as did our previous gripper cum IR source/detector. The advantage is the localization of the sensor to the gripper.

Finally, we note that the radial sensor is simply a description of the information required to insert the peg into the hole. A real sensor need not be implemented as a "lighthouse". This is really just a metaphor describing the required information. An excellent implementation of a radial sensor occurred long ago, in the Charles Stark Draper Labs. [Nevins et al. 1975] used force sensing to detect the moments that result when a peg overlaps a hole. The perpendicular to this moment describes a radial direction of motion, and thus implements a radial sensor

4.9 Summary

The cones of progress specify exactly what knowledge the robot must have in order to make progress, and how the available information affects the speed of progress. A sensor should be viewed as recognizing cones of progress. A perfect sensor maps an observed sensed value to all cones of progress that contain the robot's actual location. A noisy sensor maps the observed sensed value to a smaller collection of cones of progress. Given this description, one can then design physical sensors that try to recognize as many cones of progress as one requires in order to accomplish a task quickly.

5 Acknowledgments

Many thanks to Bruce Donald, Tomás Lozano-Pérez, Matt Mason, Tamara Abell, Yan-Bin Jia, and Nina Zumel for their insights, discussions, and comments. Support for this research was provided in part by Carnegie Mellon University, an equipment grant from AT&T, an NSF Research Initiation Award (IRI-9010686), and an NSF Presidential Young Investigator award (IRI-9157643).

6 Bibliography

Brooks, R. A. 1982. Symbolic Error Analysis and Robot Planning. *Int J Rob Res.* 1(4):29–68.

Buckley, S. J. 1987. Planning and Teaching Compliant Motion Strategies. AI-TR-936. Ph.D. Thesis. MIT.

Canny, J. F. 1988. *The Complexity of Robot Motion Planning.* MIT Press.

Canny, J. F. 1989. On Computability of Fine Motion Plans. *Proc 1989 IEEE ICRA*, pp. 177–182.

Canny, J. F., and Goldberg, K. Y. 1991. "RISC-Robotics," talk given at CMU, Oct 11, 1991.

Canny, J., and Reif, J. 1987. New Lower-Bound Techniques for Robot Motion Planning Problems. *28th FOCS.*

Donald, B. R. 1988. The Complexity of Planar Compliant Motion Planning with Uncertainty. *ACM S Comp Geom.*

Donald, B. R. 1989. *Error Detection and Recovery in Robotics.* Springer LNCS No. 336.

Donald, B. R., and Jennings, J. 1992. Constructive Recognizability for Task-Directed Robot Programming. *Robotics and Autonomous Systems.* 9:41–74.

Erdmann, M. A. 1984. On Motion Planning with Uncertainty. AI-TR-810. S.M. thesis. MIT.

Erdmann, M. A. 1986. Using Backprojections for Fine Motion Planning with Uncertainty. *Int J Rob Res.* 5(1).

Erdmann, M. A. 1990. On Probabilistic Strategies for Robot Tasks. AI-TR-1155. Ph.D. thesis. MIT.

Goldberg, K. 1990. Stochastic Plans for Robotic Manipulation. Ph.D. Thesis. CMU.

Latombe, J.-C. 1991. *Robot Motion Planning.* Kluwer.

Lazanas, A. and Latombe, J.-C. 1992. *Landmark-Based Robot Navigation. Proc AAAI-92.*

Lozano-Pérez, T. 1976. The Design of a Mechanical Assembly System. AI-TR-397. S.M. thesis. MIT.

Lozano-Pérez, T., Mason, M. T., and Taylor, R. H. 1984. Automatic Synthesis of Fine-Motion Strategies for Robots. *Int J Rob Res.* 3(1):3–24.

Mason, M. T. 1981. Compliance and Force Control for Computer Controlled Manipulators. *IEEE Trans Sys Man Cyber.* SMC-11(6):418–432.

Mason, M. T. 1984. Automatic Planning of Fine-Motions: Correctness and Completeness. *Proc 1984 IEEE ICRA.*

Natarajan, B. K. 1986. An Algorithmic Approach to the Automated Design of Parts Orienters. *Proc 27th FOCS.*

Natarajan, B. K. 1988. The Complexity of Fine Motion Planning. *Int J Rob Res.* 7(2):36–42.

Nevins, J., Whitney, D., Drake, S., Killoran, D., Lynch, M., Seltzer, D., Simunovic, S., Spencer, R. M., Watson, P., and Woodin, A. 1975. Exploratory Research in Industrial Modular Assembly. Report R-921. C. S. Draper Laboratory, Cambridge, Mass.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The Complexity of Markov Decision Processes. *Math Op Res.* 12(3):441–450.

Raibert, M. H., and Craig, J. J. 1981. Hybrid Position/Force Control of Manipulators. *J Dyn Sys Meas Cont.* 102:126–133.

Taylor, R. H. 1976. A Synthesis of Manipulator Control Programs from Task-Level Specifications. AIM-282. Ph.D. thesis. Stanford University.

Whitney, D. E. 1977. Force Feedback Control of Manipulator Fine Motions. *J Dyn Sys Meas Cont.* 98:91–97.