MOTION PLANNING WITH UNCERTAINTY: A LANDMARK APPROACH

ANTHONY LAZANAS
lazanas@flamingo.stanford.edu

JEAN-CLAUDE LATOMBE
latombe@cs.stanford.edu

ROBOTICS LABORATORY, DEPARTMENT OF COMPUTER SCIENCE
STANFORD UNIVERSITY, STANFORD, CA 94305, USA

Abstract

In robotics uncertainty exists at both planning and execution time. Effective planning must make sure that enough information becomes available to the sensors during execution, to allow the robot to correctly identify the states it traverses. It requires selecting a set of states, associating a motion command with every state, and synthesizing functions to recognize state achievement. These three tasks are often interdependent, causing existing planners to be either unsound, incomplete, and/or computationally impractical. In this paper we partially break this interdependence by assuming the existence of landmark regions in the workspace. We define such regions as "islands of perfection" where position sensing and motion control are accurate. Using this notion, we propose a sound and complete planner of polynomial complexity. Creating landmarks may require some prior engineering of the robot and/or its environment. Though we believe that such engineering is unavoidable in order to build reliable practical robot systems, its cost must be reduced as much as possible. With this goal in mind, we also investigate how some of our original assumptions can be eliminated. In particular, we show that sensing and control do not have to be perfect in landmark regions. We also study the dependency of a plan on control uncertainty and we show that the structure of a reliable plan only changes at critical values of this uncertainty. Hence, any uncertainty reduction between two consecutive such values is useless. The proposed planner has been implemented. Experimentation has been successfully conducted both in simulation and using a NOMAD-200 mobile robot.

Acknowledgments: This research was funded by DARPA/ONR contract N00014-92-J-1809. The authors thank Craig Becker, Philippe Moutarlier, Joaquin Salas, Benjamin Schalet, Ken Tokusei, and David Zhu for their experimental work with the planner using the NOMAD-200 robot. Section 11, Experimentation with a Real Mobile Robot, is mainly due to Craig Becker and Ken Tokusei.

1 Introduction

Motion planning with uncertainty is a critical problem in robotics. Indeed, even the most complex models of the physical world cannot be perfectly accurate. Furthermore, increasing model complexity often adversely affects the ability of a robot to plan its actions efficiently. Thus, the use of simplified models seems to be the only practical way to proceed. All details omitted from these models unite to form uncertainty. A robot planner must produce plans that reliably achieve their goals despite errors, i.e., differences between the models and the real world.

Taking uncertainty into account at planning time is essential when potential errors are comparable to or larger than the tolerances allowed by the task. This is the case, for example, in mechanical part-mating tasks, where both errors and tolerances are usually small, and mobile robot navigation tasks, where both errors and tolerances may be large. For such tasks, classical path planning methods [23], which use simple geometric models while assuming null uncertainty, are clearly insufficient. At best, they produce paths that require frequent replanning to deal with discrepancies detected by sensors during execution. But, due to errors in sensing, they may also lead the robot to incorrectly believe that it has achieved some expected state or, on the contrary, that it has failed to achieve this state. To generate reliable plans, the planner must choose courses of actions whose execution is guaranteed to make enough knowledge available to allow the robot to correctly identify the states it traverses, despite execution-time errors in control and sensing [2]. The overall task of the planner is to select an adequate set of states, associate appropriate motion commands with these states, and synthesize state-recognition functions. All three subtasks are interdependent.

One often expects too much from a system reasoning in the presence of uncertainty. This attitude has often led to engineering brittle complicated systems whose actual capabilities are difficult to assess. Experiments may show that these systems work beautifully on some tasks, but fail ungracefully on seemingly simpler ones, without one being able to predict one outcome or the other. We believe that the design of a reliable system dealing with uncertainty must be based on bounded, well-defined expectations of what may actually happen in the real world, allowing some desirable operational properties to be achieved. In this paper we propose a precisely defined motion planning problem with uncertainty, and a sound, complete, and polynomial algorithm that solves this problem. By sound, we mean that the planner only generates correct plans, which are guaranteed to succeed if some predefined assumptions bounding uncertainty are satisfied. By complete, we mean that the planner returns a correct plan whenever one exists; otherwise, it declares failure. By polynomial, we mean that the worst-case asymptotic time complexity of the planner is a polynomial function of the size of the input problem.

While several motion planners with uncertainty have been proposed (e.g., [31, 39, 4, 36, 15, 25, 10, 22, 32, 16, 11, 24, 38]), soundness and completeness are provable for only a few of them. Furthermore, most known sound and complete planning algorithms take exponential time in some measure of the size of the input problem (typically, the complexity of the robot's environment) [7, 12], which makes them virtually inapplicable. In fact, it has been shown that these motion planners attack problems which are inherently intractable [6]. In our planner, like in previous planners, soundness and completeness are proven under the assumption that all errors are restricted to lie within bounded uncertainty intervals. We achieve polynomial-time planning both by making assumptions in the problem formulation that eliminate the source of intractability and by using algorithms that take advantage of these assumptions. The key notion underlying these assumptions is that of a landmark, an "island of perfection" in the robot configuration space where position sensing and motion control are accurate (however, we will see later that neither control nor sensing need be perfect in landmark regions). This notion considerably simplifies the selection of the set of states that the robot may traverse and the synthesis of the state-recognition functions. It mainly reduces planning to selecting motion commands to navigate from landmark to landmark until the goal is achieved. For a given problem, our planner generates a plan represented in the form of a collection of reaction rules. Each rule is attached to some region of the robot configuration space and specifies the motion command to be executed if this region is attained. Such a plan is reminiscent of a reaction plan as proposed in [8, 14, 37].

Creating landmarks requires some prior engineering of the robot and/or its environment. Though we believe that such engineering is unavoidable in order to build reliable practical robot systems, it is important that its cost be reduced as much as possible (after all, industrial mobile robots tracking wires have been in use for many years). This leads us to analyze how uncertainty bounds affect plans. We show that the structure of a correct plan must change only at a finite number of critical values of these bounds. Hence, any additional engineering reducing uncertainty bounds without crossing a critical value is useless. We also show that some uncertainty in control and sensing can be allowed in landmark regions without affecting the correctness of a generated plan.

The current geometric techniques used by our algorithms are restricted to two-dimensional configuration spaces. This limits the application of the planner, mainly to circular mobile robots on flat terrain. The planner was implemented for such robots. We have successfully experimented with it on many problems, both in simulation and using a real robot in our laboratory environment. Since the plans generated by our planner are provably correct, the main goal of our experiments with a real robot was to verify that our concept of a landmark did not induce unacceptable engineering costs. Actually, we will see that our work led us to design different sorts of relatively low-cost landmarks. Our experiments also illustrate the need for considering and dealing with different uncertainty bounds.

The main contribution of this paper is to provide solid, theoretically and experimentally proven foundations for implementing mobile robot navigation systems, along with specific algorithmic planning techniques. We believe, however, that a potential impact of our work is to redefine the purpose of experimentation in robot planning with uncertainty (more generally, in autonomous systems). When the assumptions underlying the design of a planner are left implicit or informal, experimentation is aimed at verifying that the planner works adequately on a representative sample of tasks. When assumptions are precisely stated and the planning algorithm is provably sound and complete for these assumptions and reasonably efficient, as is the case of our planner, experimentation is aimed at verifying that satisfying the assumptions does not entail excessive engineering. This second purpose results in an easier experimental task. It may also lead to defining standard sets of assumptions, known to induce acceptable engineering costs, hence allowing subsequent research to be specifically aimed at developing more efficient planners operating under such assumptions.

Section 2 situates our work relative to previous related work. Section 3 precisely states the planning problem addressed in this paper, lists the main results obtained, and illustrates them with an example. Section 4 outlines the principle of our planning algorithm. Section 5 identifies criticalities that partition the continuous search space of this algorithm into a polynomial-size discrete space, without loosing completeness. Section 6 instantiates the planning algorithm in the case where the uncertainty in control is fixed. Sections 7 and 8 deal with variable control uncertainty and present algorithms to generate one-step and multi-step plans, respectively. Section 9 briefly describes the implementation of the planners and gives planning times. Section 10 analyzes assumptions made in our work and discusses its relevance to actual robot navigation. Section 11 describes an effective landmark design and reports on experimental results with a NOMAD-200 mobile robot equipped with a navigation system to execute plans generated by the planner of Section 6. Section 12 discusses other extensions of the planner.

This paper builds upon and extends previous work presented in [27]. There is some limited overlap between the two publications, so that the current paper is self-contained.

2 Background

Motion planning with uncertainty has been a research topic in robotics for almost two decades. Two main application domains have been considered: part mating for mechanical assembly and mobile robot navigation. Various approaches have been proposed, which are applicable to one domain, or the other, or both. They include skeleton refinement [31, 39, 4, 36], inductive learning from experiments [15], iterative removal of contacts [25, 10, 22], and preimage backchaining [32, 34, 16]. Skeleton refinement, inductive learning, and iterative removal of contacts essentially operate in two phases: a motion plan is first generated assuming no uncertainty and then transformed to deal with uncertainty. Instead, preimage backchaining takes uncertainty into account throughout the whole planning process. In principle, it can tackle problems where uncertainty shapes the structure of a plan to the extent that it cannot be generated by transforming an initial one produced under the no-uncertainty assumption. Our work is a direct continuation of a series of work on preimage backchaining. We focus most of our discussion on this series.

Preimage backchaining considers the following class of motion planning problems [32]: A plan is a sequence (more generally, an algorithm) of motion commands, each defined by a *commanded direction of motion* d and a *termination condition* **TC**. When the robot executes such a command in free space, it moves along a direction contained at each instant in a cone of half-angle θ about d and stops as soon as **TC** becomes true. (In contact with an obstacle, the robot may stop or slide, depending on the particular control law that is used. In this paper we will simply forbid contacts with obstacles.) The angle θ is the largest expected control error and models the *directional uncertainty* of the robot. The termination condition **TC** is a boolean function of sensory data s. At any one time, these data measure physical parameters (e.g., the robot position coordinates) with some error. The actual parameter vector lies anywhere in a region $\mathcal{U}(s)$ modeling the robot's *sensory uncertainty*. A planning problem is specified by a workspace model, an *initial region* where the robot is known to be prior to executing the plan, a *goal region* in which the robot must stop when plan execution terminates, the directional uncertainty θ, and the sensing uncertainty $\mathcal{U}$. A *correct* plan is one whose execution guarantees the robot to enter the goal region and stop in there despite errors in control and sensing. A *sound* planner is one which generates only correct plans. A *complete* planner is one which returns a correct plan whenever one exists, and failure otherwise.

The above problem formulation admits many variants. For example, one may introduce time and consider uncertainty in the robot velocity, allowing the construction of more sophisticated termination conditions [34, 16]. The workspace model (e.g., the location and the shape of the obstacles) may also be subject to errors, yielding a third type of uncertainty [11]. For the sake of simplicity we will not discuss these variants here (see [23]).

The *preimage* of a goal region for a given motion command $\mathbf{M} = (d,\mathbf{TC})$ is the set of all points in the robot's configuration space such that if the robot starts executing the command from any one of these points, it is guaranteed to reach the goal and stop in it. Preimage backchaining consists of constructing a sequence of motion commands $\mathbf{M}_i$, $i = 1, \ldots, n$, such that, if $\mathcal{P}_n$ is the preimage of the goal for $\mathbf{M}_n$, $\mathcal{P}_{n-1}$ the preimage of $\mathcal{P}_n$ for $\mathbf{M}_{n-1}$, and so on, then $\mathcal{P}_1$ contains the initial region.

One source of difficulty in computing preimages is the interaction between goal reachability and goal recognizability. The robot must both reach the goal (despite directional uncertainty) and stop in the goal (despite sensing uncertainty). Goal recognition, hence the termination condition of a command, often depends on the region from where the command is executed. This region, which is precisely the preimage of the goal for that command, also depends on the termination condition. This recursive dependence was noted in [32]. Despite this difficulty, Canny [7] described a complete planner with very few restrictive assumptions in it. This planner generates an r-step plan by reducing the input problem to deciding the satisfiability of a semi-algebraic formula with $2r$ alternating existential and universal quantifiers. Such a decision takes double exponential time in r. Moreover, the smallest r for which a plan may exist grows with the complexity of the environment. Actually, various forms of the above motion planning problem have been proven intrinsically hard [6, 35, 7].

At the expense of completeness, Erdmann [16] suggested that goal reachability and recognizability be treated separately by identifying a subset of the goal, called a *kernel*, such that when this subset is attained, goal achievement can be recognized (by $\mathbf{TC}$) independently of the way it has been achieved. He defined the backprojection of a region $\mathcal{R}$ for a motion command $\mathbf{M}$ as the set of all points such that, if the robot executes $\mathbf{M}$ starting at any one of these points, it is guaranteed to reach $\mathcal{R}$. He proposed an $O(n \log n)$ algorithm to compute backprojections in the plane when the obstacles are polygons bounded by n edges. An implemented planner based on this approach is described in [24].

Once the kernel of a goal has been identified, a remaining issue is the selection of the commanded direction of motion to attain this kernel, since the backprojection of the kernel depends on this direction. The planner described in [24] only considers a finite number of regularly spaced directions; hence, it is incomplete, and usually not very efficient. The continuous set of backprojections of a region $\mathcal{R}$ for all possible directions of motion is called the *nondirectional backprojection* of $\mathcal{R}$. In the plane (i.e., for a point robot moving in the plane), Donald [12] showed that, as far as planning is concerned, this set is sufficiently described by a polynomial number of (directional) backprojections. These backprojections are computed at critical directions of motion where the topology of the backprojection's boundary changes. Donald proposed an $O(n^4 \log n)$ algorithm, with n being the number of obstacle edges, to compute nondirectional backprojections and embedded this algorithm into a polynomial one-step planner. Briggs [3] reduced the time complexity of computing a nondirectional backprojection to $O(n^2 \log n)$. Fox and Hutchinson [19] extended the algorithm to exploit visual constraints and allow visual compliant motions.

However, even when nondirectional backprojections are used, one last difficulty to construct a multi-step planner is backchaining. The difficulty comes from the fact that backchaining introduces a twofold variation: when the commanded direction of motion varies, both the backprojection of the current kernel and the kernel of this backprojection (which will be used at the next backchaining iteration) vary. In this paper (as in [26, 27]) we deal with this difficulty by introducing *landmarks*. We define a landmark region as a subset of the robot's configuration space where position sensing and motion control are perfect, while outside landmark regions sensing is null and control is imperfect with errors bounded by some given directional uncertainty θ (see Section 3). We show that backchaining is then reduced to iteratively computing a relatively small set of (directional) backprojections for a growing set of landmark regions. This result directly yields a complete planning algorithm that takes polynomial time in the complexity of the landmark and obstacle regions. Previously, Friedman [20] had proposed another polynomial multi-step planner for a compliant point robot in a polygonal workspace by assuming that sensing exactly detects when the robot traverses line segments joining vertices of the workspace.

The above definition of uncertainty corresponds to treating control and sensing errors as random variables with uniform distribution over bounded domains. The advantage of bounding errors over, say, a Gaussian distribution model is that it yields the neat and convenient notion of a correct motion plan. However, a given planning problem may admit no such plans, while it could have admitted one if uncertainty had been set slightly smaller. Furthermore, when correct plans exist, they may be overly complex. To deal with this drawback, Donald [11] proposed the notion of an Error Detection and Recovery Strategy, defined as an r-step plan ($r > 1$) that attains the goal whenever it is recognizably reachable and signals failure whenever chances that it serendipitously achieve the goal vanish. Erdmann [17] introduced randomized plans whose execution converges toward the goal with probability one. In this paper we propose a different approach which consists of allowing the directional uncertainty θ to vary over some interval. We propose computational tools to build planners that can select values of θ according to various optimization criteria. For example, θ may be actually tunable by the robot. (Most robots can achieve such control; they may reduce θ by slowing down and devoting more computation time to determine which commands to send to the actuators, allowing more accurate dynamic models of the robot's mechanical structure to be used.) Then, if no correct plans exist for some value of θ, or if the existing plans are too complex, the planner may try to use smaller values of θ. However, reducing directional uncertainty generates some cost (e.g., lower speed), which the planner should strive to minimize. As we will see, allowing θ to vary has other interesting applications, such as dealing with unexpected obstacles and generating probabilistic plans. As indicated earlier, understanding the dependency of correct plans on directional uncertainty may also facilitate the task of engineering the robot and its environment.

Our notion of a landmark corresponds approximately to a recognizable feature of the workspace that induces a field of influence (the landmark region); if the robot is in this field, it senses the landmark. Similar notions have been previously introduced in the literature with different names, e.g., landmarks [30], atomic regions [5], signature neighborhoods [33], perceptual equivalent classes [13], sensory uncertainty field [38], and visual constraints [21, 19]. Our landmarks are mainly aimed at simplifying the selection of the set of states that the robot may traverse and the synthesis of the state-recognition functions. Instead, research like the one described in [13] is aimed at automating state selection and state recognition. Although it has great potential in reducing the engineering cost and the limitations associated with our landmarks, it is not clear as yet whether it can yield time-efficient planners.

Over the past decade, there has also been a substantial amount of work at reducing uncertainty (mainly position uncertainty) while a robot is moving. For example, for mobile robots, many techniques have been proposed to combine the estimates provided by both dead-reckoning and environmental sensing (e.g., see [1, 9, 29, 40]). These techniques address the problem of tracking a selected motion plan as well as possible, not the problem of generating this plan. The goal of planning in the presence of uncertainty is to make sure that executing the plan will reveal enough information to guarantee reliable execution. Notice also that planning and execution may use different models. For instance, modelling errors as random variables uniformly distributed over compact subsets makes sense at planning time. However, it may be preferable to use more sophisticated probabilistic distributions at execution time to better use all sources of information then available.

Reaction plans have been previously proposed as a way to deal with uncertainty at planning time [37, 8, 14]. Such plans consist of goal-oriented rules triggered by the data available at execution time. In particular, Schoppers [37] developed the notion of a universal plan whose rules cover all possible situations that may occur at execution time. The plans generated by our planner are very similar. They consist of motion commands attached to regions of the robot's configuration space. When such a region is entered by the robot, the associated motion command is executed.

The interesting point is that our planner takes polynomial time, while the generation of universal plans is often believed to be exponential. However, unlike the planners mentioned above, our planner addresses a restricted family of planning problems in which uncertainty is well-bounded.

3 Statement of Problem and Results

Problem Statement: The planning problem considered in this paper is the following:

The robot is a point[1] moving in a plane, called the *workspace*, containing forbidden circular regions, the *obstacles*, and other circular regions, called the *landmark disks*. Both the obstacles and the landmark disks are stationary. The robot is not allowed to collide with any of the obstacles. The number of landmark disks is finite and equal to ℓ. The number of obstacle disks is also finite and in $O(\ell)$. (Throughout this paper the number ℓ is used to measure the size of the input problem.)

The robot has perfect position sensing in the landmark disks and no sensing outside the landmark disks. It can move in either the *perfect* or the *imperfect-control* mode. In the perfect-control mode, it navigates without error. In the imperfect-control mode, its actual direction of motion at any instant differs from the commanded direction of motion by an angle bounded by θ (directional uncertainty). The perfect-control mode is feasible only in landmark disks. If several such disks form a connected area, called a *landmark region*, the robot can move in the perfect-control mode over all this area. The imperfect-control mode is applicable everywhere. The robot has no sense of time (thus, the modulus of its velocity is irrelevant to the planning problem).

The value of θ is controllable by the robot in a connected interval $[\theta_{min}, \theta_{max}] \subset [0, \pi/2)$. A motion command in the imperfect-control mode is specified as a triplet $(d, \theta, \mathcal{L})$, where $d \in S^1$ (the unit circle) is the commanded direction of motion, $\theta \in [\theta_{min}, \theta_{max}]$ is the directional uncertainty, and $\mathcal{L}$ a set of landmark disks defining the termination condition (the robot stops as soon as it enters one of these disks). $\mathcal{L}$ is called the *termination set* of the command. A decreasing function defines the cost of navigating with uncertainty θ. (Throughout this paper, we will remain intentionally vague about the cost function. Our main goal is to show that controllable directional uncertainty can be handled by a planner.)

Prior to execution, the robot is known to lie anywhere in an initial region $\mathcal{I}$ that consists of one or several disks. The number of disks in $\mathcal{I}$ is small enough to be

---

[1] For a circular mobile robot, this is obtained by shrinking the robot to its center, while isotropically growing the obstacles by the robot's radius.

considered in $O(1)$. The robot must move into a given goal region $\mathcal{G}$, which may be any subset, connected or not, of the workspace, whose intersection with the landmark disks takes $O(\ell)$ time to compute.

The problem is to generate a sequence of motion commands to make the robot move from its initial position into the goal region and stop there. In doing so we wish to minimize the cost paid for the various choices of directional uncertainty in the imperfect-control motion commands.

Main Results: In the context of the above problem, the main results presented in this paper are the following:

(i) We show that the four-dimensional set of backprojections of any given collection of landmark disks, for all directions of motion d in S^1 and all values of the directional uncertainty θ in $[\theta_{min}, \theta_{max}]$, is sufficiently represented (as far as planning is concerned) by a polynomial number of backprojections, each computed for a specific value of d and θ.

(ii) From result (i), we derive three polynomial planning algorithms:

- The first algorithm assumes given constant directional uncertainty. It is sound and complete, and produces plans minimizing the number of motion commands to be executed.

- The second algorithm assumes controllable directional uncertainty. It generates correct one-step motion plans maximizing directional uncertainty. For one-step plans, it is sound and complete.

- The third planner extends the second one and uses a greedy algorithm to generate multi-step plans in which each step allows maximal directional uncertainty. It is sound and complete.

(iii) We have implemented all three algorithms, and we show sample runs obtained with the planners.

(iv) We briefly discuss other applications of the techniques described in this paper: least commitment planning to deal with unexpected obstacles, planning with anisotropic uncertainty, and generation of probabilistic plans.

Examples: Figures 1 and 2 illustrate the problem formulation given above. Each figure depicts a plan generated by one of the implemented planners.

Example 1: Fig. 1 shows an example run with the first planner. The workspace contains 23 landmark disks (shown white or grey) forming 19 landmark regions, and 25 obstacle disks. The directional uncertainty is fixed and set to 0.09 radian. The

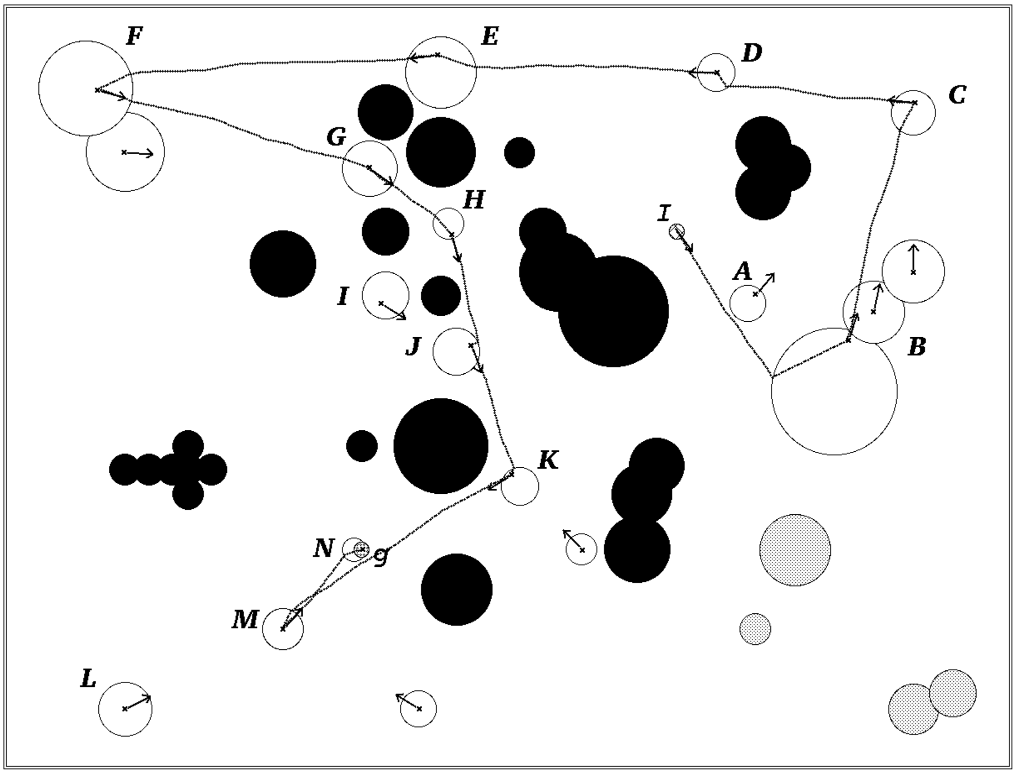


Figure 1: Planning Example 1

initial (resp., goal) region is a small disk designated by $\mathcal{I}$ (resp., $\mathcal{G}$). The white disks are those with which the planner has associated imperfect-control motion commands to attain another landmark disk. The arrow attached to the initial region or a white disk is the commanded direction of such a command. There is at least one arrow per landmark region not intersecting the goal. The arrow attached to a landmark disk originates at a point called an *exit point*.

Hence, the plan consists of motion commands distributed over landmark disks. Its execution begins with performing the command attached to $\mathcal{I}$. When the robot reaches a landmark disk in the termination set of this command, it executes a perfect-control motion command, either to a point in the goal $\mathcal{G}$, if the landmark disk belongs to a landmark region intersecting $\mathcal{G}$, or to an exit point of the landmark region. In the second case, the imperfect-control motion command associated with that point is executed, and so on.

The figure shows the path produced by a sample execution of the plan (in simula-

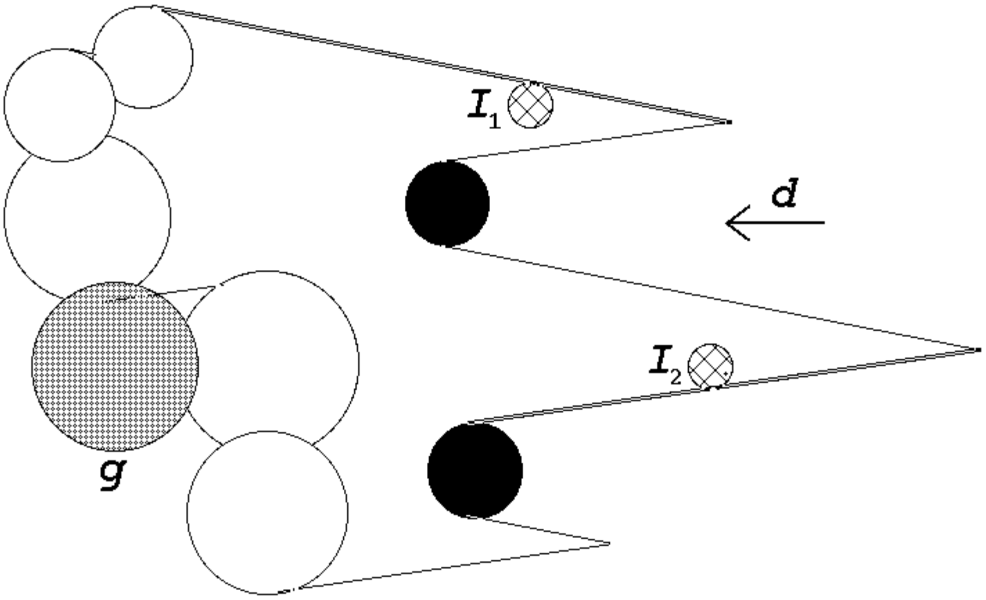


Figure 2: Planning Example 2

tion). This path first takes the robot from the initial region to the landmark region designated by B. From there, it successively attains and traverses the landmark regions marked C, D, E, F, G, H, J, K, M, and N. The number of imperfect-control motion commands executed along this path is 11. However, the generated plan could have necessitated up to 12 commands. Indeed, the termination set of the command from K contains L, M, and N. Another execution (with different control errors, but the same value of θ) could have caused the robot to reach L rather than M. The command attached to L would then have allowed the robot to reach M. No correct plans require less than 12 imperfect-control motion commands to be executed in the worst case.

Example 2: Fig. 2 shows a simple example run with the second planner. The workspace contains 5 landmark disks, with two landmark regions intersecting the goal $\mathcal{G}$, and 2 obstacle disks. The initial region consists of two disks, $\mathcal{I}_1$ and $\mathcal{I}_2$. The directional uncertainty is controllable in the interval [0.1,0.5] radian. The planner produces a one-step plan, i.e., a plan containing a single imperfect-control motion command $(d, \theta, \mathcal{L})$, with d shown in the figure, $\theta = 0.16$ radian, and $\mathcal{L}$ made up by the two landmark regions intersecting $\mathcal{G}$. The set of points in the workspace from which this command is guaranteed to reach $\mathcal{L}$ (i.e., the backprojection of $\mathcal{L}$) is outlined in the figure. It fully contains the initial-region disks $\mathcal{I}_1$ and $\mathcal{I}_2$. The value of the control uncertainty selected by the planner is maximum over all correct one-step plans. Any slight variation of the direction of motion and/or directional uncertainty would cause an initial-region disk to be partially outside the backprojection of $\mathcal{L}$.

Discussion: The above problem statement is certainly a simplification of a real mobile robot navigation problem, but we think it is not oversimplified. We will discuss the assumptions made at greater length later in this paper (mainly in Section 10). Before entering the technical details, let us survey the main points of our future discussion:

- It is straightforward to generalize our algorithms in order to deal with generalized polygonal landmark and obstacle regions, that is, regions bounded by straight and circular edges.

- Assumptions made outside landmark regions concerning sensing and control are somewhat conservative. However, they do not prevent more sophisticated models to be used at execution time. Hence, our planner may be somewhat pessimistic. But it can be run for several uncertainty bounds, including optimistic ones.

- Assumptions made inside landmark regions are anti-conservative, since neither sensing or control can ever be perfect. However, we will see that some uncertainty is acceptable within these regions, yielding the concept of a "generalized landmark region." We will describe an effective implementation of this concept.

4 Outline of a Planning Algorithm

Under the assumptions made in the problem of the previous section, if a connected set of landmark disks, i.e., a landmark region, intersects the goal region, the robot can move into the goal from any point in this set in the perfect-control mode. We call the union of the landmark regions intersecting the goal, the *extension* of the goal;[2] the other landmark disks are called the *intermediate-goal disks*. If the goal does not intersect any landmark disk, then it is considered unachievable, since the robot cannot sense its achievement. By definition of the goal extension, the robot cannot move into it using the perfect-control mode.

Given a goal $\mathcal{G}$, we first compute its extension $\mathcal{E}(\mathcal{G})$. If the initial region $\mathcal{I}$ lies

---

[2] In previous papers [26, 27], we called this set the kernel of the goal. However, this terminology was somewhat confusing, since the word 'kernel' had been previously used in preimage backchaining, with a different meaning.

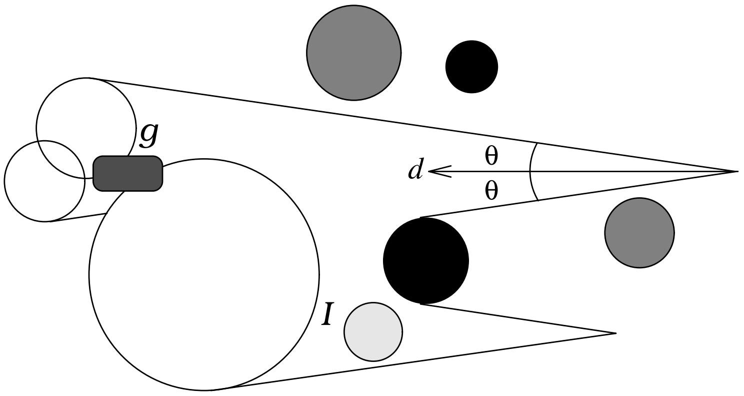


Figure 3: Backprojection of a goal extension

entirely in $\mathcal{E}(\mathcal{G})$, no further planning effort is needed since a correct plan to achieve the goal has already been found. Indeed, in a landmark region, a plan is simply a geometric path, whose computation is straightforward. Such a plan is called a *zero-step plan.* (In the following, we will measure the length of a plan by the number of imperfect-control motion commands it contains.)

The backprojection of $\mathcal{E}(\mathcal{G})$ for the pair (d, θ) is the maximal set of points, such that executing the imperfect-control motion command $(d, \theta, \mathcal{E}(\mathcal{G}))$ from any of these points is guaranteed to reach $\mathcal{E}(\mathcal{G})$. If there does not exist a zero-step plan to achieve $\mathcal{G}$, the planner may try to find a pair (d, θ), such that the initial region $\mathcal{I}$ is contained in the backprojection of $\mathcal{E}(\mathcal{G})$ for (d, θ). If one such pair (d, θ) is found, the command $(d, \theta, \mathcal{E}(\mathcal{G}))$ starting from anywhere within $\mathcal{I}$ is guaranteed to attain and terminate in $\mathcal{E}(\mathcal{G})$. From there a zero-step plan will achieve the goal $\mathcal{G}$. We call this plan a *one-step* plan.

Fig. 3 shows an example with five landmark disks (displayed grey and white), and two obstacle disks (black). The white disks form the extension of the goal $\mathcal{G}$; the grey disks are intermediate-goal disks. The initial region $\mathcal{I}$ consists of a single disk. This disk is totally included in the backprojection outlined for direction d and uncertainty θ. The backprojection is bounded by circular arcs and straight edges. The latter are supported by rays erected from landmark and obstacle disks. These rays are tangent to the disks and parallel to the directions $d \pm \theta$. Two intersecting rays form a *spike*. The backprojection of Fig. 3 has two spikes. (See [27] for a more detailed description of a backprojection.) The boundary of the backprojection consists of $O(\ell)$ arcs and edges.

A one-step plan may not exist, or may not be desirable, if its cost is too high. Then the planner can attempt to create a *multi-step* plan iteratively. At each iteration, it selects a pair (d, θ), such that the corresponding backprojection of the current goal extension intersects one or more intermediate-goal disks (landmark disks not contained in the current goal extension). All the landmark disks in the landmark regions containing these intersected disks are added to the goal extension to generate a larger extension for the next iteration; they are no longer intermediate-goal disks. The backprojection of the new goal extension is computed, and so on, until the initial region is contained in a backprojection, or no new intermediate-goal disks can be intersected, in which case a correct plan cannot possibly exist.

Let $s \in O(\ell)$ be the number of landmark regions; the number of iterations performed by the planner is bounded by s. The computation of every backprojection can be done in time $O(\ell \log \ell)$ using a traditional sweep-line algorithm given in [27]. Determining which initial-position disks are contained in the backprojection and which intermediate-goal disks are intersected by it is done while the backprojection is being computed, within the same asymptotic time complexity.

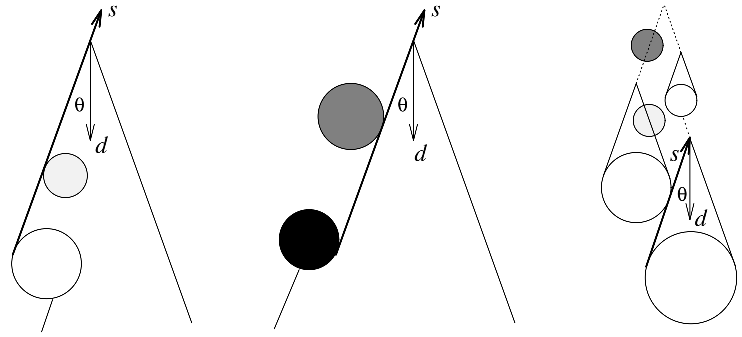
The construction of the search space explored by the above algorithm requires discretizing $S^1 \times [\theta_{min}, \theta_{max}]$, i.e., the continuous $d\theta$-space. In other words, we must answer the following question: At each iteration, which values of (d, θ) should the planner consider? In the next section we show that the $d\theta$-space can be decomposed into a finite number of cells and that one pair (d, θ) need be considered in each cell to ensure that the planner is complete. From this result we derive a finite search space that can be explored exhaustively, if necessary.

Since we allow θ to vary, one may wonder if the planner can ever return failure. The answer is obviously yes if the lower bound on θ is strictly positive. It remains yes, even if θ is allowed to become zero and $\mathcal{G}$ intersects a landmark disk. For example, this happens if the workspace contains a single landmark disk intersecting $\mathcal{G}$, while $\mathcal{I}$ consists of a single disk that is bigger than the landmark disk.

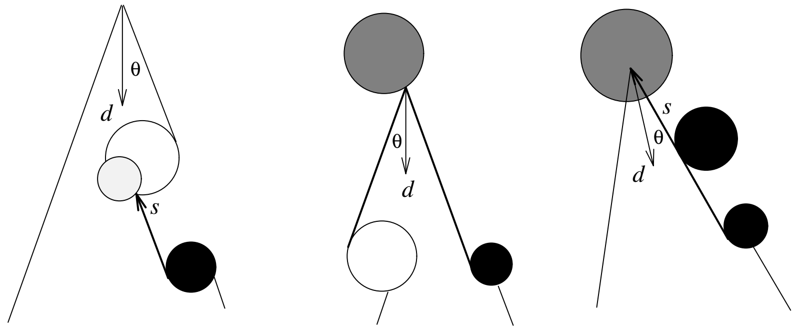
5 Building the Discrete Search Space

Each planning iteration requires selecting a pair (d, θ) such that the backprojection of the current goal extension either contains the initial region $\mathcal{I}$, or intersects intermediate-goal disks. We now show that the $d\theta$-space can be partitioned into an arrangement of curves defining *cells* of dimensions 2, 1, and 0 (points). Each cell is regular in the following sense: The backprojection of the goal extension for any pair (d, θ) in a cell C contains the same initial-position disks and intersects the same intermediate-goal disks as the backprojection for any other pair (d', θ') in C. The number of cells is polynomial in the number of landmark and obstacle disks.

Let us assume that no landmark disk intersects an obstacle disk (all disks are con-



(a) I-Cover (c) L-Touch (e) D-Touch



(b) I-Right-Vertex (d) L-Spike (f) O-Exit

Figure 4: Representative critical events

sidered to be closed subsets). The arrangement of cells is created by a network of curves corresponding to the following *critical events* (some of them are illustrated in Fig. 4):

- *I-Cover event*: A left ray of the backprojection is tangent to an initial-region disk, with this disk on its right-hand side.

- *I-Leave event*: A right ray of the backprojection is tangent to an initial-region disk, with this disk on its left.

- *I-Left-Vertex event*: The endpoint of a left ray of the backprojection coincides with the entry intersection point[3] of an initial position disk by a disk contained in the goal extension.

- *I-Right-Vertex event*: The endpoint of a right ray of the backprojection coincides

---

[3] When a disk δ_1 intersects another one, δ_2, we define the *entry intersection point* of δ_2 by δ_1, as the point where we enter δ_2 when we move counterclockwisely along the boundary of δ_1. The point where we exit δ_2 is the *exit intersection point.*

with the exit intersection point of an initial position disk by a disk of the goal extension.

- *L-Touch event*: A left ray of the backprojection is tangent to an intermediate-goal disk, with this disk on its left.

- *L-Exit event*: A right ray of the backprojection is tangent to an intermediate-goal disk, with this disk on its right.

- *L-Spike event*: A spike of the backprojection lies on the boundary of a intermediate-goal disk.

- *D-Touch event*: A left ray of the backprojection is tangent to a disk of the goal extension, with this disk on its left.

- *D-Exit event*: A right ray of the backprojection is tangent to a disk of the goal extension, with this disk on its right.

- *O-Touch event*: A left ray of the backprojection is tangent to an obstacle disk, with this disk on its left.

- *O-Exit event*: A right ray of the backprojection is tangent to an obstacle disk, with this disk on its right.

Consider an I-Cover event. Let s be the slope of the left ray tangent to the initial-position disk. The equation of the curve defined by this event is:

$$\theta = d - s + \pi \pmod{2\pi}.$$

Similar linear equations (with slopes equal to ± 1) can be established for the curves defined by the I-Leave, I-Left-Vertex, I-Right-Vertex, L-Touch, and L-Exit, D-Touch, D-Exit, O-Touch, and O-Exit events. The equation of the curve of an L-Spike event is significantly more complicated. However, we can show that it is of the form:

$$\theta = f_{spike}(d),$$

where f_{spike} has a single maximum and at most one intersection with any line of slope ± 1 (see [28] for a proof of these claims). We call f_{spike} a *spike curve*.

The number of I-Cover and I-Leave curves is $O(\ell)$, the number of I-Left-Vertex, I-Right-Vertex, L-Touch, L-Exit, D-Touch, D-Exit, O-Touch and O-Exit curves is $O(\ell^2)$, and the number of L-Spike curves is $O(\ell^3)$. These curves determine an arrangement of $O(\ell^6)$ cells. Fig. 5 shows a cell arrangement with $\theta \in (0, \pi/2)$.

Allowing landmark disks to intersect with obstacle disks would simply require considering additional critical events. It would not change the asymptotic complexity of the cell arrangement, nor the complexity of the planners presented in the following sections. For the sake of simplicity, we will not describe this simple extension here.

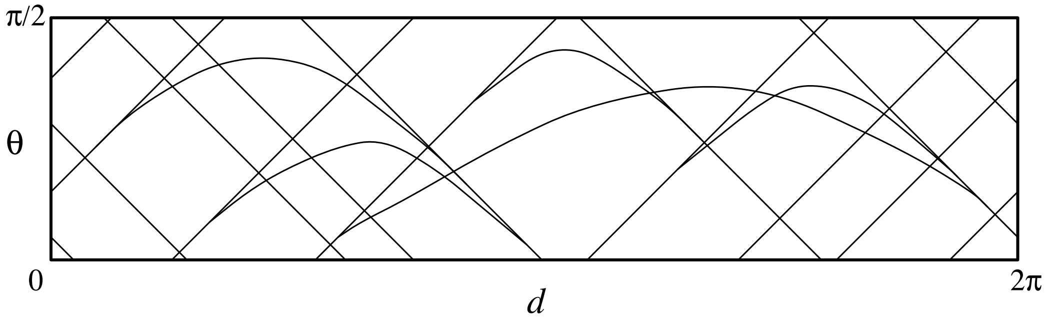


Figure 5: Cell arrangement in the $d\theta$-plane.

6 Planning with Constant Directional Uncertainty

Let us first consider the case where the directional uncertainty is fixed to some given value θ_f. We previously considered this case in [27].

In the $d\theta$-plane the line $\theta = \theta_f$ intersects the critical curves at $O(\ell^3)$ points (critical commanded directions of motion), which partition the set S^1 of values of d into $O(\ell^3)$ cells of dimensions 1 and 0 (open intervals and points). The planning algorithm sketched in Section 4 can proceed iteratively in a kind of breadth-first manner, as follows:

At each iteration, decompose S^1 into cells, select an arbitrary commanded direction of motion in each cell, and compute the corresponding backprojection (along with the initial-region disks that this backprojection contains and the intermediate-goal disks that it intersects). For every computed backprojection B, for every intermediate-goal disk L that this backprojection intersects, select an exit point in the intersection $B \cap L$. Associate the imperfect-control motion command $(d, \theta_f, \mathcal{E})$ with this point, where d is the direction used to compute the backprojection B and $\mathcal{E}$ is the current goal extension. (This command is part of the plan being generated. During plan execution, if the robot ever reaches the landmark region containing the disk L, it will move to the exit point of L in the perfect-control mode; it will then execute the imperfect-control motion command $(d, \theta_f, \mathcal{E})$ associated with this point.) If any of the computed backprojections fully contains the initial region, return success. Otherwise, include all the landmark regions containing one or several intermediate-goal disks intersected by a backprojection in the current goal extension to form the extension for the next iteration.

The computation of a backprojection takes time $O(\ell \log \ell)$. Since there are $O(\ell^3)$ cells, the time complexity of a planning iteration is $O(\ell^4 \log \ell)$. Up to s iterations are required, where $s \in O(\ell)$ is the number of landmark regions. Hence, the algorithm returns a plan, if one exists, in time $O(s\ell^4 \log \ell)$; otherwise, it declares failure in the same time.

After the kth planning iteration ($k = 1, 2, ...$), the goal extension is the largest set of landmark disks from which $\mathcal{G}$ can be attained reliably by executing at most k imperfect-control motion commands. Hence, after at most s iterations, the goal extension eventually contains all landmark disks from which the goal can be reliably attained by executing a finite number of motion commands. At any iteration, if a backprojection of the current goal extension contains $\mathcal{I}$, the planner will find it. Hence the planner is complete.

The outcome of the planner is a "distributed plan" made of imperfect-control motion commands attached to exit points in landmark regions (see Fig. 1). The algorithm may generate several exit points in the same landmark region or disk (if this area or disk is intersected by several backprojections computed at the same iteration). In principle, the planner could keep a single exit point per landmark region, and discard the others. However, it may be preferable to keep several exit points among which the robot can choose at execution time in order to minimize the length of the paths performed in the perfect-control mode in the landmark regions [27].

Several executions of the same plan may lead the robot to perform different sequences of commands, since control errors (in the interval $[0, \theta_f]$) may yield the same command to terminate in one landmark region or another (in the termination set of the command). The number of imperfect-control motion commands that the robot may have to execute is upper-bounded by the number of iterations performed by the planner. The definition of a backprojection and the construction of the cells guarantee that this number is minimal over all possible correct plans for the given initial and goal regions. Furthermore, after the execution of any sequence of steps, the subset of the motion plan that may still be used to attain the goal has the same property. In this sense, the plans produced by the above algorithm are *optimal.*

The above algorithm proceeds in a kind of breadth-first fashion by computing one backprojection per cell at every iteration. One can easily imagine variants based on other search strategies. However, not all strategies yield optimal plans.

The time complexity of the planner, $O(s\ell^4 \log \ell)$, is one order of magnitude greater than the complexity of a similar planner described in [27]. The difference comes from the fact that the above planner recomputes a backprojection from scratch in every cell of the decomposition of S^1. Instead, the planner of [27] computes a first backprojection and incrementally modifies this backprojection as it scans the cells in S^1. However, this requires tracking all the changes in the backprojection topology, yielding 33 different types of events that include the 11 types listed in Section 5. There are significant practical advantages in having less events to consider. In particular, the above algorithm is simpler to implement than the one of [27]. Because it recomputes backprojections from scratch in every cell, it is also less sensitive to floating-point computation errors.

7 One-Step Planning with Controlled Uncertainty

Now let the directional uncertainty θ be controllable by the robot within the given interval $[\theta_{min}, \theta_{max}]$. We first address the one-step plan generation problem. Multi-step planning will be considered in the next section.

To generate a one-step plan, the planner only needs to find a pair (d, θ), with $\theta \in [\theta_{min}, \theta_{max}]$, such that the backprojection of the goal extension $\mathcal{E}(\mathcal{G})$ for the imperfect-control motion command $(d, \theta, \mathcal{E}(\mathcal{G}))$ fully contains the initial region $\mathcal{I}$. Thus, the planner can discard the L-event critical curves in the $d\theta$-plane. It can also discard the O-event critical curves, because these are only used in conjunction with L-Spike event curves.[4] The remaining I- and D-event curves (all straight lines) define an arrangement of $O(\ell^4)$ cells. In every cell, the planner can select an arbitrary pair (d, θ) and compute the corresponding backprojection. In the worst case, the planner scans all the cells. Hence, it returns a plan or declares failure in time $O(\ell^5 \log \ell)$. The resulting planner is complete.

In general, if a one-step plan exists, it is preferable to generate one which allows maximal control uncertainty. The value of θ for such a plan can only be θ_{max} or the θ-coordinate of the intersection of two critical lines. This yields the following planning algorithm: Set θ to θ_{max} and compute the backprojection of $\mathcal{E}(\mathcal{G})$ for each cell of the arrangement intersecting the line $\theta = \theta_{max}$. If one backprojection contains $\mathcal{I}$, return success (and the corresponding value of d). If no backprojection contains $\mathcal{I}$, scan the intersections of I- and L-event lines verifying $\theta \in [\theta_{min}, \theta_{max}]$ in decreasing order of their θ-coordinates. For every intersection point (d, θ), compute the backprojection of $\mathcal{E}(\mathcal{G})$. If this backprojection contains $\mathcal{I}$ return success, otherwise consider the next intersection point. Return failure if all intersections have been considered without success.

Using a sweep-line technique to scan the intersection points, this algorithm takes output-sensitive time $O((\ell^2 + c\ell) \log \ell)$, where $c \in O(\ell^4)$ is the rank (in decreasing order) of the value of θ selected among the θ-coordinates of the $O(\ell^4)$ intersection points of the event lines.

Fig. 2 shows a motion command generated by the above algorithm.

---

[4] O-events serve only to indicate the limits of existence of a spike. For more details see [28].

8 Multi-Step Planning with Controlled Uncertainty

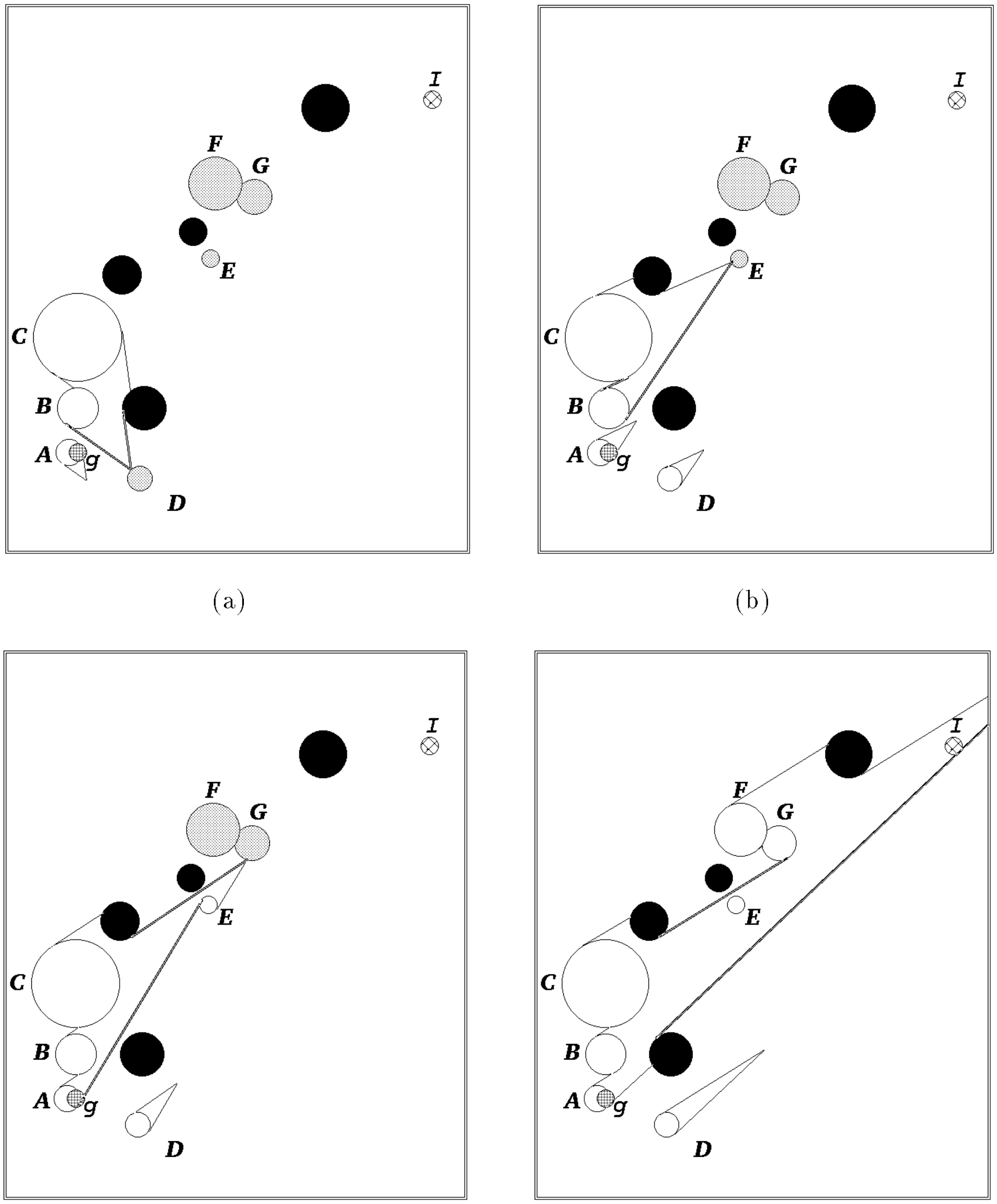
To generate a multi-step plan, we can use the following greedy algorithm:

At every iteration, sweep a line parallel to the d-axis in order to find the highest value of θ such that there exists a backprojection of the current goal extension which either contains the initial region or intersects *one* intermediate-goal disk. In the second case, add the disks in the landmark region intersected by the backprojection to the goal extension, and introduce the corresponding new event curves into the arrangement. Repeat this procedure until the computed backprojection contains all initial-position disks, or no more intermediate-goal disks can be intersected. In the latter case return failure.

At every iteration, the algorithm need not compute the intersections of the L-Spike event curves among themselves, since these intersection points cannot give rise to the highest value of θ we are looking for. (Remember we only seek the *first* intermediate-goal disk to be intersected.) Furthermore, the algorithm must consider intersection points among event curves at most once over all iterations (see proof in [28]). Hence, the total complexity of the planner is $O(s\ell^5 \log \ell)$. The output plan has s steps at most.

Fig. 6 illustrates the operation of this algorithm with an example. The workspace contains 7 landmark disks **A** through **G**, and 4 obstacles. The goal $\mathcal{G}$ intersects with landmark disk **A**. The initial region $\mathcal{I}$ consists of a single disk. The directional uncertainty lies in the interval $[0.0, 0.5]$ radian. This means that we are not interested in uncertainty angles above 0.5 radian because the robot cannot do worse than that. Using $\theta = 0.5$, the planner first finds a correct motion command to reach **A** from the landmark disks **B** and **C**. The extension of goal then consists of **A**, **B**, and **C**. For the commanded direction shown in the upper-left figure and $\theta = 0.41$, the backprojection of this goal extension touches **D**, which is added in turn to the goal extension. **E** is then touched by a backprojection for $\theta = 0.28$ (upper-right figure). **F** is touched by a backprojection for $\theta = 0.32$ (lower-left figure). Both disks **F** and **G** are then added to the goal extension, since they intersect. At this point, all landmark disks are in the goal extension. The initial-position disk is then covered by a backprojection for $\theta = 0.1$ (lower-right figure).

At every iteration, the algorithm maximizes directional uncertainty to achieve the current extension of the goal. In general, the generated plan, if any, would not minimize a given cost function. Generating minimum-cost plans seems intrinsically harder. It may require the selection of a smaller directional uncertainty at an early iteration, if this choice allows larger values of the uncertainty to be selected at subsequent iterations.



(a)

(b)

(c)

(d)

Figure 6: Multi-Step Greedy Planning

9 Implementation and Simulation

We implemented the planners described in Sections 6, 7, and 8 in a program written in the C language and running on a DEC 5000 workstation.

The only major issue in this implementation concerns the computation of the maxima of the L-Spike event curves and their intersections with other event lines. We have not been able to calculate analytical expressions for these points. Therefore, our planners use traditional numerical techniques, which require some care to avoid inconsistent topological results in constructing backprojection.

In order to visualize the plans generated by the planners, we have developed a simple simulator. Imperfect-control motion commands are discretized into short segments and a directional error is randomly selected for each segment. The simulator allows the user to tune segment length and error distribution in the interval $[0, \theta]$, to generate various sample runs.

The examples shown in Fig. 1, 2, and 6 were generated using the planners described in Section 6, 7, and 8, respectively. The respective computation times for these examples were 3.5 minutes,[5] 10 seconds, and 80 seconds. We have run the planner on many other examples. Though our evaluation of the theoretical complexity of the planner is rather high, we were able to handle problems with up to 50 landmark disks reasonably fast (in the order of minutes).

In general, our experiments tend to show that the planners are more efficient than the asymptotic worst-case complexity analysis of the previous sections suggests. We believe that deeper combinatorial analysis of critical curves, cost amortization over iterations, and output-sensitive complexity evaluation should make it possible to produce tighter complexity results.

10 Relevance to Robot Navigation

At this point it is worth looking again at the assumptions made in the problem statement of Section 3. As mentioned earlier, this statement is certainly a simplification of a real mobile-robot navigation problem, but we think it is not oversimplified. Indeed, it captures the essence of the actual problem and, as argued below, the assumptions made are more realistic than it may appear at first glance.

Some assumptions do not correspond to actual limitations of the approach. For example, the restriction of landmark and obstacle regions to being unions of disks could easily be removed to allow these areas to be described as generalized polygons

---

[5] Actually, this time was obtained with the planner described in [27] and mentioned in Section 6.

(regions bounded by simple closed curves made of straight and circular edges). This extension only requires a straightforward adaptation in the construction of the critical curves. The case of intersecting landmark and obstacle regions can be handled with the addition of special critical events, that correspond to the transition of rays sliding on the circumference of disks, to rays anchored at obstacle-landmark region intersection points. The critical curves that are generated by these events are no more complicated than the ones we have already handled.

The fact that the robot is a point in a two-dimensional space is slightly more serious since it limits the configuration space of the actual robot to be two-dimensional. Hence, either the actual robot is circular, or it can only translate. In higher dimensional spaces critical curves become critical hyper-surfaces yielding more complex arrangements that are also harder to compute.

Perhaps the most disturbing assumptions are those used in the definition of the landmark regions, namely that control and position sensing errors are null within these areas, while position sensing is inexistent outside. We now focus our discussion on these assumptions in the context of a mobile robot.

A typical mobile robot uses two techniques to continuously estimate its position: dead-reckoning and environmental sensing. Environmental sensing provides pertinent information only when some characteristic features of the workspace (i.e., "landmarks") are visible by the sensors. Then the robot knows its position with good accuracy. When no or few features are visible, the robot mostly relies on dead-reckoning, which yields cumulative errors that we model by the directional uncertainty cone. Our assumption that sensing outside landmark regions is null is usually conservative, but it does not prevent the robot's navigation system from using all available sensing information at execution time to better determine the robot's current position. (As mentioned in Section 2, the navigation system does not have to use the same model as the planner; it may use a more sophisticated one, if this is possible.) In the worst case, the no-sensing assumption outside landmark regions may only lead our planners to return failure, while reliable plans exist in practice and, possibly, could have been found by more powerful planners able to deal with more sophisticated models.

On the contrary, the assumption that control is perfect in the landmark regions is anti-conservative; we believe, however, that it is a reasonable one, provided that we choose safe features and equip the robot with the right sensors. landmark regions with sharp boundaries can be obtained by introducing artificial landmarks (e.g., radio or magnetic beacons) and/or thresholding an estimate of the sensing uncertainty. For example, the notion of a "sensory uncertainty field" (SUF) is introduced in [38]. At every possible point q in the configuration space, the SUF estimates the range of possible errors in the sensed configuration that the navigation system would

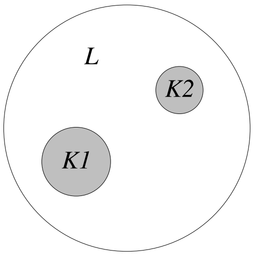


Figure 7: A generalized landmark region L with two kernels K_1 and K_2.

compute by matching the sensory data against a prior model of the workspace, if the robot was at q. The SUF is computed at planning time from a model of the robot's sensing system. Thresholding it yields landmark regions. Uncertainty in the location of a landmark and/or fuzziness of its boundary can be handled by defining a smaller landmark region for our planner.

More interestingly, however, in our first planner (Section 6), perfect control and sensing in landmark regions are not strictly needed. Indeed, once the robot enters a landmark region it is sufficient that it reaches an "exit region" of non-zero measure prior to executing the next imperfect-control command. This region is the intersection of the backprojection that yielded this command with the landmark region. The exit point selected by the algorithm of Section 6 is just one particular point in this area. Position sensing uncertainty in a landmark region could be half the radius of the largest disk fully contained in the exit region of the landmark region without putting plan execution at risk. Thus, although the planner assumed perfect sensing in landmark regions, we can now create these areas by engineering the workspace in such a way that the sensors just provide the information that is needed by the plan (see [18] for a similar idea).

Going further, the definition of a landmark region can even be modified without having to significantly change the planning techniques developed above. For example, we could accept a landmark disk (or generalized polygon) L such that the robot knows at any time if it is inside or outside L, but if it is in L it does not know where. During planning, if a backprojection intersects L, this is not sufficient to include L in the extension of the goal. L must be completely contained in the backprojection. The critical events for which this has to be tested are exactly those used to check the containment of an initial-region disk.

The above variant can be extended into the notion of a generalized landmark region, as follows: A *generalized landmark region* L is a connected subset of the workspace that contains smaller regions K_j, $j = 1, \ldots, r_i$, $r_i > 0$, called the landmark *kernels*. See Fig. 7. Sensing and control in L are such that:

1. If the robot ever enters region L, sensing will reliably tell that it is in L, though it may not tell precisely where in L.

2. Once in L, the robot can reliably navigate into any selected kernel K_j, though it may not accurately attain a given point in K_j.

During planning, if a backprojection contains a kernel K_j (for any j) of L, then L can be included in the extension of the goal.

Notice that this notion of a generalized landmark region does not require sensing or control to be perfect anywhere in the workspace. However, it still does assume perfect state identification, i.e., perfect detection of the robot's entry in a landmark region and its kernels. This remaining assumption seems needed to make planning tractable.

11 Experimentation with a Real Mobile Robot

To support the above discussion we implemented a variant of the planner of Section 6 that accepts generalized landmarks; we designed a simple implementation of this landmark notion; we equipped a mobile robot with a navigation system able to execute plans generated by our planner; and we experimented with this system. We report of this work below.[6]

Landmark Design: Here we describe one possible way to enforce the assumptions made in our planner and described in Sections 3 and 10. The main issue is the creation of the landmark regions.

First, let us briefly describe our robot (although most of the engineering described below is robot-independent). It is a NOMAD-200 from Nomadic Technologies. This is a two-degree-of-freedom nonholonomic zero-turning-radius cylindrical robot with three parallel wheels. The base can translate and rotate independently. During a pure translation, the robot moves along the direction of the wheels. During a pure rotation, it moves around its axis. The robot includes an independently rotating turret mounted above its moving base. This turret is used to carry sensory equipment and an on-board processor (an 80386-based PC in our experiments).

Let us now describe the simple design that we use to create generalized landmark regions in an indoor office-type environment.[7] Landmark regions are generated by

---

[6] The variant of the planner was implemented by Byung-Ju Kang. The landmark design and the corresponding image processing software are due to Craig Becker, Joaquin Salas, and Ken Tokusei. The navigation system was developed by Ken Tokusei.

[7] We have designed several types of landmarks and experimented with two. To our taste, the

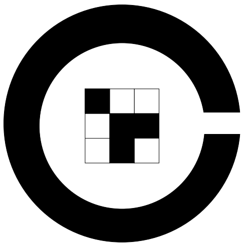


Figure 8: A landmark pattern

placing physical features, called *landmarks*, in the robot workspace. Each landmark induces a area such that, if the robot is in this area, it can detect and recognize the landmark and use sensory data to localize itself relative to the workspace with good precision. The landmark region is precisely this area (or a subset of it). The relatively precise localization of the robot in a landmark region is used to navigate into kernels defined in this region.

The landmarks must be selected to allow a sensing system to achieve three key functions: detection, recognition, and localization. All three must be extremely reliable; in addition, detection must be fast and localization must be precise. These properties led us to design artificial landmarks, rather than using "natural" ones (e.g., corners between walls). Our landmarks are black-and-white patterns (see Fig. 8) fixed to the ceiling at well-defined locations in the workspace coordinate system. These patterns are sensed by an upward-looking camera fixed to the robot and with its axis colinear to the robot's axis of rotation. All landmark patterns are designed to have the same size in the camera image.

A landmark pattern consists of three elements: (1) an outer thick circle (black), (2) an opening into this circle (white), and (3) an inner 3×3 grid of black and white tiles aligned with the opening in the circle. The first two elements are the same for all landmarks. The third is unique to every landmark ($2^9 = 512$ distinct landmarks are thus possible).

During navigation, the on-board processor processes images sequentially. Each image is acquired as a 260×240 gray-level image, and then binarized using an adaptive threshold. The algorithm then detects the connected components in the binary image. Those components which do not match the expected size of the landmark pattern are discarded. For each remaining component, the algorithm overlays a model of the landmark's outer circle and then counts the number of pixel mismatches between this model and the given sub-image. A sufficiently low number of mismatches

---

one reported here is the most elegant.

indicates that a landmark has been found. Once the landmark has been detected, finding its orientation and identifying its unique code are very simple. Computing the position of the robot and the orientation of its turret is also straightforward.

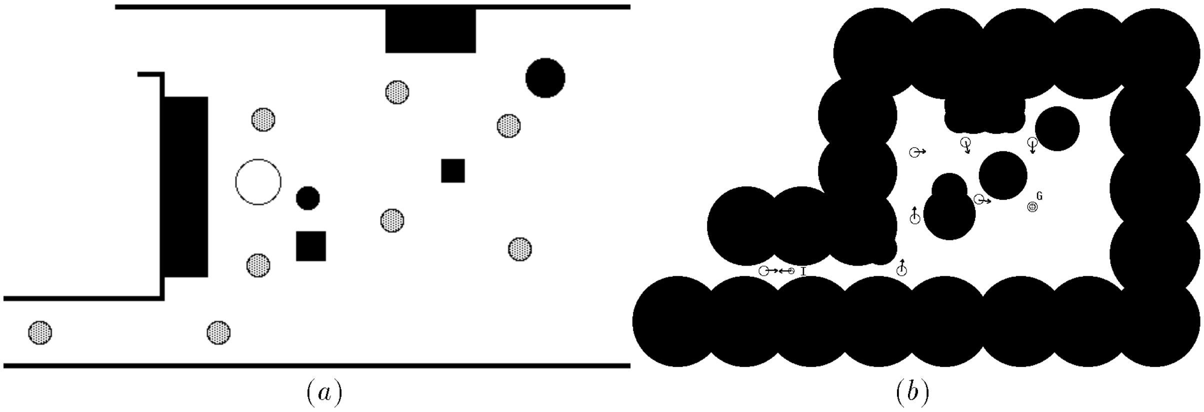
Our algorithm detects a landmark only if the outer circle is almost completely visible. Since the orientation of the camera relative to the landmark pattern can be arbitrary, a specific landmark therefore induces a circular area C in the workspace from which the pattern is guaranteed to be entirely visible. Fortuitous orientations of the camera may allow the landmark to be detected outside this area, but this does not affect the validity of a plan generated by our planner (although it may slightly speed up execution). However, sensing is not instantaneous. If the robot follows a path almost tangent to C, it may traverse and exit C while analyzing an image acquired just before entering C. This leads us to define the actual landmark disk L concentric to C and ε smaller in radius. The difference ε is derived from the maximal velocity of the robot and the time necessary to analyze an image, such that if the robot follows a path tangent to L it is guaranteed to acquire and process an image while it is still in C.

In our experiments, landmark patterns have an outer diameter of 10 inches and are are located roughly 8 feet above the camera. The image of a landmark has a diameter of 92 pixels. The diameter of the landmark region induced by a landmark is 16 inches.

We conducted many experiments in our laboratory under normal office lighting conditions (ceiling fluorescents). With limited hardware, our software requires approximately 0.6 second to process an image. Landmark detection and identification have been 100% reliable, provided that landmark patterns are not positioned adjacent to light fixtures. Measurements revealed that the error in sensing the robot's position in a landmark disk is less than 0.22 inch. The error in the sensed orientation of the robot is within ±3 degrees.

Each kernel in a landmark region is defined as a disk of radius η such that if the robot moves to a sensed position equal to the center of this disk, it is guaranteed to be in the kernel. Thus, η must be greater than or equal to the maximal localization error in a landmark region (0.22 inch). In practice, navigating to a point where the sensed position is very close to the center of the kernel is time consuming (requiring several iterations). Thus, we set η to 0.5 inch. This conservative bound also takes into account the small error in positioning the landmark relative to the workspace and the fact that the floor is not perfectly parallel to the ceiling (causing the camera to tilt slightly relative to the landmark). In our experiments, we defined only one kernel per landmark disk, located at its center (where localization is most precise).

When the robot identifies a landmark in the termination set of an imperfect-control



(*a*) (*b*)

Figure 9: (*a*) The laboratory space; (*b*) The model provided to the planner

command it stops and localizes itself by taking a new image. To move into a kernel in the same landmark disk, the robot orients its wheels toward the centerpoint of the kernel and, using odometric sensing only, it moves by the distance between its measured current position and the kernel center. At the end of this motion, a new image is taken and the robot re-localizes itself relative to the landmark. If necessary, it performs a corrective motion. If the plan requests the robot to move into a kernel located in another landmark disk (in the same connected region), a sequence of intermediate kernels is computed such that every motion between two kernels remains in the total landmark region. The robot moves through these successive kernels.

Once in a kernel (which is not contained in the goal), the robot must execute an imperfect-control command. It aligns its wheels with the commanded direction using its measured orientation and then moves along this direction. The directional error of this motion has two main causes: (1) the error in the measured orientation (within ±3 degrees) and (2) the small bumps in the floor (in our experiments, the floor combines linoleum and carpet separated by metallic joints). Experiments led us to choose $\theta = 4$ degrees as a safe bound.

Finally, near-perfect workspace modeling is obtained through precise prior measurements. Small errors in such measurements are taken into account by slightly growing the obstacles in the model given to the planner.

Navigation: We have developed a software module to execute the plans generated by our planner. It incorporates the image analysis techniques presented in the previous section. Using the planner and this additional software, we have conducted many navigation experiments with the NOMAD-200 robot in our laboratory space. Fig. 9(a) depicts a subset of this space consisting of a large open area and a relatively narrow corridor. The obstacles (walls, tables, chairs, etc.) are shown in black. The landmark patterns are depicted in gray at their actual positions. To give a sense of dimension, the robot is displayed in white at the same scale in an arbitrary position (it has a diameter of 24 inches).

Since the version of the planner used in these experiments only accepts obstacles described as disks, we input a conservatively approximated model of the workspace. This model (obtained after growing all obstacles by the radius of the robot) is shown in Fig. 9(b), along with a plan generated by the planner to go from the initial to the goal region.

Experimentation has shown that our complete navigation system is extremely reliable. The system is capable of navigating the robot between landmark regions for the entire battery life of the robot—just under one hour. It is clear, however, that we need to perform further experiments in larger workspaces over longer periods of time.

Among the several difficulties encountered, two are worth mentioning. Initially, we placed some patterns too close to lighting fixtures, causing incorrect detection or recognition due to assumptions made by our image processing algorithm. Moving the patterns away from the lights solved the problem. Also, localization in some landmark regions was not as good as predicted. This is caused by small, local slopes in our floor. To resolve this, we avoided placing landmarks over sloping areas.

These two constraints on the possible locations of the landmarks generated new difficulties in constrained parts of the workspace, mainly in the narrow corridor. We could not place enough landmarks to allow the robot to reliably navigate from one end of the corridor to the other. In fact, using a safe value of θ led the planner to fail finding a motion plan through the corridor. Using a smaller (but unsafe) value of θ resulted in unreliable plans that sometimes resulted in collisions. To deal with this problem we developed a refined version of the image analysis techniques to compute the robot's orientation with greater accuracy (since this is the main cause of errors in executing inperfect-control commands). By extracting contours at a sub-pixel resolution we were able to reduce θ from 4 to 2 degrees, which was sufficient to navigate reliably through the corridor. Smaller values could even be possible by designing a landmark pattern containing a better orientational feature or by equipping the camera with a zoom to get a higher-resolution image of the landmark, once it has been detected.

Extracting contours at sub-pixel resolution takes more time than working at pixel resolution. Since we do not want the robot to stop when this is not needed, we use two values of θ, θ_1 (the directional uncertainty with the simpler image analysis technique) and θ_2 (the uncertainty with the more sophisticated technique). The planner uses θ_1 until it fails (if this eventually happens). $\theta_1 > \theta_2$. If an iteration fails (i.e., no new landmark region can be added to the goal extension), it is repeated with θ_2. If it is then successful, the planner shifts back to θ_1 at the next iteration. And so on. The planner associates the value of directional uncertainty, θ_1 or θ_2, to every imperfect-control command associated with a landmark region. At navigation time, this value is used to decide whether the more sophisticated image analysis techniques must be run, or not.

The above development is a good illustration of the advantage of treating directional uncertainty as a variable parameter in a planner. In fact, instead of considering only two values of θ, we could use the planner of Section 8 and generate plans that maximize the acceptable directional uncertainty at every step. Then, with appropriate image analysis techniques (perhaps involving the control of a zoom), the robot could spend minimal time in every landmark kernel to determine its orientation with an error guaranteed to be contained in the acceptable uncertainty interval. However, we have not conducted such experiments yet.

12 Other Extensions

In Section 10 we mentioned several important, but rather straightforward extensions of the current planners. Here, we briefly discuss other possible extensions.

Dealing with Unexpected Obstacles: Assume that the workspace may contain unexpected obstacles, i.e., obstacle regions that are not in the planner's model. Assume, further, that the robot sensors can detect the unexpected obstacles interfering with a motion command just before this command is executed. (By interfering obstacle, we mean one that intersects the set of all positions that the robot may traverse or reach during the execution of the command. This set is called the *forward projection* of the motion command [5, 7, 12]. A slight variation of the algorithm computing backprojections can be used to compute forward projections.)

Fig. 10 shows an example with a single goal-extension disk in white, an intermediate-goal disk in grey, and an unexpected obstacle in black. If θ_f is the given constant directional uncertainty of the robot, the planner of Section 6 computes a backprojection $\mathcal{B}_f$ of the goal extension that intersects the intermediate-goal disk and provides the corresponding value of d, e.g., the direction d_f shown in the figure. However, if the robot executes a motion commanded along d_f starting from any point in the intersection of $\mathcal{B}_f$ with the intermediate-goal disk, it may hit the unexpected

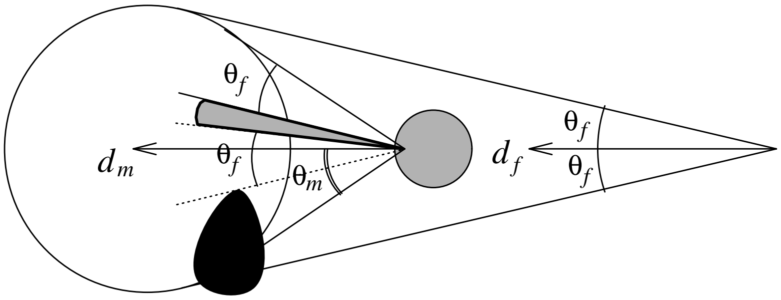


Figure 10: Least commitment navigation technique.

obstacle.

Instead, we can compute the maximal value of θ, θ_m, for which there exists a backprojection $\mathcal{B}_m$ of the goal extension that intersects the intermediate-goal disk, yielding a direction of motion d_m. Let $C(d_m, \theta_m)$ be the cone of half-angle θ_m about d_m. Let the robot move in the perfect-control mode to the intersection of $\mathcal{B}_m$ with the intermediate-goal disk before executing an imperfect-control motion command of uncertainty θ_f toward the goal extension. In the absence of unexpected obstacles, the axis of any cone of half-angle θ_f contained in $C(d_m, \theta_m)$ is a commanded direction of motion guaranteed to achieve the goal extension with directional uncertainty θ_f. In the presence of unexpected obstacles, as is the case in Fig. 10, if there exists a cone of half-angle θ_f included in $C(d_m, \theta_m)$ but not intersecting the unexpected obstacles, its axis is a valid commanded direction. The grey cone in Fig. 10 contains all valid commanded directions for this example.

Generating Probabilistic Plans: The planners of Sections 6 and 8 are complete for multi-step plans. If they fail to generate a plan, the input problem has no correct solution. However, before failing, they may associate imperfect-control motion commands to landmark regions from which the goal can be reliably achieved. In other words, these planners always return a plan. If the problem admits no solution, the plan is incomplete, i.e., no motion command is associated with the initial region. The robot may nevertheless attempt to reach a landmark region with an associated command by executing a random walk. For a bounded planar workspace, the probability that such a motion enters a set of disks converges toward 1 as time grows. The larger this set, the faster the convergence.

Another way to deal with failure is to allow uncertainty to decrease below its minimal value until a backprojection of the current goal extension intersects an intermediate-goal disk or contains the initial region. Any motion command planned in this way

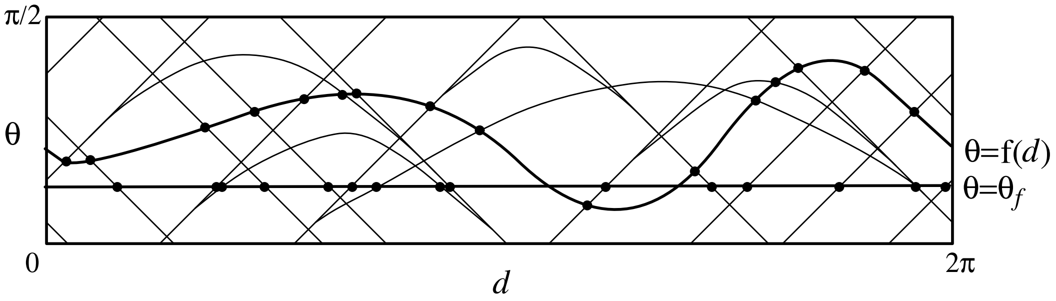


Figure 11: Modeling anisotropic directional uncertainty

is no longer guaranteed to attain the goal extension it is aimed at. This approach thus yields the concept of a non-correct plan that maximizes probability of success.

One major drawback of a plan generated as above is that its execution may fail in a non-recognizable way. Indeed, a non-correct motion may miss all the disks in its termination set and continue for ever. Introducing some awareness of time is one way to address this drawback. Another approach, inspired by Donald's EDR strategies [11], is to make sure that every non-correct imperfect-control motion command inserted in the plan will either succeed or fail recognizably. Such commands have a termination set that they are guaranteed to reach, but some disks in this set are not part of the goal extension at the time they were selected. The generation of non-correct plans that recognize failure, while maximizing probability of success, is an interesting topic still requiring additional research.

Planning with Anisotropic Uncertainty: In some applications, the directional uncertainty θ depends on the commanded direction of motion. For example, this can happen when a wheeled robot moves on a carpet, or more generally for motions in any flow field (e.g., a river). If we know the function $\theta = f(d)$, then we may compute its intersections with the event curves (see Fig. 11). These intersections correspond to critical directions that partition the curve $\theta = f(d)$ in regular subsets (open intervals and points). We need to compute only one backprojection per subset.

13 Conclusion

This paper describes sound, complete, and polynomial planning algorithms in the presence of uncertainty. The key notion underlying these algorithms is that of a landmark region where control and position sensing are perfect. An important aspect of our algorithms, not investigated before, is that they allow uncertainty (more precisely, directional uncertainty) to vary over a continuous domain. Varying uncertainty has multiple applications. If the robot can control uncertainty (but with some cost associated with reduced uncertainty), the planner can try to generate a minimal-cost plan. If uncertainty is not controllable, varying uncertainty can be used to generate probabilistic plans or plans that can deal with unexpected obstacles.

The main technical result presented in this paper is that planning does not require computing all the backprojections of the successive goal extensions, for all directions of motion d in S^1 and all values of the directional uncertainty θ in $[\theta_{min}, \theta_{max}]$. The $d\theta$-plane can be decomposed by critical curves into a polynomial collection of cells such that all backprojections in a cell are equivalent relative to planning. This result directly yields our planners.

In general, the task of a motion planner in the presence of uncertainty is threefold: (1) cover the set of configurations reachable by the robot by a finite collection of states, (2) synthesize functions able to recognize the achievement of these states, and (3) plan motion commands to move from state to state. The most important effect of introducing landmarks in the configuration space is the reduction of the interdependence between these three activities. Actually, the states that the robot may attain are the landmark regions and the complement of their union. The recognition functions are also trivialized by the definition of the landmarks. Hence, planning is reduced to finding appropriate commands to move from landmark to landmark. To construct the termination sets of commands, the planner may have to consider any combination of states. In theory, there is an exponential number of such combinations, but the algorithms presented in this paper show that only a growing sequence of combinations (the successive goal extensions) must actually be considered. The length of this sequence is bounded by the number of landmark regions.

The simple landmark design presented in this paper and the associated experimental work show that the assumptions made in our planner can be enforced at a relatively low cost. Nevertheless, the creation of landmarks requires some prior engineering. Actually, what the planner does not have to do, e.g., selecting a set of states and synthesizing state-recognition functions, is hidden either in this engineering work, or in the fact that our planner only solves a limited class of problems. One issue is therefore: How can we minimize the necessary engineering work? For instance: How could we minimize the number of landmarks necessary to perform a given family of tasks? This issue still requires additional research.

References

[1] Ayache, N., *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception*, The MIT Press, Cambridge, MA, 1991.

[2] Brafman, R.I., Latombe, J.C., and Shoham, Y., Towards Knowledge-Level Analysis of Motion Planning, *Proc. of the 11th Nat. Conf. on Artificial Intelligence*, AAAI, Washington, DC, 670-675, July 1993.

[3] Briggs, A.J., An Efficient Algorithm for One-Step Planar Compliant Motion Planning with Uncertainty, *Proc. of the 5th Annual Symp. on Comp. Geom.*, Saarbrücken, Germany, 1989.

[4] Brooks, R.A., Symbolic Error Analysis and Robot Planning, *Int. J. of Robotics Research*, 1(4):29-68, 1982.

[5] Buckley, S.J., *Planning and Teaching Compliant Motion Strategies*, Ph.D. Dissertation, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1986.

[6] Canny, J.F. and Reif, J.H., New Lower-Bound Techniques for Robot Motion Planning Problems, *Proc. of the 28th SYMP. on the FOCS*, Los Angeles, CA, 49-60, 1987.

[7] Canny, J.F., On Computability of Fine Motion Plans, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 177-182, 1989.

[8] Chapman, D. and Agre, P.E., Abstract Reasoning as Emergent from Concrete Activity, *Reasoning about Actions and Plans*, edited by Georgeff, M.P. and Lansky, A.L, Morgan Kaufmann Publishers, Los Altos, CA, 411-424, 1986.

[9] Crowley, J.L., World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 674-680, 1989.

[10] Dakin, G.A. and Popplestone, R.J., Augmenting a Nominal Assembly Motion Plan with a Compliant Behavior. *Proc. of the 9th Nat. Conf. on Artificial Intelligence*, Anaheim, CA, 653-658, July 1991.

[11] Donald, B.R., A Geometric Approach to Error Detection and Recovery for Robot Motion Planning with Uncertainty, *Artificial Intelligence J.*, 52(1-3):223-271, 1988.

[12] Donald, B.R., The Complexity of Planar Compliant Motion Planning Under Uncertainty, *Algorithmica*, 5:353-382, 1990.

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

[14] Drummond, M., Situated Control Rules, *Proc. of the First Int. Conf. on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, Los Altos, CA, 103-113, 1989.

[15] Dufay, B. and Latombe, J.C., An Approach to Automatic Robot Programming Based on Inductive Learning, *Int. J. of Robotics Research*, 3(4):3-20, 1984.

[16] Erdmann, M., *On Motion Planning with Uncertainty*, Tech. Rep. 810, AI Lab., MIT, Cambridge, MA, 1984.

[17] Erdmann, M., Randomization in Robot Tasks, *Int. J. of Robotics Research*, 11(5):399-436, 1992.

[18] Erdmann, M., *Towards Task-Level Planning: Action-Based Sensor Design*, Tech. Rep. CMU-CS-92-116, Dept. of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA, February 1992.

[19] Fox, A. and Hutchinson, S., *Exploiting Visual Constraints in the Synthesis of Uncertainty-Tolerant Motion Plans*, Tech. Rep. UIUC-BI-AI-RCV-92-05, The University of Illinois at Urbana-Champaign, October 1992.

[20] Friedman, J., *Computational Aspects of Compliant Motion Planning*, Ph.D. Dissertation, Report No. STAN-CS-91-1368, Dept. of Computer Science, Stanford University, Stanford, CA, 1991.

[21] Hutchinson, S., Exploiting Visual Constraints in Robot Motion Planning, *Proc. of the IEEE Int. Conf. of Robotics and Automation*, Sacramento, CA, 1722-1727, 1991.

[22] Gottschlich, S.N. and Kak, A.C., Dealing with Uncertainty in CAD-Based Assembly Motion Planning, *Proc. of the 9th Nat. Conf. on Artificial Intelligence*, Anaheim, CA, 646-652, July 1991.

[23] Latombe, J.C., *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[24] Latombe, J.C., Lazanas, A., and Shekhar, S., Robot Motion Planning with Uncertainty in Control and Sensing, *Artificial Intelligence J.* 52(1):1-47, 1991.

[25] Laugier, C. and Théveneau, P., Planning Sensor-Based Motions for Part-Mating Using Geometric Reasoning Techniques, *Proc. of European Conf. on Artificial Intelligence*, Brighton, UK, 1986.

[26] Lazanas, A. and Latombe, J.C., Landmark-Based Robot Navigation, *Proc. of the 10th Nat. Conf. on Artificial Intelligence*, San Jose, 697-702, July 1992.

[27] Lazanas, A. and Latombe, J.C., *Landmark-Based Robot Navigation*, Tech. Rep. STAN-CS-92-1428, Dept. of Computer Science, Stanford Univ., Stanford, CA, 1992 (to appear in *Algorithmica*).

[28] Lazanas, A., *Motion Planning with Uncertainty: A Landmark Approach*, Ph.D. Dissertation, Tech. Rep., Dept. of Computer Science, Stanford Univ., Stanford, CA, 1994 (in preparation).

[29] Leonard, J.J. and Durrant-Whyte, H.F., Mobile Robot Localization by Tracking Geometric Beacons, *IEEE Tr. on Robotics and Automation*, 7(3), 376-382, 1991.

[30] Levitt, T.S., Lawton, D.T., Chelberg, D.M. and Nelson, P.C., Qualitative Navigation, *Proc. of DARPA Image Understanding Workshop*, Los Angeles, CA, 447-465, 1987.

[31] Lozano-Pérez, T.,*The Design of a Mechanical Assembly System*, Tech. Rep. AI-TR 397, Artificial Intelligence Laboratory, MIT, 1976.

[32] Lozano-Pérez, T., Mason, M.T. and Taylor, R.H., Automatic Synthesis of Fine-Motion Strategies for Robots, *Int. J. of Robotics Research*, 3(1):3-24, 1984.

[33] Mahadevan, S. and Connell, J., *Automatic Programming of Behavior-based Robots using Reinforcement Learning*, Research Rep., IBM T.J. Watson Research Center, Yorktown Heights, NY, 1990.

[34] Mason, M.T., Automatic Planning of Fine Motions: Correctness and Completeness, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Atlanta, GA, 492-503, 1984.

[35] Natarajan, B.K., The Complexity of Fine Motion Planning, *Int. J. of Robotics Research*, 7(2):36-42, 1988.

[36] Pertin-Troccaz, J. and Puget, P., Dealing with Uncertainty in Robot Planning Using Program Proving Techniques, *Proc. of the 4th Int. Symp. on Robotics Research*, Santa-Cruz, CA, 455-466, 1987.

[37] Schoppers, M.J., *Representation and Automatic Synthesis of Reaction Plans*, Ph.D. Dissertation, Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1989.

[38] Takeda, H. and Latombe, J.C., Sensory Uncertainty Field for Mobile Robot Navigation, *Proc. of IEEE Int. Conf. on Robotics and Automation*, Nice, 2465-2472, May 1992.

[39] Taylor, R.H., *Synthesis of Manipulator Control Programs from Task-Level Specifications*, Ph.D. Dissertation, Dept. of Computer Science, Stanford University, 1976.

[40] Zhang, Z. and Faugeras, O., A 3D World Model Builder with a Mobile Robot, *The Int. F. of Robotics Research*, 11(4):269-285, 1992.