Symbolic Feedback Control for Navigation

Sean B. Andersson, *Member, IEEE*, and Dimitrios Hristu, *Senior Member, IEEE*

***Abstract*—We discuss the generation of symbolic feedback control sequences for navigating a sparsely-described and uncertain environment, together with the problem of sensing landmarks sufficiently well to make feedback meaningful. We explore the use of a symbolic control approach for mitigating the lack of a detailed map of the environment and for reducing the complexity associated with finding control laws which steer a control system between distant locations. Under our language-based approach, control inputs take the form of symbolic strings. The decision process that generates those strings is guided by estimates of the vehicle's location within a set of important landmarks and by the statistical effectiveness of each string. This arrangement, and in particular the symbolic nature of the control set, allows us to formulate and solve a class of optimal navigation problems which would be exceedingly difficult to handle if approached at the level of sensors and actuators. Our approach is illustrated in a series of numerical indoor navigation experiments.**

***Index Terms*—Markov processes, mobile robot motion-planning, motion description languages, symbolic control.**

I. Introduction

THE design of feedback laws that accomplish seemingly straightforward motion control tasks, such as navigation in environments of even moderate complexity, arguably remains a persistent challenge for automatic control. One of the reasons for this is that the achievements of systems theory vis-a-vis motion control are often made possible only by the most severe simplifications. There are at least two observations to be made in support of this assertion, having to do with the specialized nature of existing control design tools, and with the extent to which the environment is (or is not) state-space like. First, many of the available results—despite their elegance and importance—are established on the basis of structural requirements that capture only a minority of interesting and realistic situations. To apply them, one must first check whether the system in question fits in one of a variety of special classes (e.g., systems that are drift free, differentially flat [1], [2], or can be kinematically decoupled [3], [4]). Where such structure is not available, the task of specifying an appropriate feedback controller is often made difficult by technical challenges and by the richness of the set of mappings from sensor to actuator signals. A second problem to overcome has to do with the fact that most available control design tools are *local*, in the sense that they concern systems which evolve in state-space like environments. In many practical settings, this is a significant restriction: for example, knowing that a mobile robot is controllable does not reveal whether it can be steered between two locations in a realistic environment, such as an office building. Attempting to devise a state feedback law that will steer a robot through a moderately sized structure with doors, human traffic, elevators, etc., quickly leads to an intractable problem, and it becomes "evident" that one must partition the task into intermediate pieces. This partitioning can—and must—include the environment, as we will argue in the sequel, because the "usual" descriptions in terms of geometric maps endowed with global coordinate systems become difficult to generate and manage as the size of the environment grows.

Efforts to overcome the complexity of specifying motion control tasks have included the development of so-called *motion description languages*, beginning with [5], [6] and its descendents [7]–[9], as well as related work on maneuver-based planning [10], [11]. These works, together with related efforts [12]–[16] to develop suitable abstractions for describing systems and control policies, can be regarded as part of a broader research program that attempts to approach systems and control without always having to resort to analysis and design at the level of individual sensors and actuators. In symbolic control, one intentionally reduces their choice of feedback control laws to a small (usually finite) collection of specialized and straightforward-to-design "primitives". The only allowable control inputs are compositions of symbols which are identified with primitives and are interpreted by the control system to determine its actuator signals. Aside from reducing the complexity of a motion control problem, this restriction to a denumerable input set also carries intuitive appeal because tokenized descriptions are part of everyday experience; for example, a set of directions might include terms like "exit the room, turn right, walk down the hallway, enter the third door on the right" as opposed to a geometric description of the path one must take. In some cases, this viewpoint has led to new ways of thinking about basic systems concepts, such as feedback [17]. In others (including the work presented here), it can expand the domain of applicability of existing tools, by re-examining control problems at a more abstract level [18]–[21].

This paper explores the use of symbolic control for steering a control system (e.g., a robot or other autonomous vehicle) between two locations in an obstacle-rich, partially-known environment which is "large" relative to the vehicle's size and sensing range. We circumvent some of the difficulties outlined above by constructing a symbolic feedback process which accepts tokenized outputs produced from the vehicle's sensor readings, and generates tokenized inputs which the vehicle

Manuscript received October 28, 2004; revised August 4, 2005. Recommended by Guest Editor G. Pappas. This work was supported by the National Science Foundation under Grant No. EIA0088081 and by the Army Research Office under ODDR&E MURI01 Grant No. DAAD19-01-1-0465, (Center for Communicating Networked Control Systems, through Boston University). The first author was supported in part by a fellowship from the ARCS foundation.

S. Andersson is with the Department of Aerospace and Mechanical Engineering, Boston University, Boston, MA 02215 USA (e-mail: sanderss@bu.edu).

D. Hristu-Varsakelis is with the Department of Applied Informatics, University of Macedonia, Thessaloniki 54006, Greece (e-mail: dcv@uom.gr).

Digital Object Identifier 10.1109/TAC.2006.876956

interprets in order to navigate. In the sequel, we will describe this process, together with appropriate abstractions for the vehicle's inputs, outputs, and state. In particular, inputs will be drawn from the strings of a motion description language, maneuver automaton or other control-oriented language [9], [11], [22]. The same input symbols will be used to implicitly describe the environment itself. We will forgo the use of geometric information and will instead view the environment as a collection of possibly disjoint regions which are "connected" by symbolic instructions on how to reach one from another. Doing so will introduce a natural quantization of the vehicle's location by "decomposing" the environment into two classes of subsets: i) those areas that are state-space like and thus are easily navigable using simple feedback control laws and ii) those that are not, and thus require both a detailed description and more sophisticated trajectory planning tools. Finally, the vehicle's location will be described by output symbols which are generated from "raw" sensor data via signal processing or other filtering. Our approach will allow us to translate what appears to be an intractable global navigation problem into a Markov decision process (MDP) [23] where we will be able to generate suitable control strings via dynamic programming (DP). This "lifting" of the navigation problem to the symbolic domain, together with the integration of the tools that make it possible (including language-based control, MDPs and DP), are the main contributions of this paper.

Unlike previous works on language-based control, here the control instructions have an imprecise effect on the evolution of the robot due to sensor, actuator and environment uncertainties. In addition to the non-deterministic effects of the control symbols, our approach takes into account the ambiguity introduced by the fact that many regions of the environment may look similar when using a modest set of observations. Our work complements other recent approaches to symbol-level planning for deterministic systems [10]; besides the presence of uncertainty, the challenge here stems primarily from the complexity of the environment as opposed to that of the vehicle's dynamics.

Other related research includes approaches to localization that combine feature-based maps and Markov chains [24], as well as other work on autonomous navigation. In [25], a similar navigation problem is solved by "tiling" the environment with small regions on which the robot's state is quantized. The collection of all quantized states is used to construct a partially observable Markov chain. Steering the robot between two location involves storing quantized instructions at each state of that Markov chain. That approach tends to generate a very large number of states. Here we are interested in a more "global" version of the navigation problem in environments with widely separated features and little or no geometric information regarding the intermediate space or the physical relationships between landmarks [24], [25]. Furthermore, the number of Markov states we will require is a function of the number of landmarks, or "regions of interest" in the environment and independent of the environment's physical dimensions.

In [26], an MDP formulation was used to compute expected shortest paths in a feature-based environment whose representation is a special case of what is discussed here. An important difference between that work and our approach relates to the way uncertainty enters into the navigation problem. In [26], passage from one feature/landmark to another is either possible or not, and the robot is uncertain as to which case applies. Here, the robot knows which regions can be accessed from any location, however the presence of sensor, actuator and environment uncertainty make it difficult to know which region will be reached upon completion of a control program.

Posing the navigation problem in large environments where features and regions of interest are far apart and separated by areas with few distinguishable features is central to our approach, as it is those settings that reveal many of the complications which we have highlighted. Within areas of limited size, existing path planning and localization tools can be quite effective (e.g., [27]–[29]) as can be a variety of other trajectory planning methods [30]–[32]. What is discussed here is therefore aimed not at replacing existing results, but rather at extending their domain of applicability via synthesis of feedback control sequences that will allow a vehicle to span large distances.

The next section discusses the main elements necessary for constructing a symbolic feedback controller for navigation, and describes how linguistic descriptions of motion control tasks can be naturally combined with landmark-based descriptions of the environment. Doing so leads to a formulation of symbolic navigation and localization problems that is both useful and more abstract than traditional differential equation-based methods. In Section III we show how the higher level of abstraction adopted here makes it possible to solve navigation problems efficiently by drawing on existing tools, including MDPs and DP. In particular, we derive optimal control policies for navigating from one landmark to another under sensor and actuator uncertainty. Section IV contains a set of numerical experiments that illustrate the effectiveness of our approach.

II. Symbolic Navigation and Control

Consider an underlying physical system (equipped with a set of actuators and short-range sensors) for which we want to specify a motion control task. Let the system's evolution be governed by

$$dx = f(x,u)dt + G(x)dw \quad y = h(x) + \nu \tag{1}$$

where $x \in \mathcal{X}^n$, an n-dimensional manifold, $u = u(t, y(t)) \in \mathbb{R}^m$ may be an open loop input or a feedback law, $y(t) \in \mathbb{R}^p$, and G is a matrix whose columns are vector fields in $T\mathcal{X}^n$. The system is subject to sensor and actuator noise, $\nu(t)$ and $w(t)$, respectively. The noise processes are taken to be bounded and independent. We assume that $\mathcal{X}^n$ may contain obstacles and that there may be subsets of $\mathcal{X}^n$ that are uncertain or unknown. For simplicity we will assume that (1) is controllable. Doing so will allow us to focus on challenges which arise because of the complexity of the environment and the resulting difficulty in selecting effective control laws, as opposed to structural deficiencies in the dynamics.

A. *Input Symbols and Language-Based Control*

We are interested in solving a version of the following navigation problem:

Problem 1: Find a feedback control policy $u(t,y) : \mathbb{R}_+ \times \mathbb{R}^p \to \mathbb{R}^m$ that steers (1) from an initial value x_0 to a neighborhood of a final state $x_f \in \mathcal{X}^n$, in finite time, assuming that such a trajectory exists.

To help manage the complexity that can arise as a result of the assumptions made on $\mathcal{X}^n$ and (1), and to proceed with our plan for a symbolic controller, we will choose to steer (1) by supplying it with input tokens which will be identified with various control laws of the form $u(t,y)$, together with conditions under which those laws should be altered based on sensor data. We will use the term *symbolic plan* when referring to a string of such tokens. To construct symbolic plans, we will require a control-oriented language capable of supporting such an encoding. Examples include MDLe [8], [9], the maneuver automata of [11] or the language CCL [22]. In the discussion that follows we will often refer to MDLe, which we describe briefly in the next paragraph. We do this for the sake of concreteness and because that language was used to implement our symbolic controllers of Section IV. However, our approach to global navigation is independent of the choice of a particular language.

MDLe is a formal language [33] which can describe hybrid motion control tasks and which allows one to compose complex control laws from simpler ones. We let $\mathcal{U} \subset \{u | u : \mathbb{R}_+ \times \mathbb{R}^p \to \mathbb{R}^m\}$ be a finite set of feedback control laws (including the trivial $u_{\text{null}} \equiv 0$), and $\mathcal{B} \subset \{\xi | \xi = (\psi \texttt{ AND } (t \leq T)), \psi : \mathbb{R}^p \to \{0,1\}, T \in \mathbb{R}_+^* \cup \{\infty\}\}$, be a finite set of boolean functions of the output variables and time, including the null function $\xi_{\text{null}} \equiv 1$. Elements of $\mathcal{B}$ are referred to as *interrupt* functions. MDLe strings are formed using the alphabet defined by the sets $\mathcal{B}$ and $\mathcal{U}$, together with the special symbols "(", ")" and ",". The simplest MDLe strings, termed *atoms*, are pairs of the form (ξ, u), where the control u is selected from $\mathcal{U}$ and the interrupt ξ from $\mathcal{B}$. To *evaluate* or *run* the atom (ξ, u) means to apply the input u to the control system until the interrupt function ξ is "high" (logical $\texttt{1}$). Under MDLe, plans are composed by stringing together a sequence of atoms, and associating to that string an interrupt function. For example, evaluating the plan $b = (\xi_b, (\xi_1, u_1)(\xi_2, u_2))$ means evaluating the atom (ξ_1, u_1) followed by the atom (ξ_2, u_2) until the interrupt function ξ_b returns logical $\texttt{1}$ or (ξ_2, u_2) has terminated, whichever occurs first. Plans themselves can be composed together to yield higher-level strings, e.g., $(\xi_4, (b, (\xi_3, u_3)))$. See [9] for the production rules that define MDLe's grammar and a discussion of the expressive power of the language. An MDLe compiler is described in [8].

B. Symbolic Environment Descriptions

In the context of autonomous navigation, the description of the environment itself is nontrivial, especially when the latter is partially known and covers a domain at many times the scale of the vehicle's sensing range. Typically, an autonomous vehicle may navigate or localize itself by means of a map (M, ϕ), where M is a coordinate "patch" of a certain portion of the environment (perhaps all of it) and $\phi : M \to \mathbb{R}^n$ are coordinate functions defined on M. Maps with global coordinate systems may be difficult to obtain because they require the robot to "discover" the global geometry from sensor measurements. At the same time, describing the world as a collection of smaller, overlapping coordinate patches requires the vehicle to store a multitude of coordinate changes that relate overlapping regions; if a region cannot be uniquely identified from sensor data then navigation and localization become difficult, as one would be unsure which coordinate change(s) to apply during navigation. Finally, much of the detail stored in a typical map may be irrelevant, because there may be regions whose structure makes trajectory planning straightforward, or which are never visited.

Given that the vehicle may often start out with little or no knowledge of its surroundings, much of the work in describing M has focused on so-called simultaneous localization and map-building (SLAM) methods (see, for example, [34] and references therein). For large environments, such methods are typically more successful at producing maps which are *topologically accurate* and less so at discovering precise geometric relationships between terrain features [35]. This suggests that for the purposes of describing the environment, as well as specifying motion control tasks, it might be better to reserve precise descriptions for regions of limited size centered around easily distinguishable features. We will refer to such regions as *landmarks*. More precisely, a *landmark* will be defined by a pair $L = (M, \phi)$, where the map M and coordinate functions ϕ are as specified above. We will assume that there are n_L such landmarks, $L_i = (M_i, \phi_i), i = 1, 2, \ldots, n_L$, and that they are physically distant from one another (i.e., $M_i \cap M_j = \emptyset$). The set of landmarks will be denoted by $\mathcal{L} = \{L_i\}$. We note that our definition of a landmark differs from those commonly found in the robotics literature, where landmarks are typically understood as terrain features with specific coordinates. A landmark may include an area that is difficult to navigate (hence it requires a detailed description) or a region which is especially relevant to the tasks the robot is to perform. For example, landmarks could take the form of occupancy grid maps [36] constructed around GPS coordinates, visual, or sonar cues.

We do not assume that the coordinate systems ϕ_i are referenced to any global coordinate system or that the robot knows how any two such coordinate systems are related. As we have mentioned in Section I, the navigation problem can be solved effectively within the confines of a landmark through the use of a variety of map-based path-planning and localization techniques. Thus, we will take for granted the ability to plan trajectories and navigate "locally" on any M_i. Here we are interested in the *global problem* of navigating *between* landmarks, i.e., steering the robot from one landmark L_i to another L_j, and with the terrain separating L_i from L_j being at best approximately known.

One of the benefits of landmark-based descriptions of an environment is that they can be parsimonious in the amount of information required to specify them. For example, while driving, it is usually helpful to have a detailed description of highway on/off ramps or street intersections; on the other hand, proceeding from one intersection to another mainly involves steering to keep the vehicle in the proper lane. To create a map of the global environment, landmarks may only need to be related approximately, relying on the fact that they can be easily detected once one is "close enough" to any of them. Symbolic descriptions of control and control-oriented languages fit naturally within this framework because they can be used to relate landmarks *not geometrically, but in terms of what one must do to get from one to another*. Our approach will therefore be to replace—whenever possible—the details of a map by feedback programs, so that navigation between landmarks can be accomplished by executing a series of symbolic plans that lead to the desired landmark. In the absence of uncertainty, the environment could thus be visualized as a directed graph whose nodes correspond to landmarks and whose edges are identified with control strings.

In this work we assume that the set of landmarks and an appropriate collection of control plans are both given[1] and constant, although one can easily consider modifications in which landmarks and plans are added or deleted. The control plans may be determined in a variety of ways, including prior exploration of an area, or off-line planning; they must guarantee that upon their completion the vehicle has reached the interior of some landmark (i.e., its state lies in one of the maps M_i). A simple way of accomplishing this of course, is to "tile" the world with landmarks, as in [24], [25]. Here we take a more economical approach, which aims to avoid tiling by selecting plans carefully. For example, in an office environment it may be possible to create plans which ensure the system will always end up inside an office rather than in a hallway, although because of changes in the environment, such as people opening or closing their doors, the particular office cannot be specified with certainty.

Given a set of landmarks and a collection of symbolic plans, motion control problems can now be posed at a symbolic level, with control inputs specified in a control language of choice. Thus, Problem 1 can be restated as follows.

Problem 2: Find a sequence of symbolic plans that will guide the robot (1) from L_i to L_j, given $i, j \in \{1, 2, \ldots, n_L\}$.

Because of the presence of noise in (1) and uncertainty in the location of various terrain features, symbolic plans will have imprecise effects, even if the robot's initial conditions are known. The situation is akin to a human driver misinterpreting a set of directions because, for example, a street sign is not clearly visible or traffic is being diverted because of construction. This suggests that when executing a symbolic plan, one must think in terms of probabilities of reaching L_j. Therefore, Problem 2 implicitly requires the robot to know the probability of being at each landmark, based on its sensor measurements. Such knowledge is necessary for deciding which control law to apply and whether the desired landmark has been reached. This leads naturally to the interpretation of Problem 2 as a Markov decision process, where states are identified with landmarks and transitions are triggered by the execution of a plan. We will take advantage of this interpretation in Section III; see also [26], [38]–[40] for related work.

A symbolic controller that attempts to solve Problem 2 might take the form of an iterative process that collects observations, decides on an input string and applies that string to the robot. Before we can specify such a feedback process for global navigation, we need to sharpen the problem statement by making precise the effects of executing symbolic control laws in a stochastic setting, as well as the kinds of observations that are available for decision-making. In keeping with our symbolic approach, the latter will be restricted to measurements (or estimates) of which landmark the robot is currently on.

[1]While the problem of choosing landmarks is an important one, it is outside the scope of this article. For a discussion of some of the relevant issues see [37].

C. Tokenizing Observations

We do not assume that the robot knows which landmark it is on at the completion of a plan. Instead, a series of sensor measurements may be made at that time, yielding information about the local environment. We define an *observation plan* to be a symbolic plan which moves the robot locally (on a given landmark), while gathering sensor data. Likewise, we will use the term *control plan* to distinguish strings which are designed to steer the robot between landmarks. It is important to note that observation plans should be designed so as to ensure that at the completion of the plan, the robot remains on the same landmark as at the start of the plan. This can be accomplished by taking advantage of existing planning techniques, such as those mentioned in Section I. A typical observation plan might consist of a short "tour" around a landmark, using odometry or SLAM techniques to return to the initial position.

Depending on the available sensors, the time spent on identification, and the level of uncertainty in measurements, different landmarks may appear similar to varying degrees and thus may not be uniquely identifiable. We define the set $\mathcal{Z} \triangleq \{z_1, \ldots, z_q\}$, $q \leq n_L$, to be the collection of possible *observation outcomes*, or simply *observations*. This set can be viewed as a set of equivalence classes of landmarks, where two landmarks are equivalent if they cannot be distinguished using only measurements taken while on either of the two. To generate an observation the robot first obtains a sequence of measurements, possibly while in motion, and from disparate sensors. In general, taking advantage of different sensor modalities, such as vision, sonar, and laser range finders, and of multiple "viewpoints" helps to reduce the overall uncertainty in determining which landmark the robot is on. The problem of fusing these data to generate a single observation is a common one in robotics and there are a variety of ways to handle it [41], [42]; it can be described abstractly as follows. Let t_0^k denote the time at which the k^{th} observation plan begins and let t_1^k denote the time at which it ends. Let $S_{[t_0^k, t_1^k]}$ denote the set of all possible sensor data streams collected over the time interval $[t_0^k, t_1^k]$. A measurement of the current landmark is a filter, or mapping, $\mathcal{F} : S_{[t_0^k, t_1^k]} \rightarrow \mathcal{Z}$. To be more precise about the form such a filter might take, one must first specify the representation of the landmarks, that is, whether they are occupancy grid maps, visual cues, GPS coordinates, etc. In Section IV we detail one possible way of generating observations from $\mathcal{Z}$ by sampling from a probability mass function which is produced by a bank of particle filters.

III. Language-based Optimal Control via DP

A. Global Navigation as a Markov Decision Process

We assume we have available a collection of symbolic plans denoted by $\mathcal{G} = \{\alpha_1, \ldots, \alpha_{n_c}\}$. Let A denote a mapping from $\mathcal{G}$ into the set of Markov matrices, that associates to each control plan $v \in \mathcal{G}$ a Markov matrix $A(v)$ which specifies the transition probabilities between landmarks; thus $p_{ij}(v) = [A(v)]_{ij}$ is the probability of ending at landmark L_j given that the robot (1) begins at landmark L_i and executes plan v. Likewise, let $\mathcal{H} \triangleq \{\beta_1, \beta_2, \ldots, \beta_{n_o}\}$ be a collection of available observation plans and let B be a mapping from $\mathcal{H}$ into the set of Markov matrices. Thus $[B(o)]_{ij}$ is the probability of observing z_j given that the robot is on landmark L_i and that the observation plan o is executed. We will assume that the Markov matrices for all available control and observation plans are given in advance, perhaps having been computed from prior experience or from simulations.

B. Optimal Symbolic Control Sequences via Dynamic Programming

Having agreed on symbolic descriptions for the system's inputs $(\mathcal{G}, \mathcal{H})$, outputs $(\mathcal{Z})$ and states $(\mathcal{L})$, we define the sets of control actions

$$\mathcal{V} \triangleq \{A(\alpha_1), \ldots, A(\alpha_{n_c})\}$$

and observation actions

$$\mathcal{O} \triangleq \{B(\beta_1), \ldots, B(\beta_{n_o})\}.$$

In this framework time is naturally a discrete variable which is updated upon completion of a pair of control and observation plans. Let the quadruple (χ_k, v_k, o_k, z_k) denote the landmark $\chi_k \in \mathcal{L}$, control plan $v_k \in \mathcal{G}$, observation plan $o_k \in \mathcal{H}$, and observation outcome $z_k \in \mathcal{Z}$ at time k. We will use a probability mass function to describe the robot's location on the set of landmarks $\mathcal{L}$. Towards that end, let I_k denote the usual information vector

$$I_k \triangleq (v_0, o_0, z_0, \ldots, v_k, o_k, z_k) \tag{2}$$

and define the row vector of conditional probabilities

$$P_{k|k} \triangleq \begin{pmatrix} p^1_{k|k} & \ldots & p^{n_L}_{k|k} \end{pmatrix} \tag{3}$$

where $p^i_{k|k}$ is the probability of being on landmark L_i given I_k. We note that $P_{k|k}$ is a sufficient statistic for the description of the state of the system (see, e.g., Chapter 5 of [43]).

The dynamics of the conditional probability vector may be described as follows. Let e denote the vector $(1\,1\,\ldots\,1)^T$. Using Bayes rule and making the usual Markov assumption that the current observation depends only on the current position and the current observation plan, the dynamics of the conditional probability can be shown to be given by

$$P_{k+1|k+1} = \frac{P_{k|k}A(v_k)P_{z_k}(o_k)}{P_{k|k}A(v_k)P_{z_k}(o_k)e} \tag{4}$$

where $A(v_k)$ is the Markov matrix capturing the effect of applying the control plan v_k, while the diagonal matrix

$$P_{z_k}(o_k) = \mathrm{diag}\left[Pr(z_k|\chi_k = 1, o_k), \ldots, Pr(z_k|\chi_k = n_L, o_k)\right] \tag{5}$$

captures the effect of the observation generated from the sensor data measured while applying the motion plan o_k. We note that the Markov matrix for the observation plan o_k appears implicitly in (5) since the diagonal of $P_{z_k}(o_k)$ is a column of $B(o_k)$.

Finally, define the cost functions

$$g : \mathcal{P} \times \{0, \ldots, N-1\} \to \mathbb{R}^*_+ \tag{6}$$
$$g_u : \mathcal{P} \times \mathcal{V} \times \{0, \ldots, N-1\} \to \mathbb{R}^*_+ \tag{7}$$
$$g_o : \mathcal{P} \times \mathcal{O} \times \{0, \ldots, N-1\} \to \mathbb{R}^*_+ \tag{8}$$

where $\mathcal{P}$ is the space of probability mass functions over $\mathcal{L}$. The functions $g_u(\cdot,\cdot,\cdot)$ and $g_o(\cdot,\cdot,\cdot)$ capture the costs (e.g., in terms of time or energy) of the control and observation plans. For example, a robot could make a few quick measurements to get a rough estimate of its surroundings, or it could spend more time and energy to investigate the local details. Similarly, one control plan may move the robot quickly at the expense of accurate sensing while another may move the robot slowly to minimize errors.

We can now rephrase Problem 2 as a standard Markov decision process problem.

Problem 3: Given the control and observation plans associated with $\mathcal{V}$, $\mathcal{O}$, find the control policy $\pi = (v_0, o_0, \ldots, v_{N-1}, o_{N-1})$ that maximizes

$$J_\pi(P_{0|0}) = \mathcal{E}_{z_k, k=0,\ldots,N-1}\left\{g(P_{N|N}, N) + \sum_{k=0}^{N-1}\left(g_u(P_{k|k}, v_k, k) + g_o(P_{k|k}, o_k, k)\right)\right\} \tag{9}$$

subject to the dynamics (4), (5), where $\mathcal{E}(\cdot)$ denotes expected value.

Problem 3 can be solved via dynamic programming (DP) [43]. The resulting feedback control policy, selected via DP, is a sequence of motion and observation plans which seek to maximize J.

We now consider in detail two particular instances of Problem 3, namely that of maximizing the probability of arrival at a desired landmark and that of minimizing the expected total time to arrive at the destination. In each case we use DP to determine the sequence of symbols from the alphabet of control and observation plans which optimizes the appropriate cost function.

C. Maximizing the Probability of Arrival

To maximize the probability of arrival at a desired landmark in N steps, we define the cost function to be

$$J_\pi(P_{0|0}) = \max_{\substack{v_k \in \mathcal{V}, o_k \in \mathcal{O} \\ k=0,\ldots,N-1}} \mathcal{E}_{z_k, \quad k=0,\ldots,N-1}\{P_{N|N}d\} \tag{10}$$

where d is a unit column vector with a 1 in the index corresponding to the desired landmark and 0's everywhere else. Application of the DP algorithm yields the optimal return function at step k:

$$\begin{aligned} J_k(P_{k|k}) &= \max_{v,o} \sum_{j_1=1}^{m} \sum_{j_2=1}^{m} \cdots \\ &\quad \sum_{j_{N-k}=1}^{m} \left\{ P_{k|k} A(v) \cdot P_{z_k=j_{N-k}}(o) A\left(v_{k+1}^*\right) \right. \\ &\qquad \times P_{z_{k+1}=j_{N-k-1}}\left(o_{k+1}^*\right) \cdots \cdot A\left(v_{N-1}^*\right) \\ &\qquad \left. \times P_{z_{N-1}=j_1}\left(o_{N-1}^*\right) \right\} d \end{aligned} \tag{11}$$

for $k = 0, 1, \cdots, N-1$, where v_k^* and o_k^* are the optimal choice of control and observation plans at the k^{th} step. This in turn implies that the optimal control at the k^{th} step is the pair of control plan v_k^* and observation plan o_k^* given by

$$\begin{aligned} (v_k^*, o_k^*) &= \arg\max_{\substack{v \in \mathcal{V} \\ o \in \mathcal{O}}} \sum_{j_1=1}^{m} \sum_{j_2=1}^{m} \cdots \\ &\quad \sum_{j_{N-k}=1}^{m} \left\{ P_{k|k} A(v) P_{z_k=j_{N-k}}(o) \cdot A\left(v_{k+1}^*\right) \right. \\ &\qquad \times P_{z_{k+1}=j_{N-k-1}}\left(o_{k+1}^*\right) \cdots A\left(v_{N-1}^*\right) \\ &\qquad \left. \times P_{z_{N-1}=j_1}\left(o_{N-1}^*\right) \right\} d. \end{aligned} \tag{12}$$

Solving this maximization problem in practice is of course a nontrivial task. If the cardinalities of the control and observation alphabets and the length of the horizon N are all small enough then the optimal solution can be found by enumerating all possibilities. However this is generally not a feasible approach and simplifications will need to be made to find an adequate solution in a reasonable amount of time. One technique for reducing the computational complexity is to use a limited lookahead policy by truncating the number of stages N in the planning horizon. In addition, a higher-level decision maker may use prior experience to restrict the choice of plans to be searched. For example, if it is known that the robot is navigating an indoor office space, the search may be restricted to those plans known to be effective in such an environment.

D. Minimizing the Time to Reach a Landmark

To maximize the probability of arrival at a desired landmark while minimizing the time it takes to get there we associate to each control and observation plan the conditional expectation of the time to complete the plan given the current landmark. Let

$$T(v) \triangleq \left(\mathcal{E}\{\tau | v, \chi = 1\}, \ldots, \mathcal{E}\{\tau | v, \chi = n\}\right)^T, \ v \in \mathcal{G} \tag{13}$$

where τ denotes the time to complete the plan. Let $T(o)$ be defined similarly. We take as our cost function

$$\begin{aligned} J_\pi^*(P_{0|0}) = &\max_{\substack{v_k, o_k \\ k=0,\ldots,N-1}} \mathcal{E}_{z_k, k=0,\ldots,N-1} \\ &\times \left\{ a_1 P_{N|N} d - \sum_{k=0}^{N-1} P_{k|k} \left[a_2 T(v_k) + a_3 T(o_k)\right] \right\}. \end{aligned} \tag{14}$$

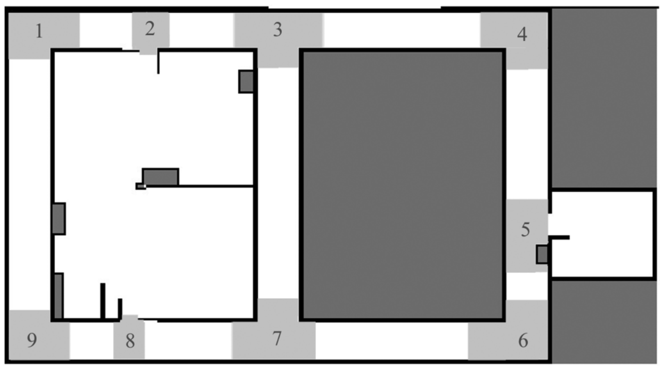


Fig. 1. An office-like environment, approximately 16 m × 30 m in size. Gray levels indicate occupancy and numbered areas indicate landmarks.

Here the a_i are weights used to set the relative importance of reaching the desired landmark versus minimizing the time to get there. To find the optimal sequence of control and observation plans we once again use DP. The optimal return function is

$$\begin{aligned} J_{k|k}(P_{k|k}) &= \max_{v,o} \left[\sum_{j_1=1}^{m} \cdots \sum_{j_{N-k}=1}^{m} \left\{ a_1 P_{k|k} A(v) \cdot P_{z_k=j_{N-k}}(o) \right. \right. \\ &\qquad \times A\left(v_{k+1}^*\right) \times P_{z_{k+1}=j_{N-k-1}} \\ &\qquad \times \left(o_{k+1}^*\right) \cdots \cdot A\left(v_{N-1}^*\right) \\ &\qquad \left. \times P_{z_{N-1}=j_1}\left(o_{N-1}^*\right) \right\} d^T \\ &\quad - \sum_{j_1=1}^{m} \cdots \sum_{j_{N-k-1}}^{m} \left\{ P_{k|k} A(v) P_{z_k=j_{N-k-1}}(o) \right. \\ &\qquad \times A\left(v_{k+1}^*\right) \cdot P_{z_{k+1}=j_{N-k-2}} \\ &\qquad \times \left(o_{k+1}^*\right) \cdots A\left(v_{N-2}^*\right) \\ &\qquad \left. \times P_{z_{N-2}=j_1}\left(o_{N-2}^*\right) \right\} \\ &\quad \cdot \left[a_2 T\left(v_{N-1}^*\right) + a_3 T\left(o_{N-1}^*\right)\right] - \cdots \\ &\quad - \sum_{j_1=1}^{m} P_{k|k} A(v) P_{z_k=j_1}(o) \\ &\quad \times \left[a_2 T\left(v_{k+1}^*\right) + a_3 T\left(o_{k+1}^*\right)\right] \\ &\quad \left. - P_{k|k}\left[a_2 T(v) + a_3 T(o)\right] \right] \end{aligned} \tag{15}$$

with the optimal (v_k^*, o_k^*) given in the usual manner.

IV. Simulations

To illustrate our approach to global navigation and localization, we performed a set of numerical experiments[2] involving a planar, direct-drive nonholonomic robot equipped with range sensors operating in the office-like environment shown in Fig. 1. In the image, white denotes open space, black denotes occupied space, and dark gray denotes no knowledge. Nine landmarks were defined in the environment. They are indicated in Fig. 1 as shaded rectangles, one at each of four hallway corners, two T-intersections, and three doorways. These landmarks were classified into four groups—*corner*, *T-junction*, *left doorway*, and *right doorway*—giving four possible observation outcomes. Each landmark was represented by an occupancy grid map with

[2]The source code for the experiments is available upon request.

cells of 10 cm square. The landmarks were numbered from 1 to 9 beginning with the upper left corner and proceeding clockwise around the map.

A. Robot Model

The nonholonomic robot was modeled as a kinematic system. The equations of motion were

$$\begin{aligned}\dot{x} &= u_f \cos(\theta)\\ \dot{y} &= u_f \sin(\theta)\\ \dot{\theta} &= u_t\end{aligned} \tag{16}$$

where (x, y, θ) denote the position and orientation of the robot. The variables u_f and u_t denote the forward and turning velocities, respectively. The actuator commands were the commanded left and right wheel velocities, u_L and u_R. These were related to the commanded forward and turning velocities by

$$\begin{aligned}u_L &= \frac{2u_f + Du_t}{2}\\ u_R &= \frac{2u_f - Du_t}{2}\end{aligned} \tag{17}$$

where D was the distance between the wheels. The actuator commands u_L and u_R were corrupted by samples from independent normal random variables to simulate actuator noise. The position of the robot at the next instance of time was calculated by using an exact solution of (16) under the assumption that the (noisy) control values were constant over the time step.

The robot was equipped with 16 range sensors, spaced equally in a circle around its perimeter. Each sensor acted as a single-ray laser range-finder, returning the distance to the nearest obstacle along the sensor ray out to a maximum of 4 m. This distance reading was corrupted by a sample from a normal random variable to simulate measurement noise.

B. Control Alphabet

As described in Section II, we chose the motion description language MDLe [7]–[9] as a means of tokenizing the robot's control and storing the instructions used to guide it from one landmark to another. In order to generate the initial "alphabet" from which MDLe strings could be constructed, we defined four control and five interrupt functions. They are listed next.

- Control functions
  - — `go` u_f u_t
    Sets commanded forward velocity to u_f and turning velocity to u_t.
  - — `goStraightAvoid` u_f d u_t d_o
    Moves the robot forward while steering away from nearby obstacles by applying

$$u_f = \begin{cases} u_f, & r > d\\ u_f(d - d_o - r), & s < r \le d\\ 0, & r < d_o\end{cases}$$
$$u_t = \begin{cases} 0, & r > d\\ \pm u_t, & r \le d\end{cases}$$

    where r is the distance to the nearest obstacle as estimated from the sensor data. The sign of u_t is chosen so as to steer away from the obstacle.
  - — `alignNearWall` k_t
    Aligns the robot to the nearest wall by setting

$$u_f = 0 \quad u_t = -k_t\theta$$

    where θ is an estimate of the angle between the heading of the robot and the nearest wall.
  - — `followNearWall` u_f k_f k_t d
    Follows the nearest wall while maintaining a distance d to that wall by setting

$$u_f = \begin{cases} -k_f\,(2(r-d)+\theta)\sin(\theta), & \mathrm{abs}(\theta) \ge \epsilon\\ u_f, & \mathrm{abs}(\theta) < \epsilon\end{cases}$$
$$u_t = \begin{cases} -k_t\,((r-d)+2\theta)\,, & \mathrm{abs}(\theta) \ge \epsilon\\ 0, & \mathrm{abs}(\theta) < \epsilon\end{cases}$$

    where r is the distance to the nearest obstacle and ϵ is a small, positive constant.
- Interrupt functions
  - — `wait` T
    Interrupts after T seconds have elapsed.
  - — `atHallPattern` f l b r d
    Interrupts if the obstacle pattern sensed around the robot matches the pattern encoded in the arguments. Here the arguments f, l, b, and r indicate the front, left, back, and right of the robot; their values indicate whether the given direction should be clear out to distance d (0), occupied before d (1), or should be ignored (2).
  - — `inHall` w
    Interrupts if the sensor data indicate the robot is in a hallway of width less than or equal to w.
  - — `alignedNearWall` θ
    Interrupts if the heading angle with respect to the nearest wall as estimated from the sensor data is within $\pm\theta$.
  - — `frontNotOpen` d
    Interrupts if an obstacle within distance d of the front of the robot is detected.
  - — `frontOpen` d
    Interrupts if there are no obstacles detected within distance d of the front of the robot.
  - — `sideOpen` s d
    Interrupts if the sensors indicate that the side encoded in the argument s has no obstacles within a distance d. The variable s can take the values 1 (left side), 2 (right side), and 3 (either side).

The control alphabet available to the robot was the collection of all possible atoms created by combining the above interrupt and control functions. We note that some of the interrupt functions shown above are redundant. For example, the functionality of the last three could be implemented using `atHallPattern`. We chose to include the additional interrupt functions here for the sake of improving the readability of the resulting MDLe code.

TABLE I
MDLe CODE FOR PLAN α_1

```
Plan 1 (wait 500)
  ((hallPattern 1 2 0 0 0.25),  (go 0 π/4))
  (frontNotOpen 0.5),           (goStraightAvoid 0.5 0.3 1 0.025))
  ((wait 1),                    (go 0 −π/2))
  ((frontOpen 0.9),             (go 0 −π/4))
  ((alignedNearWall 5),         (alignNearWall 2) )
  ((inHall 2),                  (followNearWall 0.5 20 2 0.5))
  ((sideOpen 3 3),              (followNearWall 0.5 20 2 0.5))
```

C. Control Plans

Armed with the above alphabet, we defined three control plans α_1, α_2, and α_3, designed to move the robot through the hallways of the map shown in Fig. 1. The first plan was intended to navigate from one landmark to another in a counter-clockwise manner, the second was designed to move the robot in a clockwise manner, and the third was the identity plan which applied a zero control, leaving the robot in place. The first plan is shown in Table I. The source code is written as a sequence of atoms, each of which is comprised of an interrupt function followed by a control function. The plan has a global interrupt which causes it to expire after a maximum of 500 seconds. It begins by pointing the robot toward a wall, moving close to that wall, turning to the left, and then aligning with the wall. Once aligned it follows the wall until it determines it is in a hallway (which indicates it has moved out of the current landmark) and then continues until the robot detects an opening to either side. The second plan is similarly structured.

To determine the corresponding Markov matrices, each plan was run at least 50 times from each landmark starting from randomly selected initial conditions. The robot's true position was recorded at the end of each run. The resulting matrices were as shown in the equation at the bottom of the page, and $A(\alpha_3) = \mathbb{I}$ where $\mathbb{I}$ is the identity matrix. From these matrices we see that the first plan did tend to steer the robot to the previous landmark, with the exception of landmark 5 (from which the robot was most likely to move "forward" to landmark 6) and landmark 6 (from which the plan usually drove the robot all the way to landmark 1). The second plan generally led the robot to the subsequent landmark, except from landmark 5 from which the robot usually went to landmark 4.

The times it took for the robot to complete each control plan were also recorded. The average times (in seconds) from each landmark were

$$
\begin{aligned}
T_1 &= [26.2\ \ 14.3\ \ 16.9\ \ 22.7\ \ 14.7\ \ 28.5\ \ 23.5\ \ 14.1\ \ 14.2]\\
T_2 &= [16.6\ \ 18.8\ \ 29.6\ \ 21.5\ \ 27.9\ \ 23.6\ \ 25.5\ \ 18.6\ \ 28.1]\\
T_3 &= [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0].
\end{aligned}
$$

D. Observation Plans

In addition to the three control plans, two observation plans β_1 and β_2 were created. The first moved the robot forward very briefly (0.2 sec), while avoiding any intervening obstacles. During that time, the robot could only gather four sets of sensor readings, for a quick but less-informative observation. The second observation plan moved the robot forward for a full second, gathering 20 sets of readings for a more accurate observation of the current landmark. Note that under both plans the robot moved only a short distance and consequently it did not leave the current landmark. These plans were chosen primarily for their simplicity; more elaborate plans could easily be adopted.

As discussed in Section II, the sensor data gathered during execution of the observation plan must be filtered to generate an observation of the current landmark (from the set $\mathcal{Z}$). Consider first the standard problem of localization on a given map. This problem can be solved effectively using a particle filter [44], [45]. This is a Monte Carlo-based approximation to the Bayes

$$
A(\alpha_1) = \begin{bmatrix}
0.36 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0.63\\
0.56 & 0.39 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05\\
0 & 0.83 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0.02\\
0 & 0 & 0.52 & 0.44 & 0.04 & 0 & 0 & 0 & 0\\
0.04 & 0 & 0 & 0 & 0.36 & 0.60 & 0 & 0 & 0\\
0.56 & 0 & 0 & 0 & 0 & 0.44 & 0 & 0 & 0\\
0.05 & 0 & 0 & 0 & 0 & 0.74 & 0.21 & 0 & 0\\
0.03 & 0 & 0.01 & 0 & 0 & 0 & 0.48 & 0.48 & 0\\
0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0.68 & 0.31
\end{bmatrix}
$$

$$
A(\alpha_2) = \begin{bmatrix}
0.14 & 0.74 & 0 & 0 & 0 & 0 & 0 & 0 & 0.12\\
0 & 0.14 & 0.74 & 0.12 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0.28 & 0.64 & 0.08 & 0 & 0 & 0 & 0\\
0 & 0 & 0.04 & 0.16 & 0.80 & 0 & 0 & 0 & 0\\
0.02 & 0 & 0 & 0.86 & 0.12 & 0 & 0 & 0 & 0\\
0.06 & 0 & 0 & 0 & 0 & 0.26 & 0.68 & 0 & 0\\
0.10 & 0 & 0 & 0 & 0 & 0 & 0.10 & 0.64 & 0.16\\
0.16 & 0 & 0 & 0 & 0 & 0 & 0 & 0.22 & 0.62\\
0.80 & 0 & 0 & 0 & 0 & 0 & 0 & 0.02 & 0.18
\end{bmatrix}
$$

filter which uses the sensor data to maintain a probability density function (pdf) over all possible headings and locations on the given landmark map. A finite set of samples (often called particles) is drawn from the density function describing the robot's state. The state of each particle is propagated using a motion model for the evolution of the system (1). Incoming sensor data are then used to weight the samples according to Bayes rule and a sensor model; the latter is simply a conditional probability density function over the set of sensor outputs given the state of the system.

In our setting there is not one but rather $q = \text{card}(\mathcal{Z})$ maps, one for each possible observation outcome. Independent instances of the particle filter algorithm can be run simultaneously on each of these maps as the robot executes an observation plan. To localize within the set $\mathcal{Z}$, we define a probability mass function over $\mathcal{Z}$ as follows. Let C^i, $i = 1, \dots, q$ denote the sum of the probabilities of the particles on each of the possible observation outcomes after completion of the observation plan but before normalization. By construction of the set $\mathcal{Z}$, each of these landmarks is distinguishable from the others using the sensor suite of the robot. Therefore, if the robot is not on a landmark corresponding to z_i, the surrounding environment will appear quite different from the environment encoded in the landmark map. As a consequence, the (non-normalized) weight of each particle on that map will be very small; C^i is thus an indication of the goodness of the "fit" of the landmark map to the actual environment surrounding the robot. With this in mind, Prob(observation z_i) may be approximated by the probability mass function defined by $(C^i / \sum_{j=1}^{q} C^j)$, $i = 1, \dots, q$. An observation of the current landmark is then generated by sampling from this mass function.

For the simulation experiments presented here, particle filter localization algorithms (with 300 particles) were run concurrently on each of the maps corresponding to the possible observation outcomes (corner, T, left doorway, and right doorway). Each observation plan was run at least 25 times from each landmark to obtain an estimate of the distribution for that observation. The resulting matrices for the two observation plans were

$$B(\beta_1) = \begin{bmatrix} 0.78 & 0.05 & 0.15 & 0.02 \\ 0.19 & 0.62 & 0.12 & 0.07 \\ 0.40 & 0.07 & 0.43 & 0.10 \\ 0.82 & 0.04 & 0.07 & 0.07 \\ 0.02 & 0.14 & 0.05 & 0.79 \\ 0.99 & 0.01 & 0.00 & 0.00 \\ 0.36 & 0.10 & 0.37 & 0.17 \\ 0.16 & 0.57 & 0.14 & 0.13 \\ 0.82 & 0.04 & 0.10 & 0.04 \end{bmatrix}$$

$$B(\beta_2) = \begin{bmatrix} 0.97 & 0.01 & 0.01 & 0.01 \\ 0.03 & 0.65 & 0.20 & 0.12 \\ 0.13 & 0.21 & 0.63 & 0.03 \\ 0.81 & 0.09 & 0.04 & 0.06 \\ 0.00 & 0.29 & 0.03 & 0.68 \\ 0.90 & 0.07 & 0.03 & 0.00 \\ 0.10 & 0.25 & 0.59 & 0.06 \\ 0.02 & 0.64 & 0.12 & 0.22 \\ 0.45 & 0.27 & 0.21 & 0.07 \end{bmatrix}. \qquad (18)$$

The fact that observation plans are executed only after the robot completes a control plan implies that the robot's position at the start of an observation plan will not be uniformly random. This suggests that the efficiency of our localization algorithm could be increased by restricting the choice of the particle filters' initial pdfs. Specifically, one should "seed" with particles only those areas which the robot is likely to be in at the end of a control plan. In our experiments we did precisely that, by first collecting (off-line) statistics on the robot's position after applying each of the available control plans from each landmark. Those statistics were stored and later used to generate the initial pdf for the particles in our localization algorithm.

Remarks: The efficacy of the particle filter-based approach is strongly influenced by the number of particles chosen for each instance of the algorithm. Since we are interested in localizing over the set $\mathcal{L}$, as opposed to determining a precise location on a given map M_i, the number of particles required per landmark is generally less than what would be needed for accurate localization on that M_i. However, using an insufficient number of particles will generally lead to a uniform distribution over the set of possible landmark observations. There is therefore a tradeoff between the number of particles used (and thus the required computation time) and the amount of information generated. This tradeoff can be captured under the general framework of Section III-A by assigning different costs to different choices for the number of particles used, and optimizing these costs.

It is important to keep in mind that the proposed localization algorithm is to be run only while an observation plan is being executed, and not while a control plan is in progress. We have found that on small maps (e.g., 5 m^2 with 300 particles), the algorithm tends to converge quickly to a sharp distribution, even if the robot's surroundings look very different than the map on which the particles evolve. If the algorithm were run during execution of the control plan then at the start of the subsequent observation plan the particle distribution would be highly localized but most likely erroneous.[3] As a result, the weights C^i would carry very little information as to which landmark the robot is currently on. While there are variants of the basic particle filter-based localization algorithm which can handle this difficulty [44], the additional complications can be avoided simply by running the particle filters only when observing a landmark.

E. Navigation Results

In the first of the two numerical experiments, the robot was required to reach a desired landmark while minimizing the amount of time it took to get there. This was accomplished by utilizing the optimal feedback controller of (15). We chose $a_1 = 1000$ and $a_2 = a_3 = 1$ to ensure that the controller only optimized for time over those control and observation sequences that had a high probability of arrival at the desired landmark. The number of stages for the optimal controller was set to $N = 4$. If at any time the probability of being on the desired landmark exceeded a threshold value (0.9) then the controller terminated. If at the

[3]The situation in which the underlying pdf is a sharply-peaked unimodal distribution with the peak far away from the true robot location is known in the robotics literature as the "kidnapped robot" problem.

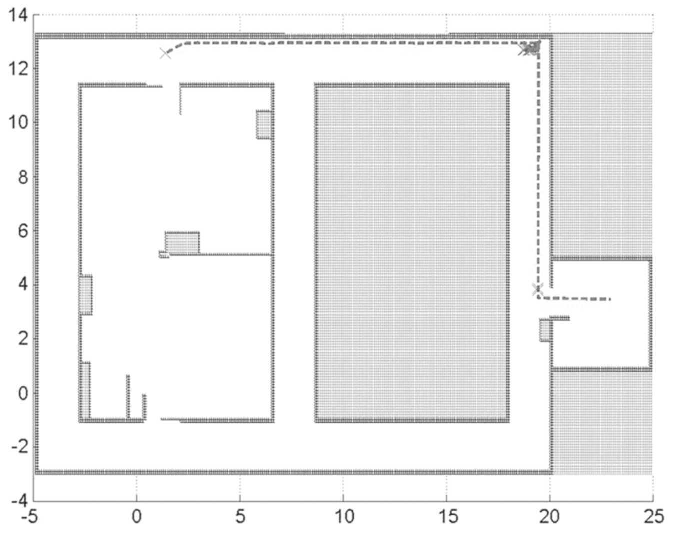


Fig. 2. Simulation 1: Robot trajectory from landmark 2 to landmark 5. Six plans were executed before the controller knew with high probability (>90%) that landmark 5 had been reached. A landmark-specific controller then guided the robot into the room. Figure axes indicate distance in meters.

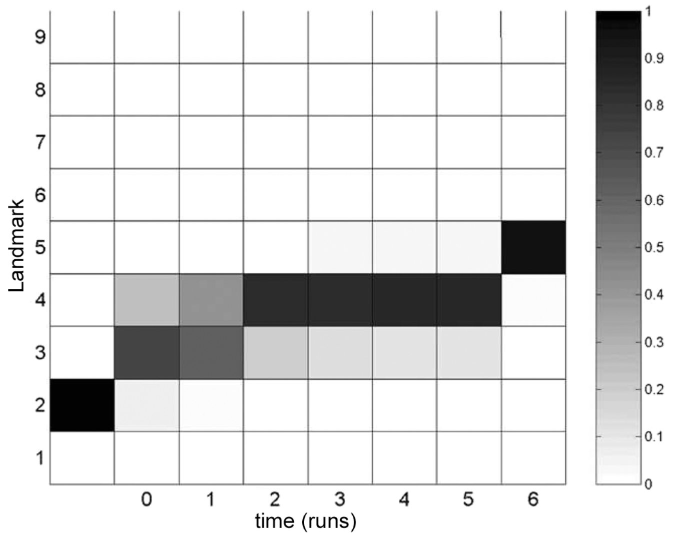


Fig. 3. Simulation 1: Evolution of the conditional probability. The controller was initially given the true location of the robot, and maintained fairly good localization throughout.

end of four steps the probability of being on the desired landmark was less than the threshold, then the controller was run again.

The robot was placed in a random starting position on landmark 2 (outside the top door of the large room) and knew with certainty which landmark it was on. The robot was commanded to go to landmark 5 (outside the open office door on the right). Once there, a "local" controller (tailored to local maps that include doorways) was used to guide the robot into the room. The resulting trajectory of the robot is shown in Fig. 2, with the position of the robot at the end of each control and observation plan indicated by an 'x'; the evolution of the conditional probability vector is shown in Fig. 3.

The optimal selection of control and observation plans together with the true landmark (after the control was applied),

TABLE II
SIMULATION 1: SEQUENCE OF PLANS, POSITIONS, AND OBSERVATIONS. THE NOTATION "IC" STANDS FOR "INITIAL CONDITION". THE MONTE CARLO-BASED OBSERVER YIELDED THE CORRECT LANDMARK CLASS IN EVERY MEASUREMENT

Step	Control plan	Obs. plan	True LM	True LM Class	Obs. LM Class
IC	-	-	2	2	-
0	2	1	4	1	1
1	2	1	4	1	1
2	2	1	4	1	1
3	2	1	4	1	1
4	2	1	4	1	1
5	2	1	4	1	1
6	2	1	5	4	4

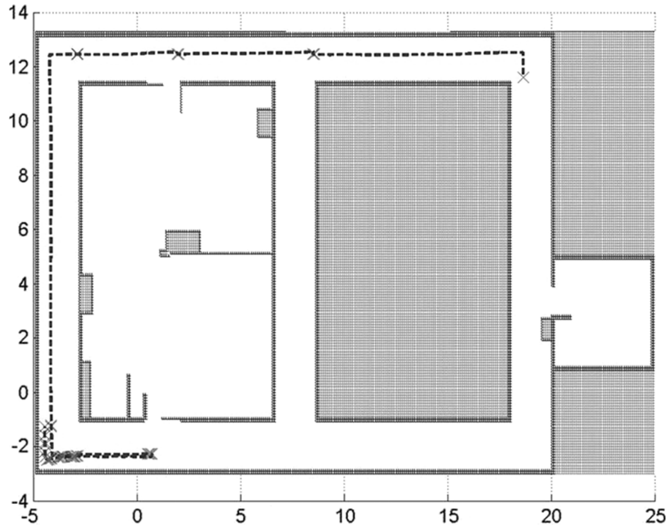


Fig. 4. Simulation 2: Robot trajectory from landmark 4 to landmark 9. The robot moved through the landmarks in a counter-clockwise fashion to reach landmark 9 in four steps. By continuing to landmark 8 the robot was able to determine its location. It then returned to landmark 9 but needed several observations to confirm its position.

as well as the true and observed landmark classes are shown in Table II. The robot navigated successfully to the desired landmark after six steps. Notice that the controller always selected the brief observation plan; this is because the improvement in accuracy when using the longer observation plan did not appear to offset the cost of the longer time needed to run it. The final transition into the room was made after the controller was almost certain that it had arrived at the desired landmark. We note that the particle filter-based observer returned the correct landmark class in every observation during this run.

In the second simulation the robot was placed on landmark 4 (the upper-right corner of the map) and asked to go to landmark 9 (the lower-left corner of the map). Unlike the previous case, the conditional probability was initialized to a uniform density over all landmarks, so that the robot had no a-priori knowledge of its location. The goal was to maximize the probability of arrival at landmark 9 and thus the controller defined by (12) was used. The resulting trajectory is shown in Fig. 4 and the evolution of the condition probability is shown in Fig. 5.

The optimal sequence of control and observation plans and the actual and observed landmark classes are shown in Table III (in the table the omitted data simply repeats the previous line).

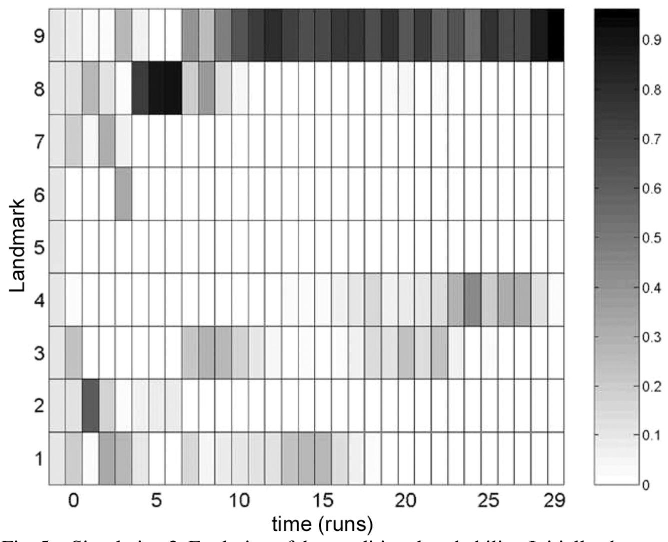


Fig. 5. Simulation 2: Evolution of the conditional probability. Initially, the controller had no information as to the robot's position, but determined the correct location within five steps. Several erroneous measurements after step 20 forced the controller to make additional observations before concluding that the desired landmark had been reached.

TABLE III
SIMULATION 2: SEQUENCE OF PLANS, POSITIONS, AND OBSERVATIONS. NOTICE THAT THE MONTE CARLO-BASED LOCALIZATION ALGORITHM RETURNED SEVERAL ERRONEOUS OBSERVATIONS. DESPITE THIS THE CONTROLLER WAS ABLE TO STEER THE ROBOT TO THE DESIRED LANDMARK

Step	Control plan	Obs. plan	True LM	True LM Class	Obs. LM Class
IC	-	-	4	1	-
0	1	1	3	3	3
1	1	1	2	2	2
2	1	1	1	1	3
3	1	1	9	1	1
4	1	1	8	2	2
5	3	1	8	2	2
6	3	1	8	2	2
7	2	1	9	1	3
8	3	1	9	1	4
9	3	1	9	1	1
⋮	⋮	⋮	⋮	⋮	⋮
13	3	2	9	1	1
14	3	1	9	1	2
15	3	1	9	1	1
16	3	1	9	1	4
⋮	⋮	⋮	⋮	⋮	⋮
19	3	2	9	1	2
20	3	1	9	1	4
21	3	1	9	1	1
22	3	1	9	1	4
23	3	2	9	1	1
24	3	2	9	1	1
25	3	2	9	1	2
26	3	2	9	1	1
27	3	1	9	1	1
28	3	2	9	1	2
29	3	2	9	1	3

The robot moved to landmark 9 quickly but at that point was unsure of its position. After moving away to landmark 8 the robot had localized itself. However, after returning to the landmark 9 the robot received several erroneous measurements, and consequently required additional observations before it could conclude that it had reached the desired landmark. Note that in this case, the controller did on occasion choose the longer of the observation plans.

Remark: The office-like environment used in these simulations was highly structured and that fact was used to design effective control laws and interrupt functions for navigation. However the general approach presented in this paper would work just as well in unstructured environments.

V. CONCLUSIONS

We have described a symbolic feedback process for navigation in a large, imprecisely described environment. In this work, the environment is described as a collection of widely separated regions (termed "landmarks") which are the only areas for which detailed descriptions (e.g., maps) are available. Landmarks are linked by symbolic strings, which a robot could interpret to steer itself from one landmark to another. We have focused on landmark-to-landmark navigation and have provided references to existing techniques which could be used to navigate locally (i.e., to specific coordinates) within a landmark. In the present setting, symbolic feedback control is enabled by the use of a motion description language which allows one to tokenize the vehicle's inputs as well as the relationships between landmarks, thus passing to a symbolic version of the navigation problem. In that new, more abstract domain, we can find solutions using techniques from Markov decision processes. In particular, feedback control policies are generated via dynamic programming and consist of: i) strings intended to move the robot towards a desired landmark, alternating with ii) observation policies that process the robot's sensor data (via a set of particle filters) to yield an approximation of the conditional probability of being on each landmark. That probability in turn is used to generate symbolic observations that encode the robot's best guess as to its location.

The approach we have presented is robust to sensor and actuator noise. However, it requires that upon their completion, the control and observation plans of choice lead the robot to one of the known landmark maps. If that assumption is not met then modifications are required to prevent the robot from being "lost". The proposed approach fits naturally with the idea of using symbolic instructions to specify motion control tasks and presents the first instance, to the authors' knowledge, of a feedback control law that is implemented at the level of a motion description language, as opposed to that of sensors and actuators.

REFERENCES

[1] P. Rouchon, M. Fliess, M. Levine, and P. Martin, "Flatness, motion planning and trailer systems," in *Proc. 32nd IEEE Conf. Decision and Control*, Dec. 1993, pp. 2700–2705.

[2] M. J. van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *Int. J. Robust Nonlinear Control*, vol. 8, pp. 995–1020, Sep. 1998.

[3] F. Bullo and K. M. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Trans. Robot. Automat.*, vol. 17, no. 4, pp. 402–412, Apr. 2001.

[4] K. A. McIsaac and J. Ostrowski, "A framework for steering dynamic robotic locomotion systems," *Int. J. Robot. Res.*, vol. 22, no. 2, pp. 83–97, 2003.

[5] R. Brockett, "On the computer control of movement," in *Proc. 1988 IEEE Conf. Robotics and Automation*, 1988, pp. 534–540.

[6] ——, "Formal languages for motion description and map making," in *Robotics*. Providence, RI: AMS, 1990, pp. 181–193.

[7] V. Manikonda, P. S. Krishnaprasad, and J. Hendler, "Languages, behaviors, hybrid architectures and motion control," in *Mathematical Control Theory*, J. W. J. Baillieul, Ed. New York: Springer-Verlag, 1998, pp. 199–226.

[8] D. Hristu-Varsakelis, P. S. Krishnaprasad, S. Andersson, F. Zhang, P. Sodre, and L. D'Anna, The MDLe Engine: A Software Tool for Hybrid Motion Control Inst. Syst. Res., Tech. Rep. TR2000-54, Oct. 2000.

[9] D. Hristu-Varsakelis, M. Egerstedt, and P. S. Krishnaprasad, "On the structural complexity of the motion description language MDLe," in *Proc. 42nd IEEE Conf. Decision and Control*, Dec. 2003, pp. 3360–3365.

[10] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA J. Guid., Control, Dyna.*, vol. 25, no. 1, pp. 116–129, 2002.

[11] ——, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1077–1091, Dec. 2005.

[12] M. Egerstedt and C. Martin, "Conflict resolution for autonomous vehicles: a case study in hierarchical control design," *Int. J. Hybrid Syst.*, vol. 2, pp. 221–234, 2002.

[13] G. Pappas and S. Simic, "Consistent hierarchies of nonlinear abstractions," in *Proc. 39th IEEE Conf. Decision and Control*, Dec. 2000, pp. 4379–4384.

[14] G. Pappas, G. Lafferriere, and S. Sastry, "Hierarchically consistent control systems," *IEEE Trans. Autom. Control*, vol. 45, no. 6, pp. 1144–1160, Jun. 2000.

[15] P. E. Caines and Y.-J. Wei, "Hierarchical hybrid systems: A lattice theoretic formulation," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 1–8, Apr. 1998.

[16] E. Lemch and P. E. Caines, "Hierarchical hybrid systems; partition deformations and applications to the acrobot system," Int. Workshop on Hybrid Systems: Computation and Control ser. LNCIS, vol. 1386, T. A. Henzinger and S. Sastry, Eds. New York, Springer-Verlag, 1998, pp. 237–252.

[17] M. Egerstedt and R. Brockett, "Feedback can reduce the specification complexity of motor programs," *IEEE Trans. Robot. Automat.*, vol. 48, no. 2, pp. 213–223, Feb. 2003.

[18] M. Egerstedt and D. Hristu-Varsakelis, "Observability and policy optimization for mobile robots," in *Proc. 41st IEEE Conf. Decision and Control*, Dec. 2002, pp. 3596–3601.

[19] D. Hristu-Varsakelis and S. Andersson, "Directed graphs and motion description languages for robot navigation and control," in *Proc. IEEE Conf. Robotics and Automation*, 2002, pp. 2689–2694.

[20] S. Andersson and D. Hristu-Varsakelis, "Stochastic language-based motion control," in *Proc. 42nd IEEE Conf. Decision and Control*, Dec. 2003, pp. 3313–3318.

[21] E. Frazzoli, "Maneuver-based motion planning and coordination for multiple unmanned aerial vehicles," in *Proc. AIAA/IEEE Digital Avionics Systems Conf.*, 2002, pp. 8D3–1–8D3–12.

[22] E. Klavins, "A computation and control language for multi-vehicle systems," in *Proc. 42nd IEEE Conf. Decision and Control*, Dec. 2003, pp. 4133–4139.

[23] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994.

[24] F. Zanichelli, "Topological maps and robust localization for autonomous navigation," in *Proc. Int. Joint Conf. Artificial Intelligence, Workshop ROB-2*, 1999.

[25] R. Simmons and S. Koenig, "Probabilistic navigation in partially observable environments," in *Proc. Int. Joint Conf. Artificial Intelligence*, 1995, pp. 1080–1087.

[26] A. Briggs, C. Detweiler, and D. Scharstein, "Expected shortest paths for landmark-based robot navigation," *Int. J. Robot. Res.*, vol. 23, no. 7–8, pp. 717–728, 2004.

[27] J. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[28] Y. Hwang and N. Ahuja, "Gross motion planning—a survey," *ACM Comput. Surveys*, vol. 24, no. 3, pp. 219–291, 1992.

[29] J.-P. Laumond, Ed., *Robot Motion Planning and Control*, ser. Lecture Notes in Control and Information Sciences. New York: Springer-Verlag, 1998, vol. 229.

[30] R. Madhavan and H. Durrant-Whyte, "Natural landmark-based autonomous vehicle navigation," *Robot. Auton. Syst.*, vol. 46, pp. 79–95, 2004.

[31] A. Schultz, W. Adams, and B. Yamauchi, "Integrating exploration, localization, navigation and planning with a common representation," *Auton. Robot.*, vol. 6, no. 3, pp. 293–308, Jun. 1999.

[32] B. Kuipers, "The spatial semantic hierarchy," *Art. Intell.*, vol. 119, pp. 191–233, 2000.

[33] R. M. J. E. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2nd ed. Reading, MA: Addison-Wesley, 2000.

[34] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Trans. Robot. Automat.*, vol. 17, no. 3, pp. 242–257, Jun. 2001.

[35] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learn. Auton. Robot.*, vol. 31, no. 5, pp. 1–25, 1998.

[36] M. Martin and H. Moravec, Robot Evidence Grids The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-96-06, 1996.

[37] S. Thrun, "Bayesian landmark learning for mobile robot localization," *Machine Learn.*, vol. 33, no. 1, pp. 41–76, 1998.

[38] A. Lazanas and J.-C. Latombe, "Landmark-based robot navigation," *Algorithmica*, vol. 13, no. 5, pp. 472–501, May 1995.

[39] A. Lambert and T. Fraichard, "Landmark-based safe path planning for car-like robots," in *Proc IEEE Int. Conf. Robotics and Automation*, 2000, pp. 2046–2051.

[40] S. Fabrizi and T. Saffioti, "Extracting topology-based maps from gridmaps," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2000, pp. 2972–2978.

[41] B. Zavidovique, "First steps of robotic perception: the turning point of the 1990s," *Proc. IEEE*, vol. 90, no. 7, pp. 1094–1112, Jul. 2002.

[42] R. Luo, C. Yih, and K. Su, "Multisensor fusion and integration: approaches, applications, and future research directions," *IEEE Sensors J.*, vol. 2, no. 2, pp. 107–119, Apr. 2002.

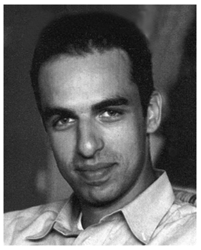
[43] D. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995, vol. 1.

[44] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, *Particle Filters for Mobile Robot Localization* (in Sequential Monte Carlo Methods in Practice), A. Doucet, N. de Freitas, and N. Gordon, Eds. New York: Springer-Verlag, 2000.

[45] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2000, pp. 1225–1232.

Sean B. Andersson (M'00) received the the B.S. degree in applied and engineering physics from Cornell University, Ithaca, NY, the M.S. degree in mechanical engineering from Stanford University, Stanford, CA, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, in 1994, 1995, and 2002, respectively.

He is a Faculty Member in the Department of Aerospace and Mechanical Engineering at Boston University, Boston, MA. His research interests include symbolic control, robotics, and control applications in scanning probe microscopy.



Dimitrios Hristu (M'00–SM'05) received the M.S. degree in applied mathematics and the Ph.D. degree in engineering sciences, both from the Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA, in 1997 and 1999, respectively.

He is a Faculty Member in the Department of Applied Informatics at the University of Macedonia, Greece. During 2000–2005, he was an Assistant Professor in the Department of Mechanical Engineering at the University of Maryland, College Park, and held a joint appointment with the Institute for Systems Research from 2002 to 2005. His research interests include control under limited communication, language-based control, and modeling of socio-economic systems.

Dr. Hristu-Varsakelis is a member of SIAM. He is a co-recipient of the 1999 Eliahu Jury Award from the Division of Engineering and Applied Sciences, Harvard University, and a co-recipient of the 2005 IFAC Young Author Prize.