Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

ThA08-2

Stochastic Language-based Motion Control

Sean Andersson
Electrical and Computer Engineering and
Institute for Systems Research
University of Maryland
College Park, MD 20742
sanderss@isr.umd.edu

Dimitrios Hristu-Varsakelis
Mechanical Engineering and
Institute for Systems Research
University of Maryland
College Park, MD 20742
hristu@isr.umd.edu

Abstract— In this work we present an efficient environment representation based on the use of landmarks and language-based motion programs. The approach is targeted towards applications involving expansive, imprecisely known terrain without a single global map. To handle the uncertainty inherent in real-world applications a partially-observed controlled Markov chain structure is used in which the state space is the set of landmarks and the control space is a set of motion programs. Using dynamic programming, we derive an optimal controller to maximize the probability of arriving at a desired landmark after a finite number of steps. A simple simulation is presented to illustrate the approach.

I. INTRODUCTION

As systems theory reaches into the domain of multi-modal systems, it reveals a complexity of behavior that is not usually encountered in classical models. This complexity is part of what motivates research in the subject but at the same time it gives rise to new challenges when it comes to answering basic system-theoretic questions in the new setting. This point is perhaps most easily illustrated in the following example: knowing that a mobile robot or other autonomous system is controllable (by checking the properties of a governing differential equation) does not tell us whether it is possible (or how) to steer the robot between two locations in a reasonably complex environment. The reasons for this difficulty are twofold. First, the environment is at best only locally state space-like, with regions that are uninteresting or should be avoided. Second, a complex environment makes it difficult to design control laws, especially if one insists on doing so at the level of sensors and actuators.

Efforts to address the latter challenge have included research on the "motion description languages" MDL and MDLe [1], [2], [3] which provide a means for abstracting from the low-level details (e.g. kinematics and dynamics) of a control system. Control programs written in these languages combine feedback control laws and logic into strings that have meaning almost independently of the underlying system, much like desktop software achieves a level of hardware independence by relying on appropriate device drivers.

The design of a motion description language shapes the set of control laws that can be formulated, as does the choice of a representation for the environment. After all, feedback control is a map between observations and inputs. Perhaps then it should come as no surprise that language can be useful not only for expressing control tasks but also for describing the environment. In particular, [4] proposed representing the environment of a language-driven dynamical system by means of landmarks, linked together not by geometric relations but by the *feedback control laws required to move from one location to another*. This gives rise to a directed graph, with nodes corresponding to landmarks and edges being identified with control programs encoded in the motion description language MDLe [2], [3]. This representation of the world makes contact with studies on human and animal navigation (see, e.g., [5]) that suggest the existence of two navigation systems used by mammals: a local response system and a global place-knowledge system. In simple terms, when the goal location is visible local information is used to navigate; when moving to locations which are not visible, stored knowledge of the spatial structure of the world is used. Although landmark-based navigation has been explored extensively by other authors for localization [6], [7], navigation [8], [9] and descriptions of "large-scale" environments [10], the novelty of the approach in [4] is that geometric relationships and global coordinates are abandoned in favor of language-based instructions that can be interpreted down to control laws suitable for driving a differential equation-based model. This results in a parsimonious description of the world, without the need for global geometry and without mapping areas that are easily navigable or uninteresting.

In this work we use [4] as a point of departure to study language-driven control and navigation in a stochastic setting. We exploit classical results on partially-observed controlled Markov chains to obtain control programs (more precisely strings in a formal language) that are optimal in the presence of uncertainty associated with the environment, the sensors and actuators of the system under consideration and with the precision of the language itself. The next section gives a brief description of MDLe. Section III presents the control problem we are concerned with and describes its Markov chain representation. In Section IV we derive control policies that are optimal for moving to a desired landmark. Section V contains simulation results that illustrate our approach.

II. MDLE

The starting point for MDLe is an underlying physical system such as a mobile robot with a set of sensors and

actuators for which we wish to specify a motion control program. The system is assumed to be governed by a differential equation of the form

$$\dot{x} = f(x) + G(x)u; \qquad y = h(x) \in \mathbf{R}^p \qquad (1)$$

where $x(\cdot) : \mathbf{R}^+ \to \mathbf{R}^n$ is the state of the system, $u(\cdot) : \mathbf{R}^p \times \mathbf{R}^+ \to \mathbf{R}^m$ is a control law of the type $u = u(t, h(x))$, and G is a matrix whose columns g_i are vector fields in $\mathbf{R}^n$. The simplest element of MDLe is the **atom**, defined to be a triple of the form $\sigma = (u, \xi, T)$, where u is as defined earlier, $\xi : \mathbf{R}^p \to \{0, 1\}$ is a boolean **interrupt** function defined on the space of outputs from p sensors, and $T \in \mathbf{R}^+$ denotes the value of time (measured from the time the atom is initiated) at which the atom will expire. To *evaluate* the atom is to apply the control law u until the interrupt ξ is low or until T units of time have elapsed. Atoms can be composed into a string, called a **behavior**, that carries its own interrupt function and timer. Behaviors can in turn be composed to form higher-level strings (called **partial plans**) and so on. We will use the term *plan* to refer to a generic MDLe string independent of the number of nested levels it contains. For more details on the language, including example programs, see [3].

III. LANDMARK-BASED NAVIGATION AMID UNCERTAINTY

We assume that there is a set, $\mathcal{L} = \{L_1, \ldots, L_n\}$, of "interesting" or useful geographical locations which we call **landmarks**. These landmarks can take various forms, such as GPS coordinates, visual cues, or evidence grid maps [11]. In general, however, they are identified with local geographical information only; that is they are not referenced to any global coordinate system. We associate to each landmark a sensor signature as follows. Let $s(t) \in \mathbf{R}^p$ be the sensor data collected at time t and let L be the current landmark taking values in $\{L_i\} \cup \emptyset$. Then

$$L = L_i \text{ if } s(t) = s_i(t) \quad t \in [t_0, t_0 + T] \qquad (2)$$

where $s_i(t)$, $t \in [t_0, t_0 + T]$ is the sensor signature of the ith landmark. We do not assume these signatures to be unique since a robot equipped with noisy sensors may at best be able to identify to within a subset of the collection of landmarks. We thus restrict our observations to the collection of equivalence classes where two landmarks are deemed equivalent if their signatures are "close" based on some metric. We refer to this set as $\mathcal{Z} = \{\tilde{L}_1, \ldots, \tilde{L}_p\}$ where $p \leq n$ and each $\tilde{L}_i$ is a representative of the equivalence class.

We will classify navigation tasks into two categories. The first involves motion on or near a landmark. In this setting the robot knows what landmark it is on and possesses a map of the nearby terrain. Assuming the robot can use its sensors to localize itself on this map, navigation is in principle solved by path planning. In this paper we are concerned with navigation *between* landmarks where, because we have assumed that we

do not have global geographical information, we cannot rely on any map. In the absence of sensing and actuator noise, one can replace geometric relationships between landmarks with instructions on how to get from one to the other [4]. The environment is then represented by a directed graph in which the nodes are the landmarks and edges are associated with MDLe plans. In order to be practical, this approach must be modified away from its deterministic setting, since we cannot guarantee that a given plan will perform as expected every time due to noisy sensing and control and environmental uncertainty.

To handle this uncertainty, we generalize the directed graph representation to a partially-observed controlled Markov chain. Given a collection of m MDLe plans denoted by $\mathcal{G} = \{\Gamma_1, \ldots, \Gamma_m\}$, we associate to each plan a Markov matrix, $A(k)$, specifying the transition probabilities between landmarks; thus $[A(k)]_{ij} = p_{ij}(k)$ is the probability of ending at landmark L_j given that we begin at landmark L_i and execute plan Γ_k. At the completion of each plan an observation is made, giving us information about the current landmark.

It is important to note that this choice of representation places some restrictions on the set of landmarks and plans. Since the system does not know with certainty which landmark it is on at the completion of a plan, the effect of applying each plan from each landmark must be known; this is precisely the meaning of the Markov matrix $A(k)$. Furthermore, each plan must guarantee that upon completion the system is at some landmark. A simple way of accomplishing this is, of course, to completely tile the world with landmarks. A more economical approach, however, is to choose plans carefully. For example, in an office environment it is possible to create plans which ensure the system will always end up inside an office rather than in a hallway, though due to changes in the environment such as people opening or closing their doors the particular office cannot be specified with certainty. Thus, the use of feedback control laws encoded as MDLe plans enables a simplified description of the environment in a manner akin to that by which feedback can reduce the complexity of motor programs [12].

IV. OPTIMAL NAVIGATION BETWEEN LANDMARKS

In order to use local navigation techniques the robot must know which landmark it is on. In this section, then, we propose a method of finding the sequence of MDLe plans that drives the robot to a desired landmark with maximal probability, in a time-optimal manner, under the assumption that such sequences exist. Recent work along these lines can be found in [13].

The navigation problem described in Section III is naturally discrete. To find the optimal sequence we turn to dynamic programming (DP) [14]. The state space for the robot is the collection of landmarks $\mathcal{L}$, the control space is the collection Γ of MDLe plans, and the observation

space is the collection of equivalence classes of landmarks, $\mathcal{Z}$. Let x_k, z_k, u_k be the state (location), observation, and control respectively at time k and let $k \in \{0, 1, \ldots, N\}$. We assume that we are given a sensor model for the robot; that is we know the distribution $\Pr(z_k = j | x_k = i)$ giving us the probability of making observation L_j given that we are currently on landmark L_i. Define the usual information vector

$$I_k \overset{\triangle}{=} (z_0, z_1, \ldots, z_k, u_0, u_1, \cdots, u_{k-1}) \tag{3}$$

and the vector of conditional probabilities

$$P_{k|k} = (p_{k|k}^1, p_{k|k}^2, \ldots, p_{k|k}^n)' \tag{4}$$

where $'$ indicates transpose and $p_{k|k}^j = \Pr(x_k = j | I_k)$ is the probability of being in state L_j at time k given the information up to the current time. Using Bayes rule and the assumption that the observation depends only on the state and not on the previous information or current control we have

$$
\begin{aligned}
p_{k+1|k+1}^j &= \Pr(x_{k+1} = j | I_{k+1}) \\
&= \frac{\Pr(z_{k+1}|x_{k+1} = j)\Pr(x_{k+1} = j | I_k, u_k)}{\sum_{i=1}^n \Pr(z_{k+1}|x_{k+1} = i)\Pr(x_{k+1} = i | I_k, u_k)}
\end{aligned} \tag{5}
$$

Now define

$$P_{k+1|k} = A(u)P_{k|k} \tag{6}$$

so that $\Pr(x_{k+1} = j | I_k, u_k) = [P_{k+1|k}]_j$. For ease of notation we also define the diagonal matrix

$$P_z = diag\left(\Pr(z|x_k = L_1), \ldots, \Pr(z|x_k = L_n)\right) \tag{7}$$

and the vector $e = (1, 1, \ldots, 1)'$. Using this notation equation (5) has the form

$$p_{k+1|k+1}^j = \frac{\Pr(z_{k+1}|x_{k+1} = j)\left[P_{k+1|k}\right]_j}{e'P_{z_{k+1}}P_{k+1|k}} \tag{8}$$

We can then write the update equation for the conditional probability as the two step iteration given by

$$P_{k+1|k} = A(u_k)P_{k|k} \tag{9}$$

$$P_{k+1|k+1} = \frac{P_{z_{k+1}}P_{k+1|k}}{e'P_{z_{k+1}}P_{k+1|k}} \tag{10}$$

where $P_{0|0}$ is a known initial distribution. To proceed with the DP algorithm we must choose the cost function we wish to minimize. We first choose to maximize the probability of arriving at a desired landmark, denoted d, at time N. To this end define the function

$$g_N(x) = \begin{cases} -1 & \text{if } x = d \\ 1 & \text{otherwise} \end{cases} \tag{11}$$

We denote a policy as $\pi = \{\mu_0, \mu_1, \ldots, \mu_N\}$ where μ_k is the control function at time k. The cost function we wish to minimize is

$$J_\pi(P_{0|0}) = E_{z_k, k=1,2,\ldots,N} \{E_x \{g_N(x_N)|I_N\}\} \tag{12}$$

subject to the dynamics of (9,10). The final cost is

$$
\begin{aligned}
J_N(P_{N|N}) &= E_x \{g_N(x)|I_N\} \\
&= \sum_{i=1}^n g_N(i) \left[P_{N|N}\right]_i = G_n' P_{N|N}
\end{aligned} \tag{13}
$$

where we have made the obvious definition for the vector G_N. Applying one step of the DP algorithm yields

$$
\begin{aligned}
J_{N-1}(P_{N-1|N-1}) &= \min_u E_{z_N} \{J_N(P_{N|N})\} \\
&= \min_u \sum_{i=1}^p G_N' P_{z_N=i} A(u) P_{N-1|N-1}
\end{aligned} \tag{14}
$$

Thus the optimal control at the $(N-1)^{th}$ step is

$$\mu_{N-1} = \arg\min_u \sum_{i=1}^n G_N' P_{z_N=i} A(u) P_{N-1|N-1} \tag{15}$$

which simply minimizes the expected value of the cost over the final observation. Carrying the DP algorithm one more step we find the $N-2$ stage cost to be

$$
\begin{aligned}
J_{N-2}\left(P_{N-2|N-2}\right) &= \min_u E_{z_{N-1}} \{J_{N-1}\left(P_{N-1|N-1}\right)\} \\
&= \min_u \sum_{i_1=1}^p \sum_{i_2=1}^p G_N' P_{z_N=i_1} A(\mu_{N-1}) \\
&\qquad \cdot P_{z_{N-1}=i_2} A(u) P_{N-2|N-2}
\end{aligned} \tag{16}
$$

The optimal control at time $N-2$ is thus

$$
\begin{aligned}
\mu_{N-2} = \arg\min_u \sum_{i_1=1}^p \sum_{i_2=1}^p G_N' P_{z_N=i_1} A(\mu_{N-1}) \\
\cdot P_{z_{N-1}=i_2} A(u) P_{N-2|N-2}
\end{aligned} \tag{17}
$$

which is the control which minimizes the expected value of the final cost over the last two observations. The general case is given by the following theorem.

Theorem 4.1: For $k = N-1, \cdots, 0$ the optimal cost to go is given by

$$
\begin{aligned}
J_k\left(P_{k|k}\right) = \min_u \sum_{i_1=1}^p \sum_{i_2=1}^p \cdots \sum_{i_{N-k}=1}^p G_N' P_{z_n=i_1} \\
\cdot A(\mu_{N-1}) P_{z_{N-1}=i_2} A(\mu_{N-2}) \cdots P_{z_{k+1}=i_{N-k}} A(u) P_{k|k}
\end{aligned}
$$

∎

The usual corollary yields the optimal control policy.
Corollary 4.2: The optimal control at time k is

$$
\begin{aligned}
\mu_k = \arg\min_u \sum_{i_1=1}^p \sum_{i_2=1}^p \cdots \sum_{i_{N-k}=1}^p G_N' P_{z_n=i_1} A(\mu_{N-1}) \\
\cdot P_{z_{N-1}=i_2} A(\mu_{N-2}) \cdots P_{z_{k+1}=i_{N-k}} A(u) P_{k|k}
\end{aligned}
$$

∎

A simple extension allows us to maximize this probability of arriving at the desired landmark in the minimum amount of time. To this end we define the functions

$$g_k(x) = \begin{cases} -b_k & x_k = d \\ b_k & \text{otherwise} \end{cases} \qquad (18)$$

and seek to minimize the cost function given by

$$J_\pi(P_{0|0}) = E_{z_k, k=1,\ldots,N} \left\{ G_N' P_{N|N} + \sum_{k=0}^{N-1} G_k' P_{k|k} \right\} \quad (19)$$

The DP solution is given by the following theorem.

Theorem 4.3: For $k = N - 1, \cdots, 0$ the optimal cost to go is given by

$$J_k(P_{k|k}) = \min_u \left[G_k' P_{k|k} \right.$$

$$+ \sum_{i_1=1}^{p} \left(G_{k+1}' P_{z_{k+1}=i_1} A(u) P_{k|k} \right)$$

$$+ \sum_{i_1=1}^{p} \sum_{i_2=1}^{p} \left(G_{k+2}' P_{z_{k+2}=i_1} A(\mu_{k+1}) P_{z_{k+1}=i_2} A(u) P_{k|k} \right)$$

$$+ \cdots + \sum_{i_1=1}^{p} \cdots \sum_{i_{N-k}}^{p} \left(G_N' P_{z_N=i_1} \right.$$

$$\left. \cdot A(\mu_{N-1}) \cdots P_{z_{k+1}=i_{N-k}} A(u) P_{k|k} \right) \right]$$

■

The optimal control follows immediately from this theorem. We note that while the complexity of finding the optimal control increases exponentially with the number of stages, it grows only *linearly* in the number of landmarks.

V. SIMULATION RESULTS

To illustrate the proposed representation and the derived optimal control laws, a simple simulator was developed. The robot is modeled as a direct drive system obeying the following nonholonomic kinematics

$$\dot{x} = u_f \cos(\theta), \qquad (20)$$

$$\dot{y} = u_f \sin(\theta), \qquad (21)$$

$$\dot{\theta} = u_\theta, \qquad (22)$$

where

$$u_f = \frac{u_L + u_R}{2}, \quad u_\theta = \frac{u_L - u_R}{w}. \qquad (23)$$

Here u_f and u_θ are the forward and heading velocities, u_L and u_R are the left and right wheel velocities, and w is the distance between the wheels. It is equipped with a set of range sensors. The environment is modeled by a set of polygons. The simulator accepts an MDLe plan specified as a list of atoms and at each time step the current interrupt function is evaluated. If it has fired the next atom is loaded and if not the control function is evaluated to determine u_L and u_R. To model actuator noise, independent samples from a normal distribution are added to u_L and to u_R. The system

equations are then integrated forward by one time step and the sensors evaluated by intersecting each ray with the set of polygons modeling the environment. The process then repeats until the list of atoms is exhausted.

The office-like environment used for these simulations is shown in Figure 1 together with a virtual robot. Three
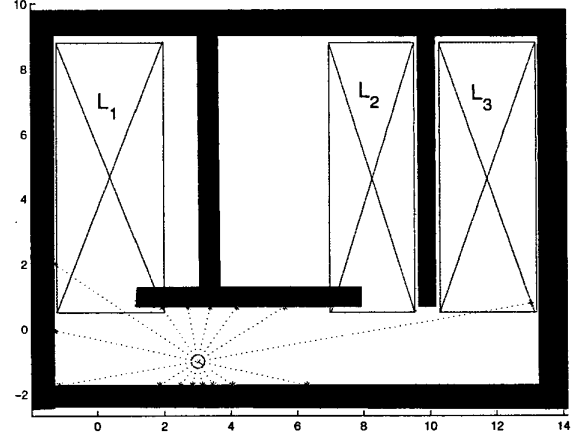

Fig. 1. Environment and robot

landmarks, denoted L_1, L_2, and L_3, were defined. Their (x, y) regions are shown in Figure 1. Each covered headings of $(-80, -100)$ degrees. The following control functions were created.

- *go* $[u_f \; u_\theta]$: Applies controls u_f and u_θ.
- *goAvoid* $[u_{f_N} \; d \; k_\theta]$: In the absence of obstacles within d of the front, sets $u_f = u_{f_N}$. If an object is detected within d, sets $u_f = u_{f_N}(d - r_{min})$ and $u_\theta = \pm k_\theta$ with the sign chosen to steer away from the obstacle. ($r_{min} =$ distance to obstacle.)
- *followWall* $[u_{f_N} \; k_f \; k_\theta \; d]$: Maintains distance and heading to wall by setting $u_f = -k_f(2(d - r_{min}) + \widehat{\theta}) \sin(\widehat{\theta})$ and $u_\theta = -k_\theta((d - r_{min}) + 2\widehat{\theta})$ where r_{min} is the measured distance to the closest side wall and $\widehat{\theta}$ is the estimate of the heading with respect to the wall. If both distance and heading errors are small then sets $u_f = u_{f_N}$ and $u_\theta = 0$.
- *alignWall* $[k_\theta]$: Sets $u_f = 0$ and $u_\theta = -k_\theta \widehat{\theta}$ where $\widehat{\theta}$ is the estimate of the heading with respect to the closest side wall.
- *rotateAway* $[k_\theta]$: Sets $u_f = 0$ and $u_\theta = -k_\theta \widehat{\theta}$ where $\widehat{\theta}$ is the estimate of the heading with respect to the rear wall.

The following interrupt functions were also defined.

- *wait* $[\tau]$: Fires after τ seconds.
- *sideOpen* $[side \; d \; \tau]$: Fires if sensor on side indicated by *side* (with 1 indicating left, 2 indicating right, and 3 indicating either) reads less than d or if τ seconds have passed.

- *alignedWall* [ψ τ]: Fires if the estimated heading with the nearest side wall is less than $|\psi|$ or if τ seconds have passed.
- *rotatedAway* [ψ τ]: Fires if the estimated heading with the rear wall is less than $|\psi|$ or if τ seconds have passed.
- *atWall* [d τ]: Fires if the front sensor reads less than d or if τ seconds have passed.

From these functions various atoms were constructed and from the atoms five plans were defined including the identity plan (denoted $\mathbb{I}$) which applies a zero control. The remaining four (L_1^2, L_1^3, L_2^3, and L_3^1) were designed to steer the robot in the absence of noise from landmark i to j. As an example, plan L_2^3 is

{ (sideOpen [3 6 5])   (followWall [1 20 2 0.4])
 (atWall [0.3 30])   (goAvoid [1 0.05 1 0.025])
 (wait [0.75])   (go [0 1.57])
 (alignedWall [5 10])   (alignWall [2])
 (wait [0.5])   (followWall [1.25 20 2 0.4])
 (sideOpen [1 6 5])   (followWall [1 20 2 0.4])
 (wait [0.5])   (go [0 1.57])
 (rotatedAway [3 0.1 5])   (rotateAway [3])
 (wait [3.5])   (goAvoid [1 0.4 1 0.025])
 (wait [2])   (go [0 1.57])
 (alignedWall [1 10])   (alignWall [2])}

where the notation is (interrupt) (control). This plan reads as follows. Follow the nearest wall until either side reads greater than six meters, then go straight until a wall is reached. Turn counter-clockwise, align along that wall, and follow it for half a second. Continue following the wall until the left side sensor reads greater than six meters. Rotate and align to the wall behind, move forward for three and a half seconds (but do not run into any intervening obstacles), and then rotate counter-clockwise 90°. Finally align to the wall.

It should be noted that the plans were chosen to be somewhat brittle with respect to the simulated noise. In L_2^3, for example, the robot attempts to detect the opening to the next room quickly. Due to noise the robot may not have moved far enough and the interrupt will fire too soon, causing the robot to end back on landmark two. While more robust plans could certainly be designed, some level of uncertainty was desired to show the use of the optimal controller.

The a priori observation probabilities were chosen to be (with the notation $Pr(i|j) = Pr(z = i|x = L_j)$)

$$Pr(1|1) = 0.5 \quad Pr(1|2) = 0.3 \quad Pr(1|3) = 0.2$$
$$Pr(2|1) = 0.2 \quad Pr(2|2) = 0.6 \quad Pr(2|3) = 0.1$$
$$Pr(3|1) = 0.3 \quad Pr(3|2) = 0.1 \quad Pr(3|3) = 0.7$$

The Markov matrices were determined by running each plan 100 times from each landmark. Actuator noise was sampled from a $\mathcal{N}(0, 0.01)$ distribution. The resulting Markov

matrices were

$$A_{L_1^2} = \begin{bmatrix} 0 & 0 & 0 \\ 0.43 & 0 & 0 \\ 0.57 & 1 & 1 \end{bmatrix} \quad A_{L_1^3} = \begin{bmatrix} 0 & 0 & 0 \\ 0.12 & 0 & 0 \\ 0.88 & 1 & 1 \end{bmatrix}$$

$$A_{L_2^3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad A_{L_3^1} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The optimal controller of Corollary 4.2 was used as follows. The state was initialized and a three-stage controller run to steer the robot to the desired landmark. At the end of three stages the probability vector was tested and if the probability of being at the desired landmark was less than 0.95 the process was repeated.

In Figures 2, 3, and 4 we show the evolution of the state, the true and observed landmarks, and the selected plan at each time step for a sample run with a true initial position on L_1, an initial state of a uniform distribution across the states, and a desired final position on L_2. This run shows the robustness of the approach to both the actuator and sensing noise; despite driving to an unintended location twice and getting several incorrect readings (including the final one) the controller was successful in achieving the objective.
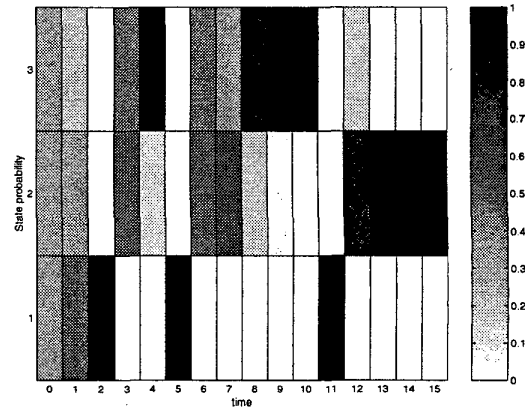

Fig. 2. L_1 to L_2: State evolution

VI. CONCLUSIONS

In this paper we presented an approach to landmark-based navigation for mobile robots intended for applications in expansive or sparse environments and designed to handle the noisy sensors and actuators one finds in real-world robotics. Under this approach the set of landmarks is viewed as a controlled Markov chain where the controls are feedback control laws encoded in a motion description language. Global information is thus replaced by local information around each landmark and the connections between those landmarks.

An optimal controller was developed using dynamic programming that maximizes the probability of steering the
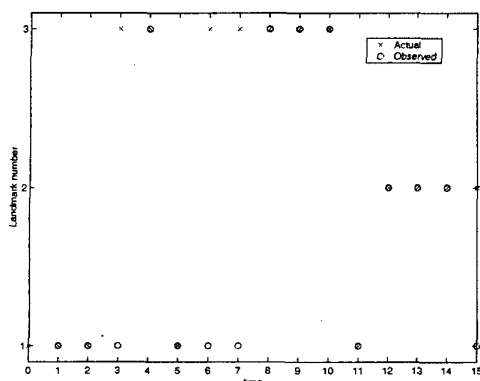
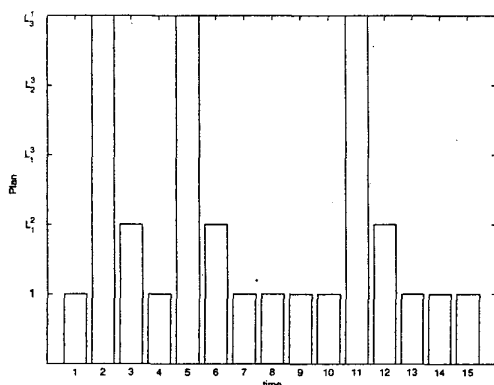Fig. 3. L_1 to L_2: True and observed landmarks

Fig. 4. L_1 to L_2: Executed plans

robot to a desired landmark in N steps. This controller was applied to a simulated robot and a typical run presented. The simulation shows the robustness to actuator and sensor noise afforded to the controller by the design of the underlying framework. We note that the controller presented here is quite simple one; more effective ones can certainly be designed.

There are several areas of ongoing work. We are currently implementing the approach on a physical system in a large environment. Since it is not practical to run a plan thousands of times in the physical world, we are developing a simulator which interfaces to our implementation of MDLe [3] to determine the Markov matrices. We are also exploring techniques to identify which landmark the robot is currently on, questions about when we can uniquely localize ourselves on a given set of landmarks (an observability question related to work in [13]), and how to autonomously explore an unknown environment and develop the Markov-chain based representation proposed here.

VII. ACKNOWLEDGEMENTS

VIII. REFERENCES

[1] R. W. Brockett. Hybrid models for motion control systems. In H. Trentelman and J. Willems, editors, *Perspectives in Control*, pages 29–54. Birkhäuser, Boston, 1993.

[2] V. Manikonda, P. S. Krishnaprasad, and J. Hendler. Languages, behaviors, hybrid architectures and motion control. In J. Baillieul and J.C. Willems, editors, *Mathematical Control Theory*, pages 199–226. Springer, 1998.

[3] D. Hristu, S. Andersson, F. Zhang, P. Sodre, and P.S. Krishnaprasad. A motion description language for hybrid systems programming. *submitted to IEEE Transactions on Robotics and Automation*, 2003.

[4] D. Hristu-Varsakelis and S. Andersson. Directed graphs and motion description languages for robot navigation and control. In *Proc. of the IEEE Conf. on Robotics and Automation*, pages 2689–2694, 2002.

[5] T. J. Prescott. Spatial representation for navigation in animats. *Adaptive Behavior*, 4(2):85–123, 1996.

[6] A. Bandera, C. Urdiales, and F. Sandoval. Autonomous global localisation using Markov chains and optimised sonar landmarks. In *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 288–293, 2000.

[7] B. Yamauchi. Mobile robot localization in dynamic environments using dead reckoning and evidence grids. In *Proc. of the IEEE Int. Conference on Robotics and Automation*, pages 1401–1406, 1996.

[8] A. Schultz, W. Adams, and B. Yamauchi. Integrating exploration, localization, navigation and planning with a common representation. *Autonomous Robots*, 6(3):293–308, June 1999.

[9] A. Lambert and Th. Fraichard. Landmark-based safe path planning for car-like robots. In *Proc of the IEEE Int. Conference on Robotics and Automation*, pages 2046–2051, 2000.

[10] B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[11] M. Martin and H. Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, The Robotics Institute, Carnegie Mellon University, 1996.

[12] M. Egerstedt and R. Brockett. Feedback can reduce the specification complexity of motor programs. *submitted to IEEE Trans. Robotics and Automation*, 2002.

[13] M. Egerstedt and D. Hristu-Varsakelis. Observability and policy optimization for mobile robots. In *Proc. of the 2002 IEEE Conference on Decision and Control*, pages 3596–3601, 2002.

[14] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 1995.