Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Language-based Feedback Control Using Monte Carlo Sensing *

Sean B. Andersson
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
sanderss@deas.harvard.edu

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

***Abstract*— Landmark-based graphs are a useful and parsimonious tool for representing large scale environments. Relating landmarks by means of feedback-control algorithms encoded in a motion description language provides a level of abstraction that enables autonomous vehicles to navigate effectively by composing strings in the language to form complex strategies that would be difficult to design at the level of sensors and actuators. In such a setting, feedback control requires one to pay attention not only to sensor and actuator uncertainty, but also to the ambiguity introduced by the fact that many landmarks may look similar when using a modest set of observations. This work discusses the generation of language-based feedback control sequences for landmark-based navigation together with the problem of sensing landmarks sufficiently well to make feedback meaningful. The paper makes two contributions. First, we extend previous work to include the costs of sensing with varying degrees of accuracy. Second, we describe a Monte Carlo based approach to landmark sensing which relies on the use of particle filters. We include simulation results that illustrate our approach.**

***Index Terms*— Motion description language, Mobile robot motion-planning, Dynamic programming, Hidden Markov models**

I. Introduction

The design of feedback control laws to accomplish seemingly straightforward tasks, such as navigation in everyday environments or object manipulation, is arguably a persistent challenge in the area of robotic motion control. This is partly because much of systems theory applies in domains (defined by various structural requirements on the dynamics of a system) which are too narrow to capture a variety of interesting and realistic situations. The available control design tools are particularly effective for systems that evolve in state-space like environments and for control specifications which are narrowly defined (e.g. stabilization or trajectory tracking). While such tools are necessary in almost every application, they are useful only locally in time and space. For example, attempting to devise a state feedback law that will steer a robot through a moderately sized building with doors, human traffic, elevators, etc., quickly gets one mired in complexity and leads to the idea of "breaking-up" the task into many intermediate pieces.

Attempts to do precisely this have seeded research in the area of motion description languages [1]–[3], or MDLs, via which motion control tasks are described by symbolic strings. Those strings are themselves composed from simple control primitives that are eventually interpreted down to precise feedback control laws. The specification of motion control in terms of language agrees with human intuition and enables one to manage the complexity of motion control tasks. Furthermore, language-based descriptions have a chance of being universal because the same set of instructions can be interpreted by robots that differ in their kinematics, mass, sensor configuration, etc., to produce the same effect. Most importantly, linguistic descriptions of control tasks provide a useful abstraction that allows one to design feedback control laws at the level of strings and primitives (each with a mathematical interpretation as a feedback controller), rather than at the level of sensors and actuators. Although this is perhaps one of the greatest potential advantages of language-based control, it has only recently begun to be explored [4]–[8].

One of the problems where the benefits of language-based control are most evident is that of navigation and localization. If a robot is given a map of its environment and sensors to investigate its surroundings, the navigation problem can be solved effectively using a variety of path-planning techniques (see e.g. [9]–[11]), while efficient global localization can be achieved using Monte Carlo methods [12]. These approaches become infeasible as the size of the environment becomes large compared to the robot's sensing range. At the same time, it is often the case that only a small portion of the environment is "interesting". For example, in an office environment hallways are often used only as a means for moving from office to office. Language-based feedback control laws (formalized in an MDL) can be used to form sequences of instructions that steer a robot from one interesting region to another. Each of these regions, termed *landmarks*, is endowed with its own local map on which traditional map-based navigation and localization can be performed. This approach yields a natural but parsimonious representation of the environment [6] which incorporates sensor and actuator uncertainties [7]. Furthermore, it suggests favoring linguistic instructions (to

*This work was supported by the NSF under Grant No. EIA0088081 and by ARO ODDR&E MURI01 Grant No. DAAD19-01-1-0465, (Center for Communicating Networked Control Systems, through Boston University).

be interpreted down to sensor and actuator-level feedback laws) over geometric relationships and global coordinates.

In a landmark-based setting, several methods have been investigated for localization (e.g. [13]–[15]) and for navigation (e.g. [16]–[18]). In this work we explore the construction of optimal motion control sequences, paying special attention to the problem of sensing landmarks. Rather than being just simple features in the environment, we allow landmarks to be (small) regions, such as a corner in a hallway or the area around a door. To navigate effectively, a robot must determine which landmark it is on using multiple measurements from its sensor suite. Here we propose a Monte Carlo-based method to accomplish this. The essential idea is to derive a probability distribution over the set of landmarks (using the results of independent particle filter-based localization algorithms) and to use that distribution to inform the navigation problem.

Of course, the accuracy with which landmarks can be "identified" depends on the type and quality of the sensors used, the amount of data which is used to make the measurement and the "uniqueness" of the landmark. There is a tradeoff between the cost of operating the sensors, the time spent collecting data, the computational cost, and the reliability of the measurement. It is thus desirable, and in fact natural in our formulation, to include a measure of the cost of sensing in the performance functional which is to be optimized.

II. Stochastic language-based motion control

As described in [7], we let $\mathcal{L} = \{L_1, \ldots, L_{n_L}\}$ denote a collection of interesting or useful geographical areas in the environment. We call these areas *landmarks* and associate to each a local map and coordinate system. The problems of navigation and localization are each divided into a pair of subproblems, namely the local problem of navigation and localization on a given landmark, and the global problem of navigation between landmarks and localization on the set of landmarks. Since each landmark is equipped with a map, the local problem can in principle be solved through the use of map-based path-planning and localization techniques. Our focus here will be on the global version of the problem.

When traveling between landmarks, the robot has no global geographical information and therefore map-based path-planning and localization algorithms cannot be used. In lieu of the geographical information, we equip the robot with a set of feedback-control laws encoded as sequences of motion control primitives in the motion description language MDLe. Each such sequence is known as a *plan* and steers the robot through the environment from landmark to landmark. Due to the noise inherent in real world sensors and actuators and random (small) changes in the environment, the actual outcome of a plan cannot be known exactly even if the robot knows with certainty where it begins. We therefore represent the action of each plan by a Markov matrix $A(i)$ whose jk-th element gives the probability of ending on landmark k given that the robot started on landmark j.

At the completion of a plan the robot makes an observation as to which landmark it is currently on. We do not assume that each landmark is uniquely identifiable since, depending on the sensor suite of the robot, different landmarks may appear to be similar to varying degrees. Therefore we define the set $\mathcal{Z} = \{z_1, \ldots, z_m\}$, $m \leq n$, to be the collection of possible *observation outcomes*. This can be viewed as the set of equivalence classes of the landmarks where two landmarks are deemed *equivalent* if they cannot be distinguished using only measurements taken while on either of the two. To generate an observation, the robot collects sensor measurements from its current local environment and uses them to obtain a measurement from $\mathcal{Z}$. To model this process, we define an *observation policy* to be a motion plan which moves the robot locally (on a fixed landmark), while gathering data using a collection of sensors. These sensor measurements are uncertain. We thus associate to each observation policy a Markov matrix $O(i)$ whose jk-th element gives the probability of observing z_k given that the robot is on landmark j.

To mathematically formulate the global navigation and localization problem we take the set $\mathcal{L}$ of landmarks as the state space for the robot, the set $\mathcal{Z}$ as the observation space, and define the sets $\mathcal{U} = \{\mathcal{A}(1), \ldots, \mathcal{A}(n_c)\}$ of control actions and $\mathcal{O} = \{O(1), \ldots, O(n_o)\}$ of observation policies. Let x_k, u_k, o_k, z_k denote the actual landmark, control action, observation policy, and observation at time k, $k \in \{0, 1, \ldots, N\}$. Note that in this framework time is naturally a discrete variable which is updated after the completion of a control action and observation policy. Let I_k denote the usual information vector

$$I_k \stackrel{\triangle}{=} (u_0, o_0, z_0, \ldots, u_k, o_k, z_k) \tag{1}$$

and define the row vector of conditional probabilities

$$P_{k|k} \stackrel{\triangle}{=} \begin{pmatrix} p^1_{k|k} & \ldots & p^{n_L}_{k|k} \end{pmatrix} \tag{2}$$

where $p^i_{k|k}$ is the probability of being on landmark i given I_k. Using Bayes rule we can update these probabilities after executing a control action, an observation policy, and generating an observation. We have

$$p^i_{k+1|k+1} = \frac{Pr(z_{k+1}|x_{k+1}=i, o_{k+1}) Pr(x_{k+1}=i|I_k, u_k)}{\sum_{i=1}^m Pr(z_{k+1}|x_{k+1}=i, o_{k+1}) Pr(x_{k+1}=j|I_k, u_k)} \tag{3}$$

where

$$Pr(x_{k+1} = i|I_k, u_k) = \sum_{j=1}^n Pr(x_{k+1} = i|x_k = j, u_k) p^j_{k|k} \tag{4}$$

and we have made the Markov assumption that the observation z_{k+1} depends only on the current landmark and observation policy. For ease of notation define the diagonal matrix

$$P_{z_k}(o_k) = \text{diag}(Pr(z_k|x_k = 1, o_k), \ldots, Pr(z_k|x_k = n_L, o_k)) \tag{5}$$

and the column vector $e = (1, \ldots, 1)'$. The update equation for the conditional probability vector can then be written compactly as

$$P_{k+1|k+1} = \frac{P_{k|k} A(u_k) P_{z_k}(o_k)}{P_{k|k} A(u_k) P_{z_k}(o_k) e}. \tag{6}$$

Note that the observable z_k is a random variable on $\mathcal{Z}$ whose distribution is defined by the choice of o_k.

The probability of arriving at a desired landmark clearly depends on the sequence of control actions and observation policies. In addition to their different distributions, there may be different costs associated to the various actions and policies. For example, a robot could make a quick observation involving just a few measurements to get a rough estimate of its surroundings or it could spend more time to investigate the local details. Similarly, a control action may move the robot very quickly at the expense of accurate sensing while another may move the robot very slowly to minimize errors. To handle such differences, we define the cost functions

$$g_u : \mathcal{P} \times \mathcal{U} \times \{0, \ldots, N-1\} \to \mathbb{R}, \tag{7}$$

$$g_o : \mathcal{P} \times \mathcal{O} \times \{0, \ldots, N-1\} \to \mathbb{R}. \tag{8}$$

where $\mathcal{P}$ is the space of distributions over $\mathcal{L}$. Let π denote a policy $\pi = (u_0, o_0, \ldots, u_{N-1}, o_{N-1})$. We then define the optimal control problem

$$\begin{array}{l} \min_\pi J_\pi(P_{0|0}) = E_{z_k, k=0,\ldots,N-1} \left\{ g(P_{N|N}, N) \right. \\ \left. + \sum_{k=0}^{N-1} \left(g_u(P_{k|k}, u_k, k) + g_o(P_{k|k}, o_k, k) \right) \right\} \end{array} \tag{9}$$

which naturally fits into the framework of dynamic programming (DP) [19]. The resulting feedback controller, selected via DP, is a sequence of motion and observation plans which seek to minimize J. By choosing appropriate cost functions, a variety of different goals can be achieved. For example, to maximize the probability of arrival at a desired landmark in N steps, one could choose the final cost to be $g(P_{N|N}, N) = -P_{N|N} d$ where d is a column vector containing a 1 in the index of the desired landmark and a 0 in every other position. To minimize the actual time it takes to move to a given landmark, one could choose the per-stage cost to be proportional to the expected time to complete the selected control and observation plans.

III. Monte Carlo-based landmark sensing

To generate an observation of the current landmark, the robot must fuse a number of sensor measurements collected at different times and different positions from possibly disparate sensors. There are a variety of ways to handle this problem; here we propose a solution based on the technique of Monte Carlo (or *particle filter*) localization. Particle filtering is a grid-less simulation-based filtering technique in which the probability density of a stochastic process is represented by a collection of samples (particles) generated by a Monte Carlo method. It has been used effectively in a variety of estimation problems including localization in mobile robotics [12].

A. Particle filters

We outline here the basic particle filter algorithm for a stochastic system with discrete dynamics. See [20] for a more complete description. Consider the discrete-time stochastic dynamical system given by

$$\begin{array}{rcl} x_{k+1} & = & f_k(x_k, u_k) + G_k(x_k) w_k, \\ y_k & = & h_k(x_k) + \nu_k \end{array} \tag{10}$$

where w_k, ν_k are independent noise processes and the distribution of x_0 is given and independent of w_k, ν_k. Define $\mathcal{D}_k = \{y_0, u_0, \ldots, y_k, u_k\}$ to be the collection of observations and controls up to time k. The propagation of the conditional density is given by first diffusing the density

$$p_{k+1|k}(x_{k+1}|\mathcal{D}_k, u_{k+1}) = \int p(x_{k+1}|x_k, u_k) p_{k|k}(x_k|\mathcal{D}_k) dx_k$$

and then updating it according to Bayes' rule

$$p_{k+1|k+1}(x_{k+1}|\mathcal{D}_{k+1}) = \alpha p(y_{k+1}|x_{k+1}) p_{k+1|k}(\mathcal{D}_k, u_{k+1})$$

where α is a normalizing factor. The density $p(x_{k+1}|x_k, u_k)$ is termed the *motion model* and describes the effect of the control action on the state of the system. The density $p(y_{k+1}|x_{k+1})$ is called the *sensor model* and is the probability density of the observation given the state of the system.

Under some mild assumptions, the resulting conditional density is exact but generally corresponds to an infinite dimensional filter. Particle filtering is an approximation method that mimics the above calculations using a finite number of operations. The algorithm is as follows.

1. Initialization: Sample N_p particles $x_0^1, \ldots, x_0^{N_p}$ according to $p_0(x)$.
2. Diffusion: Find $\hat{x}_{k+1}^1, \ldots, \hat{x}_{k+1}^{N_p}$ from $x_k^1, \ldots, x_k^{N_p}$ using the dynamic rule (10).
3. Form the empirical distribution:
$$p_{k+1|k}^{N_p}(x) = \frac{1}{N_p} \sum_{j=1}^{N_p} \delta_{\hat{x}_{k+1}^j}(x)$$
4. Use Bayes' rule:
$$p_{k+1|k+1}(x) = \frac{\alpha}{N_p} \sum_{j=1}^{N_p} \delta_{\hat{x}_{k+1}^j}(x) \Psi_{k+1}(x)$$
where α is a normalizing constant.
5. Re-sample: Sample $x_{k+1}^1, \ldots, x_{k+1}^{N_p}$ according to $p_{k+1|k+1}(x)$.

Here $\delta_v(w)$ is the Dirac delta function and $\Psi_k(x)$ is the conditional density of the observation y_k given the state x, i.e. the sensor model. It has been shown that, under some technical assumptions, the approximate density of this algorithm converges almost surely to the true density as the number of particles goes to infinity [21].

Note that the total weight of the particles before normalization, given by

$$C_{k+1} = \sum_{j=1}^{N} \delta_{\hat{x}_{k+1}^j}(x) \Psi_{k+1}(x), \tag{11}$$

provides an indication as to how likely the current set of particles is given the current observation. We will take advantage of this below to generate a distribution on the space of observation outcomes.

B. Particle filters for mobile robot localization

The problem of localization for a mobile robot is that of specifying its position and orientation with respect to a fixed coordinate system. If the actuators and sensors are noisy, the problem becomes one of estimation of the probability density over the space of possible positions and orientations of the robot, i.e. over its position and heading angle with respect to the fixed coordinate system. In this case, the motion model is given by specifying a stochastic dynamical system describing the evolution of the robot, while the sensor model provides a probability density function over the possible sensor readings given a position and orientation on the map.

C. Particle filters for landmark sensing

The total weight of the particles before normalization, C_{k+1}, gives information as to how likely the current set of particles is. If more particles are in locations which are more likely to yield the current sensor readings, then the total weight will be higher. This fact can be used to provide a observation of the current landmark as follows.

At the completion of each motion plan, the robot executes a sensing plan to move locally while gathering data. If the robot were known to be on a given landmark, then this would be the standard localization problem, which can be solved using the particle filter algorithm. The algorithm translates the sensor data obtained while running the observation plan into a pdf over all possible headings and all locations *on the given landmark map.*

As described in Section II, among the n landmarks there are m possible observation outcomes. Independent particle filter-based localization algorithms can be run on each of the m maps simultaneously as the robot executes the observation plan. At the completion of the plan, the total weight of the particles on each map, $C^i, i = 1, \ldots, m$ can be calculated. An observation from the set of possible observations is generated by sampling from the probability distribution over the m possible observations given by

$$\text{prob(observation } i) = \frac{C^i}{\sum_{j=1}^{m} C^j}. \tag{12}$$

D. Comments

The efficacy of this approach is strongly influenced by the number of particles chosen for each particle filter algorithm. We note that this number need not be the same for each landmark; if a landmark has many identifiable features that make localization easy then fewer particles can be used. However, using an insufficient number of particles on the landmarks 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 computation time) and the quality of the observation. This tradeoff can be captured by assigning different costs to different choices for the number of particles used and optimizing these costs under the framework of Section II.

The particle filter algorithm is a powerful technique applicable to arbitrary distributions. However, it is computationally intensive. The algorithm presented here requires not just one but m instances of the algorithm, one for each possible observation outcome. This additional complexity is offset by two factors. First, high accuracy is not necessarily required; so long as the resulting distribution is not uniform the robot can be expected to determine where it is with high probability in the long term by running sequences of motions and landmark observations (see also [22] for related work). Second, the particle filter algorithm is inherently parallelizable and thus the additional complexity can be handled using multiple processors.

It is important to note that the particle filter algorithm should be run only while the observation plan is being executed. On small maps a particle filter tends to converge fairly quickly to a sharp distribution, even if the local environment of the robot looks very different than that of the landmark. If the filter were to be run during the motion plan phase, then the robot would begin the observation phase with a highly localized but most likely erroneous distribution, akin to the "kidnapped robot" problem. While there are variants of the basic particle filter-based localization algorithm which can effectively solve the kidnapped robot problem [12], the additional complications can be avoided simply by running the particle filters only when observing the landmark.

Finally, once the robot has determined with high probability that it has arrived at a given landmark, the final estimate of the distribution can be used to initialize a particle filter localization algorithm run on that map.

IV. Simulations

To illustrate our approach to global navigation and localization, we now present a simulation-based experiment of a planar, direct-drive nonholonomic robot equipped with a laser range-finder sensor operating in an office-like environment (shown in Fig. 1). In the image, white denotes open space, black denotes occupied space, and gray denotes no knowledge. Superimposed on the map are nine landmarks , one at each of the four hallway corners, one at each of the T-intersections, and one at each of the three doorways. We classify these landmarks into four groups- *corner*, *T*, *left doorway*, and *right doorway*, giving us four possible observation outcomes. Each landmark is represented as an occupancy grid map [23] with cells of $10cm$ square. The landmarks are numbered from 1 to 9 beginning with the upper left corner and proceeding clockwise around the map.

The equations of motion for the nonholonomic robot are

$$\dot{x} = u_f \cos(\theta), \quad \dot{y} = u_f \sin(\theta), \quad \dot{\theta} = u_t. \tag{13}$$

where (x, y, θ) denote the position and orientation of the robot. The control inputs are

$$\begin{aligned} u_f &= \tfrac{1}{2}(u_L + \eta_L + u_R + \eta_R), \\ u_t &= \tfrac{1}{w}(u_L + \eta_L - u_R - \eta_R) \end{aligned} \tag{14}$$

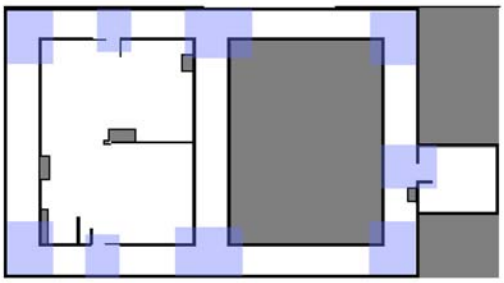

Fig. 1. Robot environment and landmarks

where u_L, u_R are the commanded left and right wheel velocities, w is the distance between the wheels, and η_L, η_R are independent random variables. We assume the control inputs are constant over Δt and use an exact discretization of (13).

The robot is equipped with three different MDLe plans for moving through the environment. The first is designed to move the robot around the landmarks in a counter-clockwise manner, the second to move it in a clockwise manner, and the third is the identity plan which leaves the robot in place (this would be used for example if the robot decides to make additional measurements as opposed to attempting to move to another landmark). To determine the corresponding Markov matrices, each plan was run at least 50 times from each landmark and the robot's position at the end of each run was recorded. As an example, the matrix for the first plan was

$$A(u_1) = \begin{bmatrix} 0.36 & 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0.63 \\ 0.56 & 0.39 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 \\ 0 & 0.83 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0.02 \\ 0 & 0 & 0.52 & 0.44 & 0.04 & 0 & 0 & 0 & 0 \\ 0.04 & 0 & 0 & 0 & 0.36 & 0.60 & 0 & 0 & 0 \\ 0.56 & 0 & 0 & 0 & 0 & 0.44 & 0 & 0 & 0 \\ 0.05 & 0 & 0 & 0 & 0 & 0.74 & 0.21 & 0 & 0 \\ 0.03 & 0 & 0.01 & 0 & 0 & 0 & 0.48 & 0.48 & 0 \\ 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0.68 & 0.31 \end{bmatrix},$$

From the matrix we see that from most landmarks, this plan does move the robot counter-clockwise around the landmarks. However, from landmark five the robot tends to move clockwise to landmark six and from landmark six it tends to move to landmark one. The times for the robot to complete the motion plans were also recorded.

In addition to the three motion plans, two observation plans were developed. The first moves the robot forward very briefly (while avoiding any intervening obstacles) and gathers only four sets of readings from the sensors. The plan was run with 300 particles on each landmark. As such it was intended to be a fast but less informative observation plan. The second moves the robot forward for a full second, gathering 20 sets of readings, and uses 300 particles on each landmark. Recall that the robot runs the observation plan only after completing a motion plan and we therefore have some information as to the possible starting position of the robot; consequently, the statistics of the robot's final positions (after having executed a motion plan) were used to generate the initial pdf for the particles in our localization algorithm. Each observation plan was run at least 25 times from each landmark to determine an estimate of the distribution on the observation. As an example, the matrix corresponding to the first (brief) plan was

$$O(1) = \begin{bmatrix} 0.78 & 0.05 & 0.15 & 0.02 \\ 0.19 & 0.62 & 0.12 & 0.07 \\ 0.40 & 0.07 & 0.43 & 0.10 \\ 0.82 & 0.04 & 0.07 & 0.07 \\ 0.02 & 0.14 & 0.05 & 0.79 \\ 0.99 & 0.01 & 0.00 & 0.00 \\ 0.36 & 0.10 & 0.37 & 0.17 \\ 0.16 & 0.57 & 0.14 & 0.13 \\ 0.82 & 0.04 & 0.10 & 0.04 \end{bmatrix}$$

TABLE I

SIMULATION: PLAN SEQUENCE

Step	Control plan	Observation plan
0	2	1
1	2	1
2	2	1
3	3	1
4	2	1
5	2	1

In the simulation presented below, we sought to ensure that the robot arrived at a desired landmark, while minimizing the amount of time it takes to get there. To achieve this, we defined the cost function (to be maximized):

$$J(P_{0|0}) = a_1 P_{N|N} d - \left(\sum_{k=0}^{N-1} a_2 P_{k|k} T'_{u_k} + a_3 T_{o_k} \right) \quad (15)$$

where d is a column vector with a 1 in the position corresponding to the desired landmark and zeros everywhere else, T_{u_k} is the row vector of expected times to run the control u_k given position of the robot, T_{o_k} is $0.2sec$ for the first observation plan and $1sec$ for the second, and the a_i are constant weights. By choosing a_1 much larger than the other weights we can ensure that the controller will only optimize for time over those control and observation sequences that have a high probability of arrival at the desired landmark. We thus set $a_1 = 1000$ and $a_2 = a_3 = 1$. The optimal control/observation plan sequence was then obtained by dynamic programming, with the number of stages in (15) set to $N = 4$. If at any time the probability of being on the desired landmark exceeded a threshold value (0.85), then the controller terminated. Otherwise, if at the end of 4 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 and the goal location was landmark 5. The initial conditional density was uniform. The trajectory and the evolution of the conditional probability are shown in Fig. 2 and 3 respectively while the optimal control and observation plans and the actual and observed landmark classes are shown in Tables I and II. In this case the robot took six pairs of control and observation plans to be certain it had arrived at the desired landmark.

V. CONCLUSIONS

We discussed the problem of landmark-based navigation and localization for mobile robots. Our approach relies

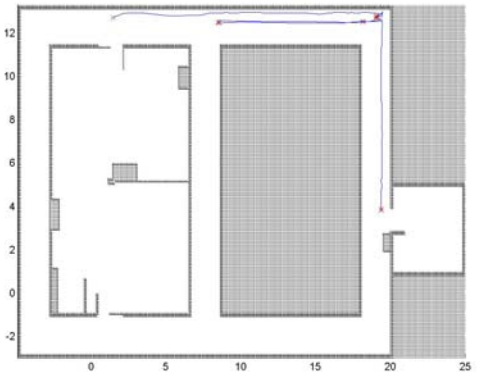

Fig. 2. Simulation: trajectory

TABLE II
SIMULATION: STATE AND OBSERVATIONS

Step	True landmark	True landmark class	Observed class
0	2	2	-
1	4	1	1
2	4	1	1
3	3	3	3
4	3	3	3
5	4	1	1
6	5	4	4

on the use of language-based control policies to connect interesting or relevant areas of an expansive environment, alternating with observation policies that aim to inform the robot's best guess as to which landmark it is currently on. The observation data are fed to a set of particle filters that numerically approximate the conditional probability of being on each landmark. The proposed approach fits naturally with the idea of using language-based 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.

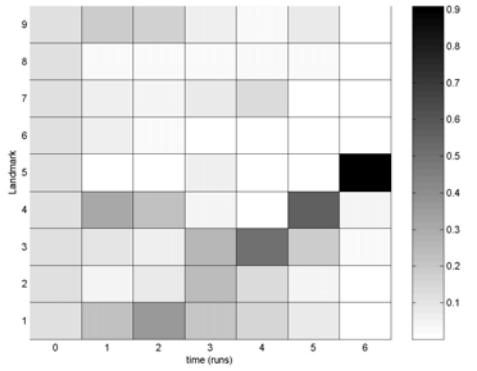

Fig. 3. Simulation 2: Condition probability evolution

ACKNOWLEDGMENTS

The authors gratefully acknowledge the help of Patrick Carlos, Aaron Greene, and Stephanie Tan in developing the simulator.

REFERENCES

[1] R.W. Brockett. On the computer control of movement. In *Proc. of the 1988 IEEE Conf. on Robotics and Automation*, pages 534–540, 1988.

[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-Varsakelis, P.S. Krishnaprasad, S.B. Andersson, F. Zhang, L. D'Anna, and P. Sodre. The MDLe engine: A software tool for hybrid motion control. Technical Report TR2000-54, The Institute for Systems Research, 2000.

[4] M. Egerstedt and D. Hristu-Varsakelis. Observability and policy optimization for mobile robots. In *Proc. of the 41st IEEE Conf. on Decision and Control*, pages 3596–3601, Dec. 2002.

[5] M. Egerstedt and R. Brockett. Feedback can reduce the specification complexity of motor programs. *IEEE Trans. Robotics and Automation*, 48(2):213–223, Feb. 2003.

[6] 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.

[7] S.B. Andersson and D. Hristu-Varsakelis. Stochastic language-based motion control. In *Proc. of the 42nd IEEE Conf. on Decision and Control*, pages 3313–8, Dec. 2003.

[8] E. Frazzoli. Maneuver-based motion planning and coordination for multiple unmanned aerial vehicles. In *Proc of the AIAA/IEEE Digital Avionics Systems Conference*, 2002.

[9] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[10] Y.K. Hwang and N. Ahuja. Gross motion planning - a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

[11] J.-P. Laumond, editor. *Robot Motion Planning and Control*, volume 229 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 1998.

[12] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2000.

[13] S. Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33(1):41–76, Oct. 1998.

[14] 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.

[15] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Int. Journal of Robotics Research*, 21(8):735–758, 2002.

[16] R. Madhavan and H.F. Durrant-Whyte. Natural landmark-based autonomous vehicle navigation. *Robotics and Autonomous Systems*, 46:79–95, 2004.

[17] 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.

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

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

[20] B. Azimi-Sadjadi. *Approximate Nonlinear Filtering with Applications to Navigation*. PhD thesis, University of Maryland, 2001.

[21] D. Crisan and A. Doucet. A survey of convergence results on particle filter methods for practitioners. *IEEE Trans. on Signal Processing*, 50(3):736–746, 2002.

[22] 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.

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