Language-based feedback control using Monte Carlo Sensing

submitted to ICRA 2005

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.**

I. Introduction

The design of feedback control laws that accomplish seemingly straightforward tasks, such as those involving navigation in everyday environments or manipulation of objects, 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 [6], [13], [20], 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 [1], [7], [8], [11], [12]. In the following, we discuss the use of language-based control in robotic navigation and localization and describe an approach for choosing optimal feedback control laws

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. [14], [16], [17]), while efficient global localization can be achieved using Monte Carlo methods [10]. These approaches become infeasible as the size of the environment becomes large compared to the robot's sensing range. At the same time however, 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 [12] and has been extended to handle sensor and actuator uncertainties [2]. Furthermore, it suggests favoring linguistic instructions (to 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. [4], [24], [25]) and for navigation (e.g. [15], [19], [23]). 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 and 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. The optimization results in optimal sequences of control plans and measurements (with respect to an appropriate objective function) to achieve a given goal.

The remainder of this paper is organized as follows. In Section II we describe a Markov-chain based approach to landmark-based navigation and localization and formulate the corresponding optimal control problem. The Monte Carlo-based approach to landmark observation is described in Section III. Section IV presents a simulation-based experiment that illustrates the effectiveness of our approach.

II. Stochastic language-based motion control

As described in [2], we let $\mathcal{L} = \{L_1, \dots, L_{n_L}\}$ 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 a variety of map-based path-planning and localization techniques. Our focus here will be on the global version of the problem instead.

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 $\mathcal{Z} = \{z_1, \dots, z_m\}$, $m \leq n$, to be the collection of possible *observation outcomes*. This set 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, thus we 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 and that policy o_i is executed.

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 $\mathcal{U} = \{A(1), \dots, \mathcal{A}(n_c)\}$ of control actions and $\mathcal{O} = \{O(1), \dots, O(n_o)\}$ 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, $k \in \{0, 1, \dots, N\}$. 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

$$I_k \triangleq (u_0, o_0, z_0, \dots, u_k, o_k, z_k) \tag{1}$$

and define the row vector of conditional probabilities

$$P_{k|k} \triangleq \begin{pmatrix} p^1_{k|k} & \dots & p^{n_L}_{k|k} \end{pmatrix} \tag{2}$$

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

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

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

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

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

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

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

$$g_u:\mathcal{P}\times\mathcal{U}\times\{0,\ldots,N-1\}\rightarrow\mathbb{R}, \quad (7)$$

$$g_o:\mathcal{P}\times\mathcal{O}\times\{0,\ldots,N-1\}\rightarrow\mathbb{R}. \quad (8)$$

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

$$\begin{aligned}\min\nolimits_\pi J_\pi(P_{0|0}) \quad = \quad & E_{z_k,k=0,\ldots,N-1}\big\{g(P_{N|N},N)\\ & +\textstyle\sum_{k=0}^{N-1}\big(g_u(P_{k|k},u_k,k)+g_o(P_{k|k},o_k,k)\big)\big\}\end{aligned} \quad (9)$$

which naturally fits into the framework of dynamic programming (DP) [5]. The resulting feedback controller, selected via DP, is a sequence of motion and observation plans which seek to maximize 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 it takes 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 [10], [18].

A. Particle filters

We outline here the basic particle filter algorithm for a stochastic system with discrete dynamics. See [3] and references therein for a more complete description. Consider the discrete-time stochastic dynamical system given by

$$\begin{aligned} x_{k+1} \quad &= \quad f_k(x_k,u_k)+G_k(x_k)w_k,\\ y_k \quad &= \quad h_k(x_k)+v_k\end{aligned} \quad (10)$$

where w_k,v_k are independent noise processes and the distribution of x_0 is assumed to be given and independent of w_k,v_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 theoretically given by

1. Initialization: $p_0(x_0|y_0)=p(x_0)$
2. Diffusion:
$$\begin{aligned}&p_{k+1|k}(x_{k+1}|\mathcal{D}_k,u_{k+1})\\ &\quad=\textstyle\int p(x_{k+1}|x_k,u_k))p_{k|k}(x_k|\mathcal{D}_k)dx_k\end{aligned}$$
3. Bayes' rule update:
$$\begin{aligned}&p_{k+1|k+1}(x_{k+1}|\mathcal{D}_{k+1})\\ &\quad=\alpha p(y_{k+1}|x_{k+1})p_{k+1|k}(\mathcal{D}_k,u_{k+1})\end{aligned}$$
where α is a normalizing constant.
4. $k\leftarrow k+1$. Go to Step 2.

The density $p(x_{k+1}|x_k,u_k)$ is called the *motion model* for the system 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; it is a probabilistic model of perception.

Under appropriate assumptions the resulting conditional density is exact but in general the steps describe 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 particles $x_0^1,\ldots,x_0^N$ according to $p_0(x)$.
2. Diffusion: Find $\hat{x}_{k+1}^1,\ldots,\hat{x}_{k+1}^N$ from $x_k^1,\ldots,x_k^N$ using the dynamic rule (10).
3. Form the empirical distribution:
$$p_{k+1|k}^N(x)=\frac{1}{N}\sum_{j=1}^{N}\delta_{\hat{x}_{k+1}^j}(x)$$
4. Use Bayes' rule:
$$p_{k+1|k+1}(x)=\frac{\alpha}{N}\sum_{j=1}^{N}\delta_{\hat{x}_{k+1}^j}(x)\Psi_{k+1}(x)$$

where α is again a normalizing constant.

5. Resample: Sample $x^1_{k+1}, \dots, x^N_{k+1}$ 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. It has been shown [22] that (under some assumptions) the approximate density of this algorithm approaches (in an appropriate sense) the true density as the number of particles goes to infinity.

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

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

provides an indication of the quality of the approximation by giving information 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 attached to the environment. 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. To use the particle filter algorithm to estimate this pdf one needs to specify the motion and sensor models that will govern the dynamics of the particles. The motion model is simply 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.

To implement the particle filter algorithm, one first samples N "particles" from an initial distribution (in the absence of any prior information this could be a uniform one.) Each particle evolves according to the stochastic equations of motion that apply to the robot, using a fixed time step. Then, an observation is made using the sensors of the robot, and the particles are weighted according to the sensor model. Finally, a new set of samples is drawn from the approximation to the pdf of the robot as given by the weighted particles.

C. *Particle filters for landmark sensing*

As mentioned above, 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.

Recall that at the completion of each motion plan, the robot is able to execute a sensing plan in which it moves locally while gathering data. If the robot were known to be on a given landmark, then we would have an instance of the standard localization problem, which can be solved using the particle filter algorithm. The particle filter 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 then 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, $C^i, i = 1, \dots, m$ can be calculated. An observation from the set of possible observations is then generated by sampling from the probability distribution over the m possible observations given by

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

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 and there is therefore a tradeoff between the number of particles used (and thus the computation time) and the amount of information generated. This tradeoff can be captured by assigning different costs to different choices for the number of particles used and optimizing these costs under the general framework of Section II.

The particle filter algorithm is an extremely powerful technique and it is applicable to arbitrary distributions. However, it is computationally intensive; while performing the calculations for a single particle is simple, a large number of particles is usually required to obtain a good estimate of the pdf. 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 some information is generated (i.e. 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 [9] 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 is to be run only while the observation plan is being executed and not while the motion plan is run. On small maps the particle filter algorithm 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. When the robot enters the actual landmark and begins the observation phase, it would then begin 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 [10], the additional complications can be avoided simply by running the particle filters only when observing the landmark.

Finally, we note also that once the robot has determined with high probability that it has arrived at the desired landmark, the final estimate of the distribution can be used as an initial density for a standard 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 Figure 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 [21] with cells of $10cm$ square. As examples, the hallway and T landmarks are shown in Figure 2. The landmarks are numbered from 1 to 9 beginning with the upper left corner and proceeding clockwise around the map.

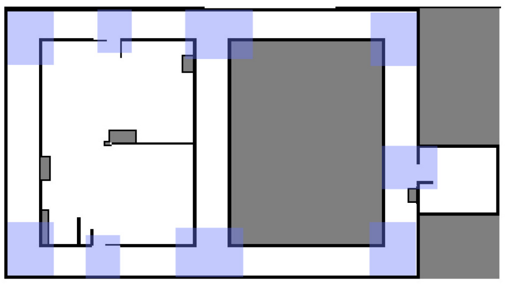

Fig. 1. Robot environment and landmarks

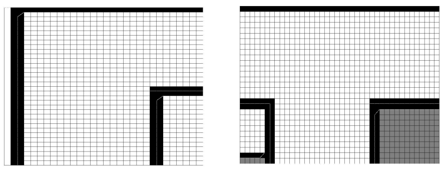

Fig. 2. Sample landmark maps of a "corner" and "T-intersection"

The equations of motion for the nonholonomic robot are

$$\begin{array}{rcl} \dot{x} & = & u_f \cos(\theta), \\ \dot{y} & = & u_f \sin(\theta), \\ \dot{\theta} & = & u_t. \end{array} \tag{13}$$

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

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

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. If we assume the control inputs are constant over the time step Δt then the dynamics can be solved exactly (see [26]) to yield the discrete time evolution equations for (x, y, θ).

$$\begin{array}{rcl} x_{k+1} & = & x_k + \frac{u_f}{u_t}\left(\sin(\theta_k + u_t \Delta t) - \sin(\theta_k)\right) \\ y_{k+1} & = & y_k + \frac{u_f}{u_t}\left(\cos(\theta_k) - \cos(\theta_k + u_t \Delta t)\right) \\ \theta_{k+1} & = & \theta_k + u_t \Delta t. \end{array} \tag{15}$$

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 applies a zero control, leaving 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. The resulting matrices for each plan were

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

$$A(u_2) = \begin{bmatrix} 0.14 & 0.74 & 0 & 0 & 0 & 0 & 0 & 0 & 0.12 \\ 0 & 0.14 & 0.74 & 0.12 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.28 & 0.64 & 0.08 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.04 & 0.16 & 0.80 & 0 & 0 & 0 & 0 \\ 0.02 & 0 & 0 & 0.86 & 0.12 & 0 & 0 & 0 & 0 \\ 0.06 & 0 & 0 & 0 & 0 & 0.26 & 0.68 & 0 & 0 \\ 0.10 & 0 & 0 & 0 & 0 & 0 & 0.10 & 0.64 & 0.16 \\ 0.16 & 0 & 0 & 0 & 0 & 0 & 0 & 0.22 & 0.62 \\ 0.80 & 0 & 0 & 0 & 0 & 0 & 0 & 0.02 & 0.18 \end{bmatrix},$$

and $A(u_3) = \mathbb{I}$ where $\mathbb{I}$ is the identity matrix. Here $[A(u_k)]_{ij}$ is the probability of ending at landmark j given that the robot starts at landmark i and executes motion plan k. The time for the robot to complete the motion plan was also recorded. The average times from each landmark were

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

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 is 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. The resulting matrices for the two observation plans are

$$O(1) = \begin{bmatrix} 0.78 & 0.05 & 0.15 & 0.02 \\ 0.19 & 0.62 & 0.12 & 0.07 \\ 0.40 & 0.07 & 0.43 & 0.10 \\ 0.82 & 0.04 & 0.07 & 0.07 \\ 0.02 & 0.14 & 0.05 & 0.79 \\ 0.99 & 0.01 & 0.00 & 0.00 \\ 0.36 & 0.10 & 0.37 & 0.17 \\ 0.16 & 0.57 & 0.14 & 0.13 \\ 0.82 & 0.04 & 0.10 & 0.04 \end{bmatrix} \quad O(2) = \begin{bmatrix} 0.97 & 0.01 & 0.01 & 0.01 \\ 0.03 & 0.65 & 0.20 & 0.12 \\ 0.13 & 0.21 & 0.63 & 0.03 \\ 0.81 & 0.09 & 0.04 & 0.06 \\ 0.00 & 0.29 & 0.03 & 0.68 \\ 0.90 & 0.07 & 0.03 & 0.00 \\ 0.10 & 0.25 & 0.59 & 0.06 \\ 0.02 & 0.64 & 0.12 & 0.22 \\ 0.45 & 0.27 & 0.21 & 0.07 \end{bmatrix}$$

In the simulations 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):

$$J(P_{0|0}) = a_1 P_{N|N} d - \left(\sum_{k=0}^{N-1} a_2 P_{k|k} T'_{u_k} + a_3 T_{o_k} \right) \tag{16}$$

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 Eq. 16 set to $N = 4$. If at any time the probability of being on the desired landmark exceeds a threshold value (0.85) then the controller terminates. Otherwise, if at the end of 4 steps the probability of being on the desired landmark is less than the threshold, then the controller was run again.

In the first simulation we placed the robot in a random starting position on landmark 2, the top door of the large room, and assumed the robot knew with certainty which landmark it was on. The robot was asked to go to landmark 5, outside the open office door on the right. The resulting trajectory of the robot is shown in Figure 3, 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 Figure 4. The optimal selection of control and observation plans is shown in Table I and the actual and observed landmark classes are shown in Table II. The robot successively navigated to the desired landmark after running only four control and observation plans but needed one additional observation before it was certain it had arrived on landmark 5. Notice that the controller always selected the brief observation plan; this is because the small improvement in accuracy when using the longer observation plan did not offset the cost of the longer time needed to run it. In other simulations (not reported here) where the weighting factor a_3 was set to zero the controller often choose the longer observation plan.

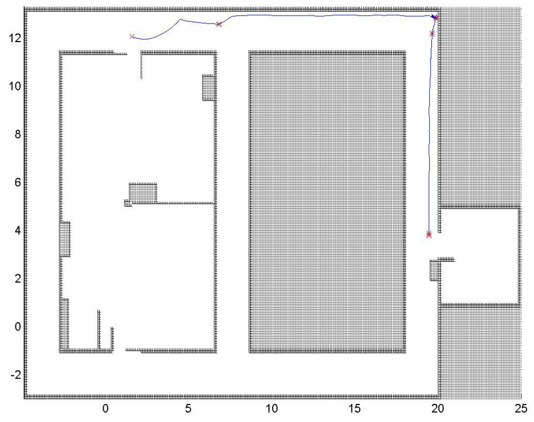

Fig. 3. Simulation 1: trajectory

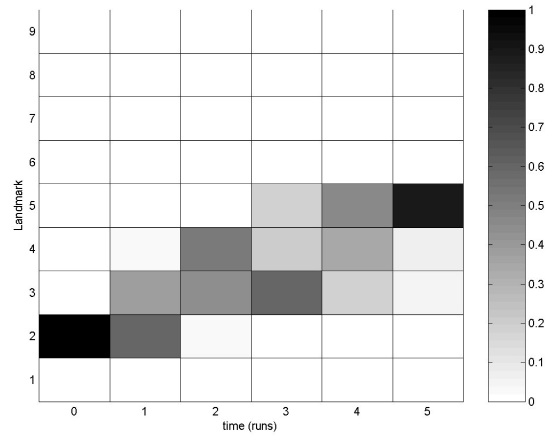

Fig. 4. Simulation 1: Condition probability evolution

In the second simulation we again placed the robot in a random starting position on landmark 2 and asked that it proceed to landmark 5. This time, however, the initial conditional density was uniform, that is the robot had no information as to its starting position. The trajectory and the evolution of the conditional probability are shown in Figures 5 and 6 respectively while the optimal control and observation plans and the actual and observed landmark classes are shown in Tables III and IV. In this case the robot took six pairs of control and observation plans to be certain it had arrived at the desired landmark.

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

TABLE I

SIMULATION 1: PLAN SEQUENCE

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

TABLE II

SIMULATION 1: STATE AND OBSERVATIONS

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

TABLE III

SIMULATION 2: PLAN SEQUENCE

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

TABLE IV

SIMULATION 2: STATE AND OBSERVATIONS

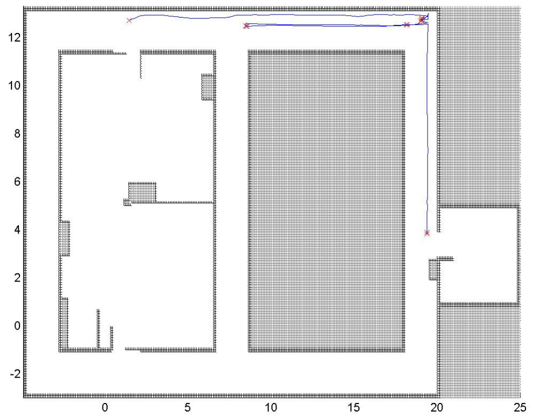

Fig. 5. Simulation 2: trajectory

V. CONCLUSIONS

We discussed the problem of landmark-based navigation and localization for mobile robots. Our approach relies on the use of language-based control policies that are used to connect the most 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 are implemented at the level of a motion description language, as opposed to that of sensors and actuators.

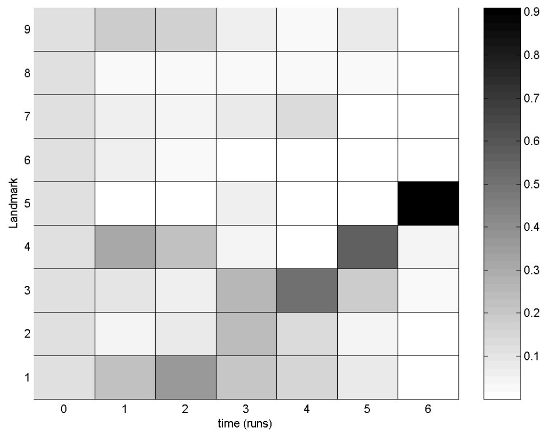

Fig. 6. Simulation 2: Condition probability evolution

VI. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the help of Aaron Greene and Stephanie Tan in developing the simulator. This work was supported by the National Science Foundation 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).

REFERENCES

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

[2] S.B. Andersson and D. Hristu-Varsakelis. Stochastic language-based motion control. In *Proceedings of the 43$^{\text{rd}}$ IEEE Conference on Decision and Control*, pages 1–6, 2003.

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

[4] 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.

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

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

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

[13] D. Hristu-Varsakelis, P.S. Krishnaprasad, S. 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.

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

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

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

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

[18] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation*, page ??, 2000.

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

[20] 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.

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

[22] P. Del Moral. Non linear filtering: Interacting particle solution. *Markov Processes and Related Fields*, 2(4):555–580, 1996.

[23] 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.

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

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

[26] D.P. Tsakiris. *Motion Control and Planning for Nonholonomic Kinematic Chains*. PhD thesis, University of Maryland, 1995.