Proceedings of the 41st IEEE
Conference on Decision and Control
Las Vegas, Nevada USA, December 2002

Observability and Policy Optimization for Mobile Robots

Magnus Egerstedt[1]
Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
magnus@ece.gatech.edu

Dimitrios Hristu-Varsakelis[2]
Mechanical Engineering and
Institue for Systems Research
University of Maryland
College Park, MD 20742
hristu@glue.umd.edu

Abstract

This work considers the use of abstract descriptions of motion control programs and of the environment, and explores some new problems of system theoretic interest that arise as a result. We study the problem of active localization for a mobile robot moving on a sparsely-described uncertain environment and show how that problem can be posed as that of observability of a finite automaton. We present algorithms (based on Hidden Markov Models) that answer the question of i) whether or not a representation of the environment (in the form of a directed graph) is observable, and ii) what is the shortest navigation policy that allows the robot to uniquely identify its location on the graph.

1 Introduction

One aspect of control systems science that sets it apart from other disciplines is its reliance on a sometimes delicate balance between problem formulations that are sufficiently narrowly defined to be solvable, yet general enough to be of practical significance. In robotics, one of the main challenges has to do with the complexity of the control programs that are needed to guide a robot through even a moderately complicated task. To avoid always having to design at the level of the actuators, it is natural to attempt to "tokenize" control inputs into primitives that can be hierarchically composed into control programs of higher complexity, similarly to the object-oriented programming paradigm. A formal framework for doing just that has been established in the work on *motion description languages* (MDLs) over the last decade [3, 13, 17]. Briefly, the idea behind MDLs is to control the evolution of a continuous-time system by symbolic commands that are eventually interpreted down to feedback laws. The lowest-level primitives (also known as "atoms") are ordered triples of the form (u, k, ξ), characterized by an open-loop control signal, u, a closed-loop mapping, k, that maps state and environment observations to a control action, as well as an interrupt function, ξ, that specifies the duration (temporally as well as spatially) of the atom. Atoms, together with simple rules of composition, can be used to write control "programs" that in principle could be machine-independent (see [3] for a full description of the syntax and definition of MDL; [12, 17] describe its "extended" counterpart, MDLe).

This paper continues the development of ideas related to language-based descriptions of control tasks and some of their implications for autonomous robot navigation. In particular, we are interested in an abstract description of the environment [13], in the same manner that MDL strings offer an abstract description of control laws. Although such descriptions are appealing for the reasons outlined above, they give rise to new control problems of higher complexity, because one now operates on a space of primitives (themselves composed of yet simpler primitives) as opposed to the spaces of input and output signals to and from the underlying dynamical system. It is exactly one such "meta"-problem that we would like to explore here, namely the problem of when (and by what actions) it is possible for a mobile robot to localize itself in its environment (represented as a "web" of landmarks together with instructions for getting from one to another). For other treatments of such "meta"-problems in robot control, see for example [9, 14, 19].

The paper is structured as follows: In Section 2 we formulate the main problem and show how it is related to the notion of observability. We then, in Section 3, reformulate the problem as a Hidden Markov Model problem; an algorithmic solution is presented in Section 4, along with a discussion of some natural extensions of this work, currently in progress.

[1]The work by Magnus Egerstedt was sponsored by the National Science Foundation Grant No. EHS NSF-01-161.

[2]The work by Dimitrios Hristu-Varsakelis was supported in part by AFOSR Grant No. F496200110415 and in part by the Minta Martin Fellowship

2 Motion Description Languages and Robot Navigation on Graphs

For the purposes of this work, we consider a mobile robot, navigating in a partially known environment, which need not be structured. The environment is represented not as a contiguous map, but rather as a series of landmarks, representing areas of the world that are deemed "interesting" or "relevant" (for work related to graph-based representations of the environment see [7, 16, 15]; for recent work combining MDLs and landmark-based navigation see [13]).

We assume that once the robot is within sensory range of a landmark it can drive towards or around that landmark using a feedback control strategy. Furthermore, the robot can move between the different regions of attraction around the landmarks using a fixed number of control strategies which are coded as "words" in a motion description language (e.g. [13]). These MDL programs serve as instructions for driving the robot between established landmarks, even though failure to reach a given landmark is still possible due to temporal or spatial environment uncertainty. We note that MDL programs are used to describe geographical relationships *not in terms of where a location is, but what one must do to get there.* This gives rise to a directed graph, whose vertices x_i, $i = 1, ..., n$ correspond to landmarks and whose edges e_{ij} correspond to the existence of a control program that drives the robot between the regions of attraction around x_i and x_j. The aim is to replace - when possible - the details of a map by a feedback program. We remark that transitions between landmark need not be deterministic, i.e. executing the instructions in e_{ij} from x_i may not always lead to x_j. This allows one to bring uncertainty into the fold, in the form of a set of probability-preserving maps from the set of probability density functions defined on $x_1, ..., x_n$, to itself. This idea will be explored in Section 5.

Under the scenario described above, the main question this paper is concerned with is that of *observability*: Given that the robot starts out at a particular landmark of type y, how should the robot move around in the environment in order to recover its current position (i.e. the current node in the reachability graph). This question is closely related to that of observability for finite automata, as defined in [5][1], and in this paper we derive an algorithm for determining whether or not a given automaton is observable, using the Viterbi algorithm (see for example [8]) for Hidden Markov Models (HMMs). Because the transitions between different nodes will have a probability of success associated with them, we will also produce an algorithm that generates the probability of recovering the state of the system.

[1]This concept of observability differs from that in [1], where observability is defined with respect to *any* input sequence.

2.1 Preliminaries

We will choose to describe the continuous-time dynamics of the robot by $\dot{z} = F(z, v)$, $w = H(z)$, where $z \in Z \subset \mathbb{R}^n$ is the state of the system, $w \in W \subset \mathbb{R}^p$ is the observation made by the robot, and $v \in V \subset \mathbb{R}^k$ is the input to the system. As already discussed in the introduction, we let the symbolic inputs be triples of the form (v, k, ξ), where $v \in V$ is the open-loop component, $k : W \to V$ is the feedback mapping, and $\xi : W \to \{0, 1\}$ is the interrupt. A string of such input-triples is operated on by the dynamic system as

$$\begin{aligned} \dot{z} &= F(z, k_1(w, v_1)); & t_0 &\le t < T_1 \\ &\vdots & &\vdots \\ \dot{z} &= F(z, k_q(w, v_q)); & T_{q-1} &\le t < T_q, \end{aligned}$$

where T_i denotes the time at which the interrupt ξ_i changes from 0 to 1.

Now, assume that the robot is moving around in a partially structured environment, populated by a fixed number of landmarks. Let X denote the set of these landmarks, and assume furthermore that that each landmark $x \in X$ is of a particular type $y \in Y$, where Y is the finite set of possible landmark types, such as "door", "stairs", "hall", etc.

We now note that if we use a particular input (v, k, ξ) for moving the robot between two landmarks, we can abstract away the continuous dynamics of the system, as long as we know that this is a successful control policy. (We will see later on that only a given probability of success is needed for this construction to be meaningful.) At this level of abstraction, the evolution of the robot can thus be defined by a finite automaton (X, Y, U, f, h), where U is the finite set of control actions, $f : X \times U \to X$, and $h : X \to Y$. The evolution of the automaton is given by $x_{k+1} = f(x_k, u_k)$, $y_k = h(x_k)$.

2.2 Observability and Reachability on Graphs

Given that the robot starts out at a landmark of type y_0, how should it move around in the environment in order to figure out where it is? In other words, we want to construct an optimal policy $\pi_{y_0} : Y \to U$ in such a way that we know exactly where we are after a minimum number of steps, given that the robot starts at any $x_0 \in h^{-1}(y_0)$.

This problem, is related to the concept of observability for finite state machines, as defined in [5]: Consider the finite automaton (X, Y, U, f, h). We let the *output sequence map* $\mathcal{O} : \mathbb{Z}^+ \times X \times U^Y \to Y^\star$ be given as

$$\mathcal{O}(q, x, \pi) = h(x_1) \cdot h(x_2) \cdots h(x_q),$$

where $\pi : Y \to U$, and $x_1 = x$, $x_2 = f(x_1, \pi(h(x_1))), \ldots, x_q = f(x_{q-1}, \pi(h(x_{q-1})))$. Note that in the output sequence map $y_1 \cdot y_2$ denotes the concatenation of the letters y_1 and y_2 from the finite alphabet Y, and $\mathcal{O}(q, x, \pi) \in Y^q \subset Y^\star$, where Y^p is the set of words of length p over Y, and $Y^\star$ is the free monoid over Y.

What we want to do is to establish conditions for when it is possible to reconstruct the state of the system, and we note that this is possible if there exists q, π, such that $\mathcal{O}(q, x, \pi)$ is injective in its second argument. It should be noted that this is in fact only a sufficient condition since injectivity of the output-sequence map implies that we can reconstruct not only the current state but also *all* previous states, including the initial state as long as we know the system dynamics. However, to be able to reconstruct the current state does not imply that we can recover the previous states due to the lack of backward uniqueness for discrete event systems [4].

Definition 2.1 *(Observability) A finite automaton (X, Y, U, f, h) is observable if there exist a positive integer q_{obs} and an output-to-input mapping $\pi_{obs} : Y \to U$ that satisfies $\mathcal{O}(q_{obs}, x_1, \pi_{obs}) \neq \mathcal{O}(q_{obs}, x_2, \pi_{obs})$ for all $x_1, x_2 \in X,\ x_1 \neq x_2$.*

It is clear that this definition does not provide us with a constructive method for determining if a given finite automaton is in fact observable, and in the remainder of this paper we investigate the following two questions:

Problem 2.1 (Observability) *Given a finite automaton, determine if it is observable?*

Problem 2.2 (Recoverability) *Given an observable finite automaton, and an initial state $x_0 \in h^{-1}(y_0)$, find the optimal policy π_{y_0} that minimizes the number of steps required in order to specify the current state of the system exactly.*

We note already at this point that if we have a constructive method for finding the smallest q, and a corresponding π that renders the output sequence map injective, then we also have found an upper bound on the minimum number of steps we need to take from *any state* in order to recover the state of the system.

3 Hidden Markov Models

A natural way of thinking about the state-recoverability questions of interest is in terms of beliefs, i.e. how likely is it that we are at a particular state, given a string of observed landmarks. We will thus be working with beliefs, distributed over the different states, which makes it very natural to cast the problem as a Hidden Markov Model problem.

Let the probability that we are at a particular state x_i at time k be given by

$$p_i(k) = P(x(k) = x_i),$$

and form the distribution vector $p(k) = (p_1(k), \ldots, p_n(k))^T$, where $card(X) = n$. If we commit ourselves to using a given policy $\pi : Y \to U$, the evolution of these probabilities is given by $p(k+1) = A_\pi p(k)$, where $A_\pi = [a_{ij,\pi}]_{ij}$ is the transition matrix associated with policy π formed by

$$a_{ij,\pi} = P(x(k+1) = x_i \mid x(k) = x_j, u_k = \pi(h(x_k))).$$

If the system is at state x, we furthermore make the observation $y = h(x)$, i.e. if we set

$$b_{ij} = P(y = y_i \mid x = x_j),$$

we can form the observation matrix $B = [b_{ij}]$, and get the linear observation relation $q(k) = Bp(k)$, where

$$q_i(k) = P(y(k) = y_i),\ i = 1, \ldots, m,$$

with $card(Y) = m$.

Even though our transitions and observations are deterministic (i.e. B only contains zeros and ones), our finite automaton can be cast as a Hidden Markov Model under a fixed policy, which allows us to draw on the rich literature on HMMs. If we assume that we traverse the state space of the automaton, using the policy π, then during q steps we observe the landmark types $y(1), y(2), \ldots, y(q)$. If we form the single observation matrix $B(y_k) = \text{diag}(b_{k,1}, b_{k,2}, \ldots, b_{k,n})$, associated with the observation y_k, and let $\eta = (1/n, \ldots, 1/n)^T$, then the forward sweep algorithm (see for example [8]) tells us that the conditional density vector $\tilde{p}_{FW}^{\pi}(y(1), y(2), \ldots, y(q))$, defined as

$$\begin{pmatrix} P(x(q) = x_1 \mid y(1), \ldots, y(q), \eta) \\ \vdots \\ P(x(q) = x_n \mid y(1), \ldots, y(q), \eta) \end{pmatrix}$$

is given by

$$\begin{aligned} &\tilde{p}_{FW}^{\pi}(y(1), y(2), \ldots, y(q)) = \\ &= \tfrac{1}{N_q} B(y(q)) A_\pi B(y(q-1)) A_\pi \cdots A_\pi B(y(1)) \eta, \end{aligned}$$

where N_q is a normalization factor.

The problem concerning the reconstruction of the current state, given a particular policy π, is thus equivalent to that of determining if $\tilde{p}_{FW}^{\pi}$ is a unit vector after a given number of steps, i.e. if $\|\tilde{p}_{FW}^{\pi}\|_\infty = 1$. That this is the case follows directly from the fact that only then do we know, for certain, exactly what the current state of the system is. The problem of observability (recovering all the previous states of the system), can also be cast on a similar form using the *Viterbi algorithm* for HMMs:

$\eta = \frac{1}{n}(1,1,\ldots,1)^T$
$\tilde{p}^{\pi}_{FW,i}(k) = P(x(k) = x_i \mid y(1), y(2), \ldots, y(k))$
$\tilde{p}^{\pi}_{BW,i}(k) = P(x(k) = x_i \mid y(k+1), y(k+2), \ldots, y(m))$
$\tilde{p}^{\pi}_{FW}(k) = \frac{1}{N_k}B(y(k))A\cdots AB(y(1))\eta$
$\tilde{p}^{\pi}_{BW}(k) = \frac{1}{M_k}\tilde{A}B(y(k+1))\tilde{A}\cdots\tilde{A}B(y(m))\eta$
$\tilde{A} = A^T D$, where D is a diagonal normalization matrix
$\tilde{p}^{\pi}(k) = \frac{1}{\gamma_k}\tilde{p}^{\pi}_{FW} \otimes p^{\pi}_{BW}$,

where $\otimes$ denotes component-wise product.

Our findings thus far can be summarized as follows:

- Observability after m steps $\Leftrightarrow$ $\|\tilde{p}^{\pi}(k)\|_{\infty} = 1, k = 1, \ldots, m$
- Recoverability after m steps $\Leftrightarrow$ $\|\tilde{p}^{\pi}_{FW}(m)\|_{\infty} = 1$

4 Deciding Observability and Related Control Policies

We can thus answer the questions posed in Problems 2.1 and 2.2 by computing the conditional densities associated with every policy starting from every state (landmark). One issue that needs to be resolved has to do with when to terminate the iterations. As a consequence of the Pidgeon Hole Principle (see for example [10] for an accessible treatment of this classic result) we can obtain such a bound, and we state this fact as a lemma:

Lemma 4.1 (Iteration Bounds) *If the automaton $\mathcal{A}$ is observable then it is observable in $card(X)$ steps, i.e. if $\mathcal{O}(q, \pi, \cdot)$ is injective for some (π, q) then it is injective for $(\pi, card(X))$ as well.*

In other words, observability can be determined by deciding if the following optimization problem has a solution:

$$\min_{\pi \in U^Y, q \in \{0,1,\ldots,n\}} \{q\}$$

subject to the injectivity constraint

$$\mathcal{O}(q, x_1, \pi) \neq \mathcal{O}(q, x_2, \pi),\ \forall x_1 \neq x_2 \in X.$$

The constraint can furthermore be reformulated as

$$\|\tilde{p}^{\pi}(h(\chi(0,x,\pi)), h(\chi(1,x,\pi)), \ldots, h(\chi(q,x,\phi)))\|_{\infty} = 1,$$

which has to hold $\forall x \in X$, where we use $\chi(k, x, \pi)$ as shorthand for

$\chi(0, x, \pi) = x$
$\chi(k, x, \pi) = f(\chi(k-1, x, \pi), \pi(h(\chi(k-1, x, \pi)))$
$k = 1, 2, \ldots$

A straight forward way to solve the last problem is to iterate over all policies, and then compute the conditional densities from each initial state over a maximum of $card(X)$ steps. Since we are multiplying $card(X) \times card(X)$ matrices (complexity $O(card(X)^2)$) in order to get $\tilde{p}^{\pi}$, the complexity of this algorithm is

$$O\left(card(U)^{card(Y)} card(X)^4\right).$$

4.1 Single State Recovery

Another relevant question we would like to answer in connection to the mobile robot application we have in mind is: given a particular $x \in h^{-1}(y)$, how many steps do we need to take in order to determine the current position? A very straight forward modification of the previous algorithm directly gives us an answer to the current state-recoverability question, and we can once again use $card(X)$ as an upper bound on the number of steps in our algorithm since if we can bound the number of steps needed to recover all states, then that bound is certainly enough when we try to recover only the current state of the system. Since the previous algorithm answers questions concerning *all* states, it simply should be modified slightly by deciding if the following optimization problem has a solution:

$$\min_{\pi \in U^Y, q \in \{0,1,\ldots,n\}} \{q\}$$

subject to the conditional density constraint

$$\|\tilde{p}^{\pi}_{FW}(h(\chi(0,x,\pi)), h(\chi(1,x,\pi)), \ldots, h(\chi(q,x,\phi)))\|_{\infty} = 1,$$

which can be solved using

$$O\left(card(U)^{card(Y)} card(X)^3\right)$$

operations.

4.2 Example

We illustrate the use of our algorithm by applying it to the finite automaton in Figure 1.

In this example, the number of policies are $card(U)^{card(Y)} = 2^3 = 8$ and if we let $\pi_1(y) = u_1$, $\forall y \in Y$, we see that we need to take three steps in order to uniquely determine the state of the system. For example, if we start at x_1 we go through y_1, y_2, y_1, y_1 in three steps, versus y_1, y_2, y_1, y_2 if we start at x_2. In the first of these cases, the conditional density vectors evolves according to

$$\begin{aligned}
\tilde{p}(y_1) &= (1/3, 1/3, 0, 1/3, 0, 0)^T \\
\tilde{p}(y_1, y_2) &= (0, 0, 1/2, 0, 1/2, 0)^T \\
\tilde{p}(y_1, y_2, y_1) &= (1/2, 0, 0, 1/2, 0, 0)^T \\
\tilde{p}(y_1, y_2, y_1, y_1) &= (0, 0, 0, 1, 0, 0)^T,
\end{aligned}$$

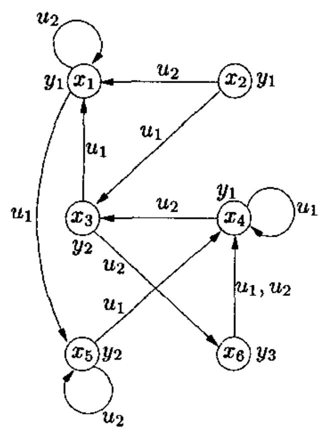

Figure 1: The finite automaton under investigation in Section 4.2.

while the second case gives

$$\begin{aligned}\tilde{p}(y_1) &= (1/3,1/3,0,1/3,0,0)^T\\ \tilde{p}(y_1,y_2) &= (0,0,1/2,0,1/2,0)^T\\ \tilde{p}(y_1,y_2,y_1) &= (1/2,0,0,1/2,0,0)^T\\ \tilde{p}(y_1,y_2,y_1,y_2) &= (0,0,0,0,1,0)^T.\end{aligned}$$

If we iterate the forward-sweep algorithm over all policies, we find that the minimum number of steps required to determine the current state of the system is one, and that the optimal policy is either

$$\pi(y) = \begin{cases} u_2 & \text{if } y = y_1\\ u_2 & \text{if } y = y_2\\ u_1 & \text{if } y = y_3\end{cases}$$

or

$$\pi(y) = \begin{cases} u_2 & \text{if } y = y_1\\ u_2 & \text{if } y = y_2\\ u_2 & \text{if } y = y_3\end{cases}$$

5 Extensions

In this section we outline some of the possible routes one might take when extending the proposed HMM-methodology to answer further "meta"-questions when controlling mobile robots, and in particular we will focus on the situation when the transitions and observations are stochastic rather than deterministic.

5.1 Probabilistic Transitions

In our current formulation, the edges of the graph/map correspond to deterministic transitions from one landmark to another, meaning that if the robot executes u_1 from x_2 (Figure 1) it will go to x_3 for sure. In reality however, a set of instructions that should have led the robot between two locations may fail, because of uncertainty in the environment or sensors (in the case of indoor navigation this could be due to obstacles or traffic in a hallway, misidentified landmarks or hallway intersections, etc). Most control engineers are well aware that uncertainty, combined with the complexity of a task make it difficult to design control algorithms that address every contingency. If one adopts that point of view, the probability assigned to a node changes not only because of incoming observations but also because the last input might have taken the robot to the wrong place. It is therefore natural to consider a probability $p(x_i, x_j, u_k)$ for each transition, and the associated map that propagates a probability distribution over a graph under the action of a control policy.

The use of HMMs for capturing probabilistic transitions is even intuitive than when letting the system dynamics be governed by deterministic transitions. In the probabilistic case we simply let the fixed-policy transition matrix A_π be such that

$$\sum_{i=1}^{n} a_{ij,\pi} = 1,\ a_{ij,\pi} \geq 0,$$

instead of, as before, $a_{ij,\pi} \in \{0,1\}$. We have thus associated a probability with the transitions and we can of course modify the observation matrix B in a similar manner by associating a probability with a certain observation at a certain state.

If we now want to solve the state-recoverability problem, we can no longer apply the algorithm derived from the Viterbi algorithm directly since the iteration bounds from Lemma 4.1 no longer hold. In fact, in a probabilistic setting we might strengthen our beliefs about the current state by further explorations, even after having taken $card(X)$ steps. However, we are primarily interested in finding the probability of being at a particular state; this can be done simply by computing the forward, conditional density $\tilde{p}^{\pi}_{FW}(m)$.

The notion of observability should now be replaced by a measure of certainty about the current state after a given number of steps. This can be done in a very straightforward manner using the HMM formalism.

5.2 Further Issues

If the graph associated with the environment is not observable, one may look for ways to remedy the situation, by refining the graph so that it becomes observable. A related question could be: "what is the minimum number of landmarks that I must add to the graph to make it observable?" This problem is nontrivial because the utility of a new landmark depends not only on its presence or absence but also on its label (its sensor signature). The latter will not always be selectable and will vary with geography.

A related problem has to do with enhancing the graph with new landmarks for the purposes of increasing the probability of success when traveling between distant vertices. Aside from deciding which portion of the graph should be refined (replacing an edge by an edge-vertex-edge), one must somehow obtain the transition probabilities $p(x_i, x_j, u_k)$. A combination of simulation and exploration might be appropriate for that purpose.

6 Conclusions

Motivated by ongoing work on motion description languages and landmark-based descriptions of the environment, we have addressed two basic questions related to the observability of an environment description. Our approach begins by describing the environment as a directed graph whose vertices are identified with landmarks and whose edges represent known feedback control instructions (MDL strings) whose execution leads a robot from one landmark to another. We showed how navigation on such graphs is related to the notion of observability for finite automata; we presented an algorithm that decides the observability of a graph associated with our knowledge of the environment and showed how to find the shortest sequence of transitions that will allow a robot to uniquely identify its position on the graph. Our formulation, based on Hidden Markov Models, lends itself well to incorporating environment uncertainty; in addition to being helpful for localization, our results can be used to decide if a robot has sufficient detail for navigating its environment, or if further exploration is needed around certain neighborhoods.

References

[1] A. Balluchi, L. Benvenuti, A.L. Sangiovanni-Vincentelli. Observers for Hybrid Systems with Continuous State Resets. In *Proceedings of the IEEE Mediterranean Conference on Control*, Lisbon, Portugal, 2002.

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

[3] R.W. Brockett. Hybrid Models for Motion Control Systems. In *Perspectives in Control*, H. Trentelman and J.C. Willems, eds., pp. 29-54, Birkhauser, Boston, 1993.

[4] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Boston, MA, 1999.

[5] M. Egerstedt and R.W. Brockett. Feedback Can Reduce the Specification Complexity of Motor Programs. In *Proc. IEEE Conference on Decision and Control*, Orlando, FL, Dec. 2001.

[6] M. Egerstedt. Some Complexity Aspects of the Control of Mobile Robots. To appear in the *American Control Conference*, Anchorage, Alaska, May, 2002.

[7] S. Fabrizi and T. Saffioti. Extracting Topology-Based Maps from Gridmaps. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2972–2978, 2000.

[8] G. D. Forney. The Viterbi Algorithm. In *Proceedings of the IEEE*, pages 268–278, 1973.

[9] D. Fox, W. Burgard and S. Thrun. Active Markov Localization for Mobile Robots. *Robotics and Autonomous Systems*, vol. 25, pages 195–207, 1998.

[10] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison Wesley, Reading, MA, 2000.

[11] D. Hristu-Varsakelis. Robot formations: Learning minimum-length paths on uneven terrain. In *Proc. IEEE Mediterranean Conf. on Control and Automation*, 2000.

[12] D. Hristu-Varsakelis, L. D'Anna, P. S. Carlos, F. Zhang, S. Andersson and P. S. Krishnaprasad. The MDLe Engine: A software tool for hybrid motion control. Tech. Report 2000-54, Institute for Systems Research.

[13] D. Hristu-Varsakelis and S. Andersson. Directed Graphs and Motion Description Languages for Robot Navigation and Control. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2689–94, 2002.

[14] J.P. Kalebling, A. R. Cassandra and J. A. Kurien. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[15] A. Lambert and Th. Fraichard. Landmark-based safe path planning for car-like robots. In *Proc IEEE Int. Conf. on Robotics and Automation*, pp. 2046–2051, 2000.

[16] A. Lazanas and J.-C. Latombe. Landmark-based robot navigation. *Algorithmica*, 13(5):472–501, May 1995.

[17] V. Manikonda, P.S. Krishnaprasad, and J. Hendler. Languages, Behaviors, Hybrid Architectures and Motion Control. In *Mathematical Control Theory*, Eds. Willems and Baillieul, pp. 199–226, Springer-Verlag, 1998.

[18] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1060–1065, 1998.

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