Simultaneous Identification and Control of a Hybrid Dynamic Model for a Mobile Robot

David J. Austin
d.austin@computer.org

Center for Autonomous Systems,
Royal Institute of Technology,
SE 100-44 Stockholm, Sweden.

Abstract

A mobile robot moving in an office environment is a hybrid dynamic system. As the robot becomes constrained by obstacles in the environment, the dynamics of the system change. Thus, the robot has a number of distinct states with differing dynamics, corresponding to the different obstacles in the environment. Discrete switches between the states occur when the motion of the robot becomes constrained by a new obstacle or the robot moves away from a previously constraining obstacle. This paper presents a new method for the identification of a model of this hybrid dynamic system. Simulated results are given demonstrating the applicability of this method to identification of hybrid dynamic systems. A simple controller is implemented indicating the possibilities for simultaneous identification and control and demonstrating the ease of controller design in a hybrid dynamic framework.

1 Introduction

Mobile robotics has long been an area of interest with the goal of expanding the robotics industry from the limited manipulators that are presently used in factories. Unfortunately, the real world presents many new challenges due to the dynamic and uncontrolled environment. Two central problems are of interest in mobile robotics: the development of a model of the environment and the navigation between goals using the environmental model. Both of these problems can be effectively addressed using a hybrid dynamic system framework [2]. This paper introduces the problem of automatic identification of a hybrid dynamic model of the system.

Many researchers have developed methods for learning metric maps of the robot's environment, which is equivalent to the identification of the continuous dynamics of a corresponding hybrid dynamic model [2]. Most recent map building has used a discretisation of the environment where the probability of occupancy of each grid cell is estimated. This area was established by Elfes [7] and enhanced by others since then [17, 16, 18, 5]. However, a grid-based map is a brute-force approach with considerable memory requirements and offers no additional higher level information. Abstract, feature-based maps offer a more tractable representation. For example, Leonard *et al.* [12] proposed a method for extracting line segment and corner features using sonar measurements and Arras and Siegwart [1] developed a method for extracting lines from laser range scanner data.

Another approach to map making is to build a topological map which completely abstracts the environment into a graph with a number of places connected by edges if there is a path between them [11, 14]. This type of map greatly simplifies the navigation problem by neglecting the details of traveling between the places. This approach is similar to just determining the discrete event model of an equivalent hybrid dynamic system. The approach presented here differs from previous work in mapping for mobile robots in that a hybrid dynamic systems framework is used to unify the identification of a metric map (the continuous part of the hybrid dynamic model) and a topological map (the discrete part). Previously, Thrun and Bucken [16] considered integration of grid-based and topological maps but their approach requires the identification of a grid-based map first and then generates a topological map from it, further aggravating the computational expense of grid-based maps.

Hybrid dynamic modeling is an ideal tool for mobile robotics as the gain and loss of constraints due to obstacles are fundamentally discrete events which cause abrupt changes in the permissible dynamics of the robot. Discrete event methods have been considered for mobile robots previously by Kosecka *et al.* [9, 10, 8] who defined the control commands as discrete states (e.g. `moving, steer_away, path_following`). However, these models are rather arbitrary without a mathematical basis for the state definitions. Instead, we propose a constraint-based method for defining the discrete states. As the mobile robot moves, it becomes

constrained by the obstacles within the environment. The gain and loss of constraint are modeled as discrete events. Navigation is achieved by ensuring that a desired sequence of discrete events occurs.

This paper will consider the identification of a hybrid dynamic model for a mobile robot and its environment. The next section presents the hybrid dynamic model that we will use and defines the class of hybrid dynamic systems for which the methods presented here are applicable. Section 3 presents techniques for identification of the continuous dynamics for each discrete state and Section 4 shows how a discrete event model can be built as the continuous dynamics are identified. Finally, Section 5 presents the controller which can use a partial hybrid dynamic model to navigate to target points. Finally, simulated results which demonstrate the identification process and the operation of the controller are presented in Section 6.

2 Hybrid Dynamic Model

Hybrid dynamic modeling is an ideal tool for mobile robotics as the gain and loss of constraints due to obstacles occur as discrete events which abruptly change the system dynamics. A further advantage of a hybrid dynamic model is that it can be readily extended to deal with dynamic obstacles, such as people or other robots. For this paper, we will consider only static obstacles that might occur in an office environment. More complex obstacles and moving obstructions can also be modeled and will be considered in future works.

The hybrid dynamic model that we will use consists of a 5-tuple $\mathcal{M} = (\mathcal{Q}, \mathbf{x}, \mathbf{f}, \Sigma, \Phi)$ where

- $\mathcal{Q}$ is a finite set of discrete states,
- $\mathbf{x} \in \mathbb{R}^2$ is the continuous state space of the robot. For a circular mobile robot moving in the plane this consists of x and y positions.
- $\mathbf{f} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ models the free space dynamics of the robot,
- Σ is a finite set of discrete events, $\Sigma \subset \mathcal{Q} \times \mathbb{R}^2 \rightarrow \mathcal{Q}$,
- $\Phi = \{\mathbf{G}_q : q \in \mathcal{Q}\}$ is a set of sub-manifolds of $\mathbb{R}^2$ to which the system is constrained for each discrete state $q \neq q_0$, where q_0 is the unconstrained state. The manifolds $\mathbf{G}_q$ are assumed to differentiable and connected.

This is sub-class of hybrid dynamic systems tailored for the consideration of mobile robots which become constrained by obstacles in their environment. Note that switching, or the occurrence of discrete events as obstacles are encountered, does not move the robot physically. Hence, the events merely change the discrete state of the system, as defined for Σ above. When the robot is in discrete state q the constraint imposed by that state is written as:

$$\mathbf{G}_q(\mathbf{x}) = \mathbf{0} \tag{1}$$

Note that it is possible to define additional discrete states in the system for other purposes but here we consider only discrete states associated with each manifold as well as the unconstrained state q_0.

Given the above definition for the hybrid dynamic model, the problem is to determine: (i) the set of manifolds $\mathbf{G}_q$ and the set of discrete states q, and (ii) the set of discrete events Σ.

3 Continuous State Identification

There are many methods for system identification for continuous dynamics [13]. Traditionally, some experiments are run, data collected and then analysed off-line. However, this approach is not well suited to a robotic application which should have the best model available at all times. Therefore, we will use a recursive approach to identification with the best model determined by integration of new sensor data into the prior model.

The identification process starts with the robot moving around the environment and becoming constrained by obstacles. The points at which the robot has observed a constraint are called the *constrained points*. We keep a record of the constrained points and denote them $\mathbf{x}_i$, $i = 1 \ldots n$, where n is the number of constrained points observed so far. We will denote the model after n constrained points have been observed as $\mathcal{M}(n)$.

3.1 Assumptions

Firstly, we assume that the robot has accurate knowledge of its position in the fixed reference frame. This may not be the case for some mobile robots but either a triangulation system or one of many self-localisation algorithms could be employed to ensure this.

In order to identify the dynamics of the constrained states $\mathbf{G}_q$, we must be able to detect when the robot is constrained. We assume that the robot is equipped with a sensor or sensors which tell us when the robot is constrained. Almost all mobile robots are equipped with bumpers or sonar sensors which can provide this information.

3.2 Primitives

We model each of constraint manifolds as a *primitive* so that Φ contains m primitive objects $\{\mathbf{G}_q : q = 1 \ldots m\}$. Note that $m \leq n$. The primitives are geometric objects which approximate the constraint manifolds for each discrete state, i.e. each primitive approximates a subset of the constrained points observed so far. Each primitive has various properties, as appropriate to represent the shape in question. In addition, each primitive has an error function which quantifies the quality of fit of

the primitive to the set of constrained points which is being approximated.

We will assume that the environment consists largely of linear segments as found in most office environments. Therefore, the obvious primitive to consider is a line segment, which is used to model the office walls. In previous work [4], both linear and curved primitives were used. However, for this paper, we will use only linear primitives for simplicity.

3.3 Model Update Process

After the observation of the nth constrained point, we must determine a new model $\mathcal{M}(n)$ which encompasses the new data. Identification of a model of a non-trivial system from sensor data requires making of many decisions. In the presence of sensor noise, it is inevitable that some of these decisions will be incorrect. Therefore, a process which is able to reverse prior decisions is required. This is known as non-monotonic reasoning in the artificial intelligence community [6]. Here we cast the process of hybrid dynamic model identification as a minimisation problem, automatically gaining the ability to reverse prior decisions. Casting the identification process as an optimisation problem gives non-monotonic reasoning because the optimal map is always sought, even if that requires changing prior decisions. The optimisation also allows us to take a rigorous, objective approach and also permits rapid experimentation or tuning of the parameters of the identification process.

Clearly we would like to estimate a model which maximises the quality. However, we have no knowledge of the correct model and so cannot directly determine the estimated model quality. Instead, we use the error function of each primitive to tell us how well that primitive approximates its set of constrained points. The error function is defined to be non-negative and we wish to minimise the error. Therefore, to find the model $\mathcal{M}(n)$, we wish to minimise the following error function:

$$f_{err}(\mathcal{M}(n)) = \sum_{q=1}^{m} err_q() + \alpha\, m \tag{2}$$

where err_q is the error function for the qth primitive $\mathbf{G}_q$, α is a positive constant, and m is the number of primitives in the model. For a given primitive, the error function err_q is the sum of the distances from the constrained points to the primitive (i.e. the error function is a distance measure). The second term of equation (2) is the term which attempts to choose a simpler model. Selection of higher values of α may cause excessive approximation of the constraint manifolds. On the other hand, selection of smaller values for α results in larger numbers of constraints. It was found that appropriate values of α lie in the range:

$$0.1 \times \epsilon \leq \alpha \leq \epsilon \tag{3}$$

where ϵ is the measurement error for the constrained points. Due to the complexity of the minimisation problem of equation (2), we cannot try any brute-force approaches to the minimisation. Instead, we restrict ourselves to a small set of possible minimisation steps, called *operations*.

3.4 Operations

The incremental updates of the model are defined by a set of *operations*. Each operation is a function which, given the existing model $\mathcal{M}(n-1)$ and the latest observation $\mathbf{x}_n$, determines a new model.

$$\mathcal{M}_k(n) = oper_k(\mathcal{M}(n-1), \mathbf{x}_n) \tag{4}$$

where $oper_k$ is the kth operation. Once we have the nth observation, each of the available operations is tried and the model which minimises the error of equation (2) is selected:

$$\mathcal{M}(n) = \arg\min_k(f_{err}(\mathcal{M}_k(n))) \tag{5}$$

There is a wide range of possible operations that one could consider for line constraints and, clearly, the selection of an appropriate set of operations is crucial to the successful and rapid identification of the model. Operation selection has been studied in a prior work [3] and the set of operations use here is based on these results.

The operations used for this paper are given in Table 1. The intuitive motivation for this set of operations is as follows. Clearly, the first operation that we require is one which adds a new constraint to the model, using the new constrained point (`AddLine`). Next, we require operations which adds the new constrained point to an existing constraint (`JoinSecondClosest` and `JoinThirdClosest`). These three operations will sometimes create multiple model constraints for a single actual constraint. Therefore, we require operations which merge model constraints together (`MergeTwo` and `MergeThree`). All of the proposed operations so far are additive: there are no operations to split constraints. There may be situations where constraint splitting is desirable and so the final operation that we will consider is a constraint split (`JoinAndSplit`).

Splitting of constraints is necessary because it allows us to recover from erroneous joining of constraints. It is a part of the non-monotonic reasoning capability of this framework. False joining of constraints is certain to happen occasionally because of noisy sensor data and because joining of constraints is encouraged by the constant α in equation (2). For any positive value of α there will be cases where false joining takes place due to measurement noise. Also, we wish to use a high value for α to permit aggressive identification of constraints. We are able to use a much higher value if we can correct the resulting mistakes.

`AddLine`	Adds a new line constraint to the model
`JoinClosest`	Joins the latest constrained point to the nearest constraint
`JoinSecondClosest`	Joins the latest constrained point to the second nearest constraint
`JoinThirdClosest`	Joins the latest constrained point to the third nearest constraint
`MergeTwo`	Merges the new point and the closest and second closest constraints
`MergeThree`	Merges the new point and the three closest constraints
`JoinAndSplit`	Joins the latest constrained point to the nearest constraint and splits it in two

Table 1: *The set of operations implemented.*

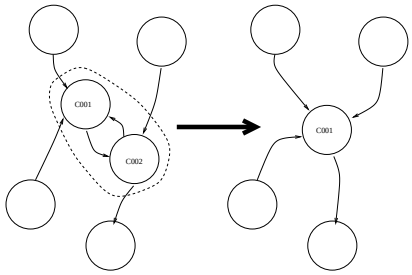

Figure 1: *Merger of two discrete states involves deleting events connecting the two states and changing other connected events to refer to the new state.*

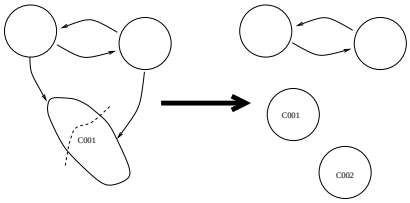

Figure 2: *Split of a discrete state (due to a split of the underlying constraint manifold) results in the loss of some information because the discrete events are simply discarded.*

4 Discrete Event Model Identification

For mobile robot modeling, the discrete event transitions do not cause the robot to instantaneously shift location. Therefore, we have a simpler hybrid dynamic model than is often used (neglecting jump conditions that exist in more general hybrid dynamic models). Thus, we identify the set of discrete events $\Sigma \subset \mathcal{Q} \times \mathbb{R}^2 \to \mathcal{Q}$.

The identification of the discrete event set Σ basically proceeds by observation. Observations of discrete state changes and the continuous state when these occur are recorded. A simplistic procedure would be to determine the set of constraint manifolds as described in Section 2 above and, once these had been fully identified, observe the system for sufficient time to determine the set of discrete events. However, this approach is unsatisfactory because it is not incremental in nature, it does not provide the set of discrete events during the identification process. In addition, this naive approach poses some non-trivial issues such as: (i) How do we determine when the full set of constraint manifolds has been determined? (ii) How do we know when all discrete events have been observed? Instead, we propose an incremental approach, identifying the constraint manifolds and adding to the set of discrete events simultaneously.

The simultaneous approach raises the issue of how do we maintain the set of discrete events when existing discrete states are modified. When two constraint manifolds are merged, the corresponding discrete states is merged (see Figure 1). In this case, discrete events involving either of the original discrete states are changed to use the new merged discrete state instead, as shown in Figure 1. When a constraint manifold is split new discrete states are created and it is not clear which of the new discrete states any discrete events should be assigned to (see Figure 2). For example, a discrete event may only apply to one end of a line constraint. If the line constraint is split then it is not obvious which of the new sub-lines should have this event (without considering additional information). For this paper, a simple approach is adopted which is to discard any discrete events involving discrete states that are split, as shown in Figure 2. This approach is acceptable because of the ease of acquiring discrete event observations.

5 Control Command Synthesis

The problem of control command synthesis is to find a command or series of commands to move to a given target point. Here we are interested in a controller which can move to the target with incomplete information about its environment. There are many possible approaches to this problem and it has been the subject of considerable research (e.g. [15, 16]). We propose a controller based upon intuitive reasoning about its behaviour.

Clearly, if the robot is unconstrained then the controller should attempt to follow the straight-line path to the target. This strategy will succeed in many cases. If the robot becomes constrained and the constraint prevents direct motion to the target then we must follow the constraint in an attempt to find a way around it. The approach to this task depends upon the amount of information that is available.

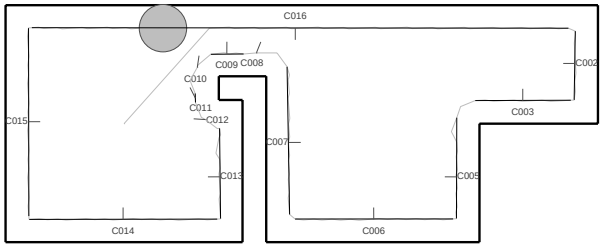

Figure 3: *Constraints identified using the wall-following controller.*

If the model contains any discrete events or transitions from the current constraint then we can search to see which option is most desirable. The proposed controller searches the discrete event graph, starting from the current state, to find the state closest to the target point. A discrete event trajectory is then planned to move from the current discrete state to the target discrete state. Here we see the power of the discrete event framework: a complex trajectory planning exercise has been reduced to a graph search and a number of simple wall-following problems.

Alternatively, if the robot is constrained and there is no model information about the current constraint, then we must attempt to follow the current constraint. There are two possible directions to turn to follow the constraint: either to the left or right. Initially, we choose the direction that leads towards the target. However, if the robot travels more than a certain distance threshold and is moving away from the target, we reverse the direction of travel and double the threshold. This process is repeated until the robot manages to move around the obstacle.

This simple controller is proposed to demonstrate the ease with which a controller can be developed using the hybrid dynamic systems framework. Also, the controller is used to demonstrate the feasibility of on-line identification i.e. simultaneous identification and control. Note however, that this controller has not been exhaustively studied and may fail for some (pathological) cases.

6 Results

A simulator for the robot and constraint identification was implemented with line primitives and the operations show in Table 1. The robot motion is simulated in steps of $0.01m$ and, hence, the robot positioning error is $\epsilon = 0.01m$. The robot has a diameter of $1m$ and the environment is $36m$ by $14m$ and a value of $\alpha = 0.002$ was used. First, results for the identification of the hybrid dynamic model will be presented, followed by results illustrating the operation of the controller with no initial knowledge of the environment.

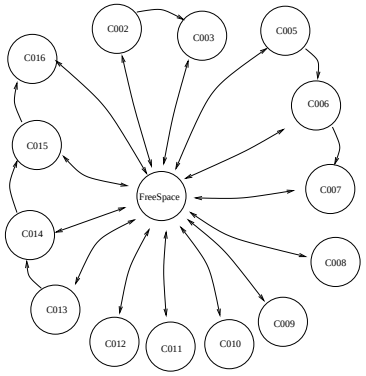

Figure 4: *Discrete event model developed by the wall-following controller.*

6.1 Hybrid Dynamic Model Identification

We demonstrate the effectiveness of the identification process with a simulation of a wall-following controller. Figure 3 shows the path that the robot followed and the constraints that have been identified. The robot has identified 218 constrained points around the perimeter of the room and these points have been formed into 14 constraints which provide a good model of the environment. The largest geometric primitive, **C016** has 75 points.

Figure 4 shows the discrete event model that was identified during the run of Figure 3. This model is almost complete and demonstrates how quickly a discrete event model can be determined. The only information missing from the discrete event model of Figure 4 is that the transitions connecting the states which involve two states should be bi-directional. This is because the wall-following controller only moved in a clockwise direction and so only a subset of the transitions were observed.

These results illustrate the ability of the proposed method to identify the constraint manifolds from few observations. Furthermore, with the power of non-monotonic reasoning, it corrects any mistakes that are made along the way, resulting in a very clean model. The discrete event model is also identified at the same time as the constraints, providing a topological map, which is useful for navigation, with a formal basis on the constraints.

6.2 Controller Operation

The operation of the controller was also simulated for the most interesting case where no *a priori* information is available. The environment and the path followed by the robot are shown in Figure 5. The robot starts at the left and finishes at the target point at the right. Initially, the robot is unconstrained and so attempts to move directly to the target. Unfortunately, there is an obstacle in the way and the robot becomes constrained by this obstacle. The initial constraint is labelled `C000` and the controller decides to maintain the constraint.

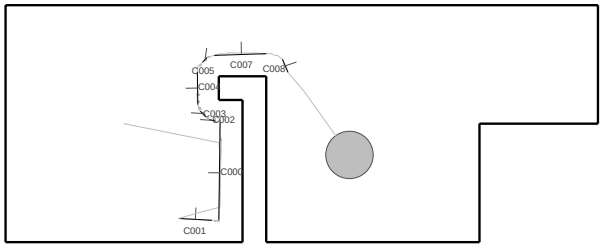

Figure 5: *Trajectory followed by the goal-seeking controller (finishing at the goal location).*

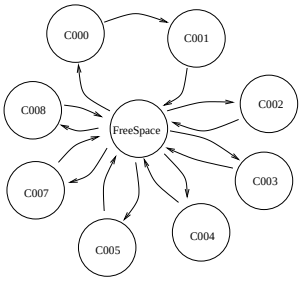

Figure 6: *Discrete event model developed by the goal-seeking controller.*

The controller moves downwards initially to try to get closer to the target. After following C000 for a while (and extending it), the robot runs into the bottom wall. This new constraint is identified as C001 and the controller follows it. After a short distance, however, the distance threshold is reached and the controller gives up on this direction of travel. Therefore, the controller attempts to move directly to the target again, once more reaching C000. This time, the controller follows the constraint upwards, identifying the new constraints C002, C003, C004, C005, C007, and C008. After the identification of C008, the robot moves into free space and again moves directly to the target. This time, the controller is successful.

Figure 6 shows the discrete event model of the environment that was developed during the navigation experiment. This example of the controller operation demonstrates how quickly a complete discrete event model can be identified while still successfully completing the navigation task.

7 Conclusions

The method presented here uses non-monotonic reasoning to identify a hybrid dynamic model for a robot moving in an office-type environment. The hybrid dynamic model has many advantages including that it can be easily used for path planning and that it explicitly models the abrupt changes in dynamics that occur as the robot becomes constrained by obstacles in the environment. This paper proposed a technique for the identification of the constraints imposed upon a mobile robot by the environment as well as identifying a discrete event model. The results presented illustrate the effectiveness of the method in identifying a good model of the environment from relatively few observations. A simple controller was also presented which can use a partial discrete event model to successfully navigate to target points. The identification process continues while the controller is operating, providing a continuously improving model of the environment. Simulated paths were presented which demonstrate the results of simultaneous identification and the control.

References

[1] K. O. Arras and R. Y. Siegwart. Feature extraction and scene interpretation for map-based navigation and map building. In *Proceedings of SPIE, Mobile Robotics XII*, volume 3210, 1997.

[2] D. J. Austin and B. J. McCarragher. Discrete event modelling for mobile robots. In *Proc. Intl Conf. on Field and Service Robotics (FSR '97)*, pages 364–369, December 1997.

[3] D. J. Austin and B. J. McCarragher. Geometric constraint identification and mapping for mobile robots. Accepted for Journal of Robotics and Autonomous Systems, 2000.

[4] David J. Austin. Non-monotonic geometric mapping for mobile robots. Submitted to IAS-00, Venice, 2000.

[5] T. R. Balch. Grid-based navigation for mobile robots. *The Robotics Practitioner*, 2(1), 1996.

[6] Edward A. Bender. *Mathematical Methods in Artificial Intelligence*. IEEE Computer Society Press, 1996.

[7] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, 1987.

[8] J. Kosecka. *A Framework for Modelling and Verifying Visually Guided Agents: Design, Analysis and Experiments*. PhD thesis, Grasp Lab, University of Pennsylvania, March 1996. Available from http://www.cis.upenn.edu/ janka/publ.html.

[9] J. Kosecka and R. Bajcsy. Discrete event systems for autonomous mobile agents. *Journal of Robotics and Autonomous Systems*, 12(B14):187–198, 1994.

[10] J. Kosecka, H. I. Christensen, and R. Bajcsy. Discrete event modelling of navigation and gaze control. *Intl. Journal on Computer Vision, Special Issue on Qualitative Vision*, 12(3):295–316, 1995.

[11] C. Kunz, T. Willeke, and I. Nourbakhsh. Automatic mapping of dynamic office environments. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 2, pages 1681–1687, 1997.

[12] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox. Dynamic map bulding for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):286–298, 1992.

[13] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, 1987.

[14] F. Schönherr, J. Hertzberg, and W. Burgard. Probabilistic mapping of unexpected objects by a mobile robot. In St. Orphanoudakis, P. Trahanias, J. Crowley, and N. Katevas, editors, *Proc. Computer Vision and Mobile Robots Workshop (CVMR-98)*, pages 46–53, Santorini, Greece, September 1998.

[15] N. M. Sgouros, G. Papakontantinou, and P. Tsanakas. Localized qualitative navigation for indoor environments. In *Proc. 1996 IEEE Intl. Conf. on Robotics and Automation*, pages 921–926, 1996.

[16] S. Thrun and A. Bucken. Integrating grid-based and topological maps for mobile robot navigation. In *Proc. of the Conf. on Artificial Intelligence*, August 1996.

[17] S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pages 1546–1551, May 1998.

[18] B. Yamauchi. Mobile robot localization in dynamic environments using dead reckoning and evidence grids. In *Proc. 1996 IEEE Intl. Conf. on Robotics and Automation*, pages 1401–1406, April 1996.