Learning Probabilistic Models for State Tracking of Mobile Robots

Daniel Nikovski and Illah Nourbakhsh

The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, {danieln,illah}@ri.cmu.edu

Abstract

We propose a learning algorithm for acquiring a stochastic model of the behavior of a mobile robot, which allows the robot to localize itself along the outer boundary of its environment while traversing it. Compared to previously suggested solutions based on learning self-organizing neural nets, our approach achieves much higher spatial resolution which is limited only by the control time-step of the robot. We demonstrate the successful work of the algorithm on a small robot with only three infrared range sensors and a digital compass, and suggest how this algorithm can be extended to learn probabilistic models for full decision-theoretic reasoning and planning.

1 Introduction

One of the primary objectives of mobile robots is to be able to navigate safely and reliably in their environments, in spite of the significant uncertainty usually accompanying their sensing and action. In this paper, we are considering a well-established robotic task — mobile robot localization and state tracking, while following walls in an unknown environment [4, 3], and propose a solution based on learning a probabilistic dynamic model in the form of a Hidden Markov Model (HMM). The traditional approach to mobile robot navigation is to supply the robot with a full and detailed model of its problem domain, and use this model for localization, state tracking, and planning. As of recently, the use of probabilistic models and decision-theoretic planning and state tracking methods is becoming increasingly popular, and has resulted in several successful mobile robot architectures [5, 2]. Consequently, our choice of representation has also been a probabilistic model.

The practical downside to providing a full and detailed world model to the robot is the significant expense of time and effort on the part of a human designer. Machine learning algorithms for acquiring autonomously a world model are very appealing and likely to result in huge savings of time and effort, and many mobile robot architectures include a learning component. Both known studies on the task we are considering — position tracking while following walls — have used a learning algorithm. However, the representation these studies used (a spreading-activation neural network) has resulted in very limited spatial resolution — a shortcoming that we have aimed to eliminate by the use of probabilistic models.

2 Position tracking during wall following by a mobile robot

Some of the basic competencies of a mobile robot are obstacle avoidance and boundary tracing, which collectively result in a wall-following behavior. Implementing these competencies on a mobile robot is relatively easy — infrared range (IR) sensors can be used as "whiskers" in a fast control loop. In our experiments, we used a simple mobile robot with two differential-drive wheels and a passive caster, three IR sensors, and a digital compass (Fig. 1). The first IR sensor (IR1) points forward, while IR2 points to the right and is placed *in front of* the right wheel, and IR3 also points to the right, but is placed *behind* the right wheel. These three IR sensors were sufficient to implement wall following behavior.

The robot is controlled by a Palm Pilot III hand-held computer with a DragonBall CPU (clock rate 16MHz), running PalmOS 3.1. The control step, including the time to record the sensor readings in a log file, is $290ms$. The robot is also supplied with a Dinsmore digital compass, which senses the local magnetic field and has a resolution of 45^o (eight directions).

Nehmzow and Smithers [4] and Matarić [3] used similar robots and considered the task of self-localization of the robot along the outer contour of its environment. The objective of the robot is to build an internal representation of this contour and be able to determine its position by means of this internal representation. It is desirable to have as high spatial resolution as possible, while at the same time maintaining high accuracy of localization. These two requirements are contradictory, because increasing the number of distinguishable states also increases the chance of erroneous localization.

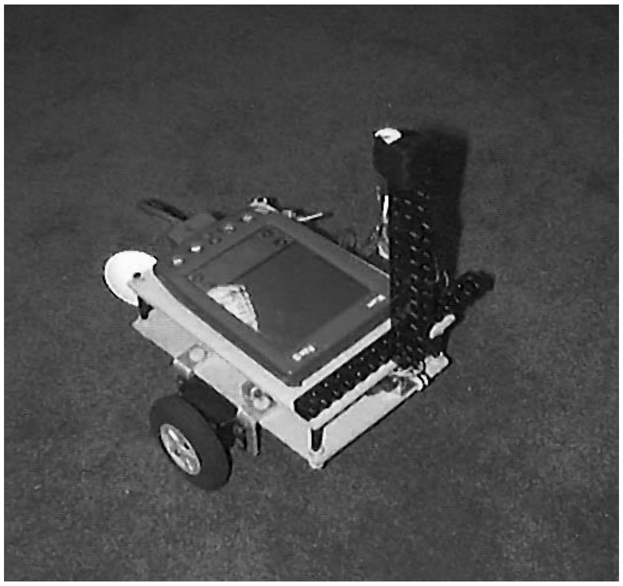

Figure 1: *The robot, Mr. Spoon, equipped with two differential-drive servo motors, three IR sensors, and a Dinsmore digital compass placed on a mast to reduce magnetic interference from the motors.*

Nehmzow and Smithers used a spreading-activation Kohonen neural network with a fixed number of nodes (neurons) as a world model. The weights of this network were updated in the process of learning a model of the environment. They used the output of the motors of the robot, instead of sensory input, to train the neural net. The biggest limitation of their approach is the fixed number of nodes in the model, and thus the resulting limited spatial resolution during localization. In fact, even though they started with a neural net of 50 nodes, the robot could ultimately distinguish only 11 locations, mostly corners of the outer boundary of the robot's environment [4].

The limited spatial resolution is the main problem with Matarić's approach too [3]. She chose three large, stable, and reliably detectable landmark types: left walls, right walls, and corridors, along with a default landmark type for irregular boundaries. While the chosen set of landmarks resulted in reliable position tracking, the robot was inherently incapable of determining where it was along a particular wall or a corridor, for example. We have sought to eliminate the limited spatial resolution resulting from the approaches described above by means of learning probabilistic models of the environment in the form of Hidden Markov Models (HMMs).

Probabilistic models of mobile robot environments have emerged as an alternative to those based on neural networks, and successful applications of probabilistic occupancy grids have been reported [8]. However, such methods typically build full two-dimensional probabilistic maps, which is not possible with the limited sensors of our experimental robot and no odometric information at all. Still, the essential ideas of learning probabilistic models for robot navigation can be applied on our robot as well, as discussed below.

3 State tracking with HMMs

An HMM has a set of states S which are not directly observable — instead, only observation symbols from the set O are perceived. In HMMs, the dependency of the emitted symbols on the state is not deterministic, but specified by a probabilistic emission function E which maps states in S on probability distributions over O. Likewise, the transition function of an HMM is not deterministic either — transitions between states are specified by a stochastic transition function T which maps predecessor states in S onto probabilistic distributions over all successor states, also in S. In addition, an initial probability distribution π over S specifies how likely it is for the HMM to start out in each of its states.

The structure, transition, and emission functions of an HMM employed by a mobile robot represent the laws, according to which the state of the robot evolves as a result of its actions. The states S of the HMM correspond to all possible locations of the robot — these states, however, are not directly observable by the robot due to its limited and noisy perceptual abilities. When the robot is in a particular state $s \in S$, it will observe one or more sets of readings from O. The observations are derived from the actual reading of the physical sensors the robot is equipped with: IR readings, sonars, compass directions, etc.

The relationship between states and observations is neither repeatable, nor unambiguous: the robot might perceive different sensor readings even when staying at exactly the same state, and it can also perceive the exact same readings at completely different states (for example, in two different corners). The former effect is a result of the inherently noisy sensors of mobile robots, while the latter effect, also known as perceptual aliasing, is due to the limited range and resolution of most sensors used on practical robotic systems.

In spite of the high uncertainty present in the operation of mobile robots, a number of algorithms exist, which allow successful localization, state tracking, and planning based on probabilistic models of the robot and its environment [2, 5]. These algorithms maintain a belief distribution $Bel(s)$ over all possible states and update it by means of the transition and emission matrices of a probabilistic model in a well-known prediction-estimation loop. During prediction, the agent estimates how the belief state at time $t-1$ will change if the system advances to the next stage at time t, based on the general knowledge about how actions affect states encoded in the transition probability function of the HMM:

$$\hat{Bel}(s_t) = \sum_{s_{t-1}\in S} P(s_t|s_{t-1})Bel(s_{t-1}),$$

where $\hat{Bel}(s_t)$ is the estimated belief state at time t for all states s in S, $Bel(s_{t-1})$ is the belief state at time $t-1$, $P(s_t|s_{t-1})$ are transition probabilities of the HMM, and the sum runs over all states in S. During the estimation phase, the newly observed percept o_t is used to update the estimated belief vector:

$$Bel(s_t) = \alpha P(o_t|s_t)\hat{Bel}(s_t),$$

where $P(o_t|s_t)$ are the emission probabilities of the HMM and α is a normalization coefficient. The most likely state at each step is the one whose corresponding element of the belief vector is largest. Within this prediction-estimation mechanism, localization and state tracking are only slightly different tasks: while for state tracking the initial state is assumed to be known (a single peak in the initial distribution π of the HMM), for the localization task π is assumed to be uniform (the robot is equally likely to be in all possible states).

Multiple real-world robotic systems exist, which use similar belief updating schemes to handle navigation in large environments [5, 2]. The practical obstacle to their widespread use is the necessity to supply the robot with a detailed probabilistic model of its environment. The following section proposes one algorithm for autonomously learning such probabilistic models for the task of mobile robot localization while following walls.

4 Learning HMMs for Mobile Robot Localization

The objective of the learning algorithm is to acquire an HMM from an observation trace recorded while the robot is following the outer boundaries of its environment, controlled by the set of rules described above. Raw sensory readings are recorded at each control step ($290ms$) and include the ranges of the three IR sensors (in centimeters), the current compass direction (discretized internally by the digital compass into eight principal directions), and the chosen action (left, right, straight).

Learning in HMMs with a known structure can be performed by means of the Baum-Welch algorithm, a form of the general Expectation Maximization algorithm [6], or by gradient-based optimization [7]. When the structure of the HMM is not known, the algorithm must additionally search the space of all structures [1], which is computationally very expensive. We have pursued an alternative approach to learning probabilistic models with hidden state, which is very different from the methods described above. The rationale behind it comes from the engineering field of system identification which deals with the almost identical problem of acquiring state-space models of dynamic systems from observations, even though the mathematical formalism most often employed there is that of linear differential or difference equations involving continuous variables. A large number of algorithms in that field use the fact that although states might not be directly identifiable by the immediate observations, a long enough sequence of past observations is sufficient to distinguish states. Likewise, if perceptual aliasing prevents us from determining unambiguously that the system has visited the same state at two different moments in time by simply looking at the emitted observations at these times, it might still be possible to make a definite determination by comparing the two sequences of observations before these two moments. Furthermore, if the whole observation trace is available, we can also compare the sequences after the two moments as well.

When the algorithm finds out by comparing trajectories that the system must have visited the same state (position) at two different moments in time, it establishes an equivalence class between these two moments. By finding the correct disjoint equivalence classes for each moment in time, the algorithm in effect introduces a set of states for the HMM and determines the sequence of states visited for each time step of the sequence by labeling it with its equivalence class.

Once it has been determined which state of the HMM was visited at each step in time, computing the probabilities in the transition and emission tables of the HMM is trivial and reduces to counting frequencies of occurrence, because at this point there are no more hidden variables in the model.

The current learning problem has significant constraints, which can be exploited to search exhaustively all possible models in a reasonable time. Such a constraint is the fact that the robot is always tracing the same boundary over and over in a loop, and if we determine that the period of the loop is L time steps, then we should merge together the states at times $0, L, 2L, 3L, \ldots$, then the states at times $1, L+1, 2L+1, 3L+1, \ldots$, and so on all the way up to merging the states at times $L-1, 2L-1, 3L-1, 4L-1, \ldots$. Consequently, we have to search only among all possible periods L, computing a matching score for each of them, and then choose the period with the highest matching score.

When computing the score for a particular period L, though, the algorithm would have to match all possible pairs of states which appear at this period throughout the whole sequence of N observations. So, if the robot has done n circles, where n is the largest integer such that $nL \leq N$, the algorithm would have to compare $n(n+1)/2$ or $n(n-1)/2$ matches, depending on whether n or $n+1$ candidate states are compared. So, the overall complexity of the matching process is $O(Nn^2)$.

We found that the sequence of compass outputs alone was enough to serve as an indicator of the location of the robot, for the purpose of matching trajectories. Hence forward we will designate this sequence of compass outputs as O_i, $i = 0, N-1$, where $O_i \in \{N, NW, W, SW, S, SE, E, NE\}$. We defined the local distance $d(i,j)$ between time steps i and j to be the circular distance between the directions at these times; this distance is always within the interval $[0,4]$. Then, the global matching score for a period L can be computed as:

$$D(l) = \frac{1}{n(n-1)l} \sum_{k=0}^{l-1} \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} d(il+k, jl+k).$$

$D(l)$ is computed for every integer l in the interval (L_{min}, L_{max}), and the best estimate for the period L is taken to be $L = argmin_l D(l)$, since $D(l)$ is a discrepancy measure (global distance). The obvious choice of $L_{min} = 1$ and $L_{max} = N$ results in a lot of wasted computation; instead, these two bounds can be easily improved by determining how many loops the robot has performed while collecting the observation data. To this end, we set a counter $C = 0$ at the start of the sequence and increment it by one any time when the compass reading changes counter-clockwise (e.g. from north to northwest), and decrement it by one if the reading changes clockwise (e.g., from south to southwest). When there is no change in compass direction, the counter remains unchanged too. After all N readings in the sequence have been processed, the integer part n of the ratio $C/8$ is a reliable estimate of how many complete loops the robot circled. Hence, the limits can be modified as $L_{min} = N/(n+1)$ and $L_{max} = N/n$. Note, however, that attempting to find the period itself as $8N/C$ results in a very imprecise estimate, and leads to the creation of unusable HMMs.

5 Experimental Environment and Results

We tested our algorithm in the experimental world shown in Fig. 2, which consists of walls in a regular office-building corridor and pieces of cardboard to close off the contour. The control loop and data trace acquisition part of the algorithm were implemented in C++ under PalmOS and ran on the robot, while the algorithm for building the HMM was implemented in Matlab on a UNIX workstation, due to the low speed of the CPU on-board the robot (16MHz).

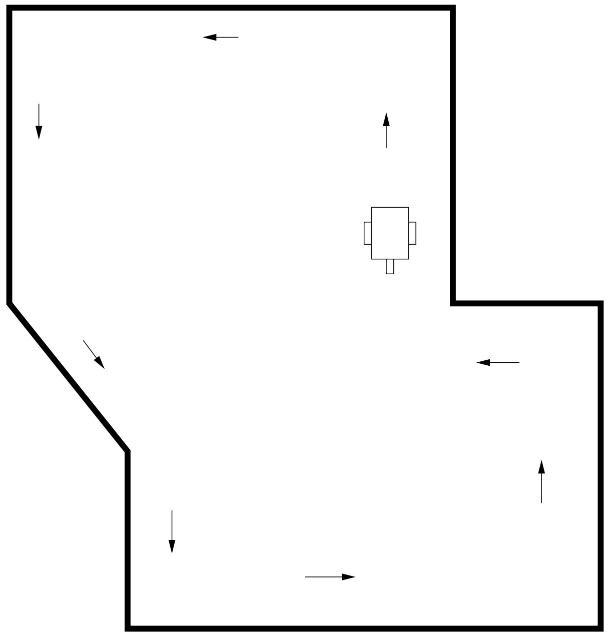

Figure 2: *Experimental world. Approximate sizes are $3m$ by $3m$. The initial location of the robot is shown along with its direction of motion.*

We collected an observation trace of $2,000$ time points while the robot was following the contour of the environment. We split these data into a training set ($N = 1,500$ data points), and testing set (500) points. The points in the training set were used by our algorithm to build an HMM, and the sequence in the testing set was used to evaluate the ability of the robot to localize itself and track its state by means of the learned HMM. The training data is shown in Fig. 3.

The cumulative number of direction changes over the $1,500$ training steps was found to be $C = 28$, or the equivalent of 3.5 circles around the contour. As noted above, this does not necessarily mean that the true period is exactly $1500/3.5 = 429$ time steps; this estimate only helps the algorithm narrow down the possible search interval for the period, so that $L_{min} = 1500/4 = 375$ and $L_{max} = 1500/3 = 500$. The global trajectory mismatch $D(l)$ was computed for the remaining 126 values; the results are shown in Fig. 4. The minimal mismatch between the trajectories of multiple loops was found to be at $L = 424$. This is close to the rough initial esti-

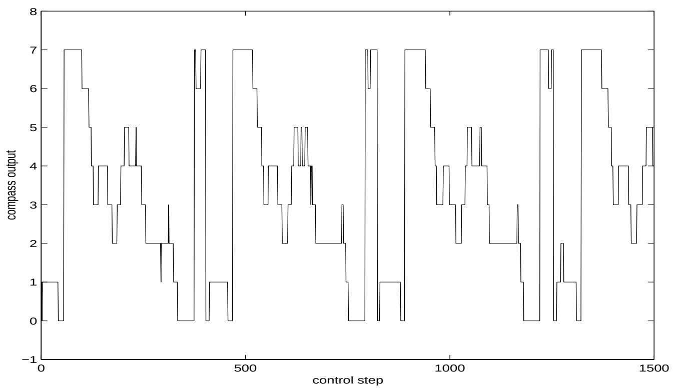

Figure 3: *Training data. Encoding of directions:* $N = 0$,$NE = 1$,$E = 2$,. . .,$NW = 7$.

mate of 429 obtained from the direction-change counter, but still sufficiently different so that the initial estimate would have led to erroneous models.

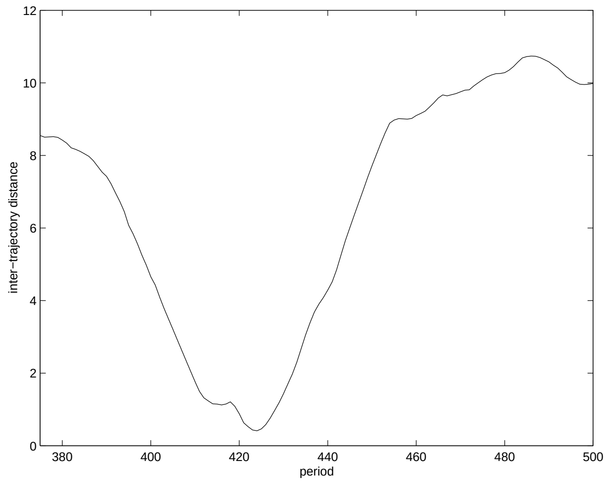

Figure 4: *Mismatch (inter-trajectory distance) as a function of the period. The minimum mismatch is at* $L = 424$.

Once the period L has been determined, the number of hidden states can be fixed to L, and the positions along the contour can be labeled with the number of the corresponding state. Estimating the emission probabilities $E_{ij} = P(O_i|S_j)$ of the HMM for a particular state S_j amounts to recording how often each observation symbol O_i (in this case, compass direction) was output by the compass, while the robot was at state S_j. Note that even when the robot is at a straight segment of the contour (a long wall), the emission probabilities for the corresponding states are often not deterministic — the reason is the well-known high-frequency wavering component of the wall following behavior, which sometimes results in rapid switching of compass directions even along straight walls. Since the phase of this high-frequency component is, in general, different between consecutive circles, the robot often observes different compass directions at the same position along the contour.

The transition probabilities are similar for all states and can be determined from the basic physical constraint that at each time step, the robot advances to the next state in the circular ring formed by all states and closed between states S_{L-1} and S_0. The transition can be deterministic, which forces the prediction part of the belief tracking algorithm to simply shift the belief distribution along the ring of states; alternatively, the transition function can have a stochastic element, allowing for the HMM to stay at the previous state (position), or even advance past the next state. There are several justifications for such a stochastic transition matrix. The exact period of the robot's motion along the contour is not an integer number, and the rounding error would accumulate, if the transition matrices do not account for it. Furthermore, the way the robot's wall-following behavior negotiates corners is visibly different at each circle, and the traveled distance, hence the exact progression along the state chain, differs too.

Fig. 5 shows a graph of the most likely state as determined by the localization algorithm, which performs the prediction-estimation cycle for belief updating by means of the learned HMM and the observations in the testing sequence. The localization algorithm starts with a uniform belief distribution, i.e. the robot has no prior knowledge as to where it is. Since we do not have the ground truth about exactly where the robot was during each control step of the testing period, as a reference we have provided the labeling of states in the testing sequence that would have resulted if the labeling of the training sequence had continued past the boundaries of the training set into the testing one.

The results show that the prediction-estimation algorithm initially cannot resolve a perceptual ambiguity and it takes more than 30 control steps (10 seconds) to recover the true position of the robot. It is also visible that for the most part the algorithm is able to track the most-likely robot state correctly, even though the robot gets lost six times. This is due to encountering an observation, for which the emission probability is zero and hence should not have been encountered. In such a case, the algorithm for belief updating realizes that it has gotten lost, resets the belief distribution to uniform, and starts the localization process again. It should be noted, though, that localization always succeeds in these re-localization cases, which is a further evidence that the learned HMM can

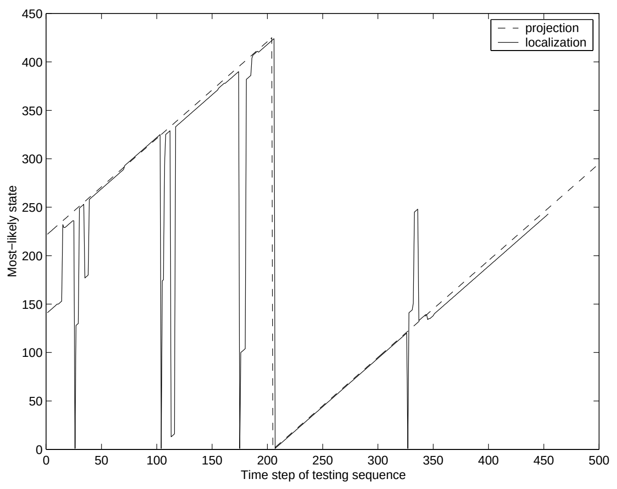

Figure 5: *Experimental results. The localization algorithm uses the learned HMM to determine the most-likely state of the robot based on observations, starting with an initially uniform belief distribution. The projected labeling of states from the training set is given for comparison, too.*

be used for robust robot localization while following the outer contour of the environment.

6 Conclusions and Future Work

We described an algorithm for learning an HMM from a sequence of directions output by a digital compass onboard a mobile robot, and used this HMM for state localization and tracking by means of updating belief distributions. The algorithm was able to perfectly localize the robot several times, and maintain (track) the correct belief distribution most of the time. The spatial resolution that the robot can achieve was much higher than competing approaches based on neural nets, and is limited only by the control hardware of the robot.

A problem for our method, however, are percepts, which have never been encountered during training and are thus considered impossible. Observing such a percept immediately causes the robot to lose track of its position and reset its belief distribution back to uniform. Several solutions to this problem are possible. First, some small prior probability for observing each percept in each state can be added to the emission tables, so that the belief distribution would not be reset when unknown percepts are seen. Second, the eight compass directions can be modeled not as independent discrete symbols, but as continuous values, and the distance between these values can be exploited while learning the HMM: for example, if the compass direction north has been observed in a particular state, the probability of also observing northwest and northeast in that state could increase too.

A promising direction for extension of the current system is to try to learn a full planar model of the whole environment of the mobile robot by adding shortcuts between sections of the outer boundary. So far, the robot can only traverse the outer contour using the wall following behavior, and has no choice of actions. The robot still can, however, turn away from the wall and follow a straight line until it hits a boundary again. Because the localization abilities have been shown above to be very good, the robot could conceivably backtrack the localized state to the point when the boundary was encountered, and thus establish a new transition in the HMM, which corresponds to a shortcut in the environment. If this is possible, the robot would have a choice of following the wall or taking a shortcut, and would be able to use the learned probabilistic model for decision-theoretic planning.

References

[1] Nir Friedman. The Bayesian structural EM algorithm. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138, San Francisco, July 24–26 1998. Morgan Kaufmann.

[2] Sven Koenig and Reid Simmons. Xavier: a robot navigation architecture based on partially observable Markov decision process models. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 91–122. MIT Press, Cambridge, 1998.

[3] M. J. Matarić. Integration of representation into goal-driven behavior-based robots. *IEEE Trans. Robotics and Automation*, 8(3):304–312, June 1992.

[4] U. Nehmzow and T. Smithers. Using motor actions for location recognition. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems. Proc. First European Conf. on Artificial Life*, pages 96–104, Cambridge, MA, USA, 1992. MIT Press.

[5] Illah Nourbakhsh. Dervish: an office navigating robot. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 73–90. MIT Press, Cambridge, 1998.

[6] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, page 4, January 1986.

[7] Stuart Russell, John Binder, and Daphne Koller. Adaptive probabilistic networks. Technical Report CSD-94-824, University of California, Berkeley, July 1994.

[8] S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*, pages 1546–1551, Piscataway, May 16–20 1998. IEEE Computer Society.