Mobile Robot Navigation based on localisation using Hidden Markov Models.

Shahriar Rahman
Department of Computer Science and Electrical Engineering
The University of Queensland, QLD 4072
Email: imti@engineer.com

Alexander Zelinsky
Research School of Information Sciences and Engineering
The Australian National University, ACT 0200
Email: Alex.Zelinsky@anu.edu.au

ABSTRACT.

In this paper, we implement a method of mobile robot navigation using Hidden Markov Models (HMMs). It is a place-based navigation, in which the robot localises itself in previously learnt environment. We use the laser range data from the robot to scan the environment and to distinguish between different places. First the robot learns after several trials about the places, it then recognises the places again by finding a match between the learnt HMM network and the current environment. The system can generate HMM network representing a sequence of places that are associated with a mission. The system is used for a tour guide robot.

1 INTRODUCTION

A mobile robot's knowledge of its position and relative orientation to its environment is essential for autonomous robot navigation in indoor environments. Automatic recognition of places is important in localisation of the robot because it gives capabilities, that are, essential in dynamic environments [Aycard *et al*, 1998].

Mobile robot navigation has been researched and implemented in a number of ways, where localisation has been a significant part in the whole process. Localisation techniques, in general, fall into two broad categories:

- Passive landmarks - natural, and
  - artificial.
- Active landmarks - beacons.

Landmark methods depend on the recognition of landmarks to keep the robot localised geometrically. Landmarks may be given *a priori* (for example, the satellites in GPS) or learned by the robot as it maps the environment (for example, sonar landmarks). While landmarks methods can achieve impressive geometric localisation, they require either engineering the environment to provide a set of adequate landmarks, or efficient recognition of features to use as landmarks. Our original work on the mobile robot was based on this method [Kittler 1998]. On the other hand, dense sensor methods attempt to use whatever sensor information is available to update the robot's position. They do this by matching dense sensor scans with a surface map of the environment, without extracting landmarks features. Thus, dense sensor matching can take advantage of whatever surface features are present, without having to explicitly decide what constitutes a landmark [Gutmann *et al.*, 1998].

In landmark-based navigation positioning is done in two ways, either based on a map or a place. In the first technique, the robot uses its sensors to create a map of its local environment. As the robot navigates, this local map is compared to a global map previously stored in memory to check for a match and if a match is found the robot can compute its actual position and orientation in the environment. Although this method can give precise orientation and position at any time, it needs sufficient distinguishable features or landmarks to match the maps accurately [Aycard *et al.*, 1998].

With the place recognition technique, the robot localises itself relative to fixed and known places, such as a T-

intersection, or a door. The robot uses its sensors to recognise places in its local environment. The environment is represented as a graph, where the nodes represent places while the edges represent path from one place to another. Using dead reckoning and matching algorithm, the mobile robot computes the drift in position and orientation as it passes a previously learned place.

Indoor environments are highly structured, and it is possible to write software that reliably recognises features, such as a corridor or a door. This is not easy to do in an outdoor situation. Hence, place-based localisation may be useful compared to map-based systems in an indoor environment. Place recognition utilises rules to build a representation of places and then a statistical system to describe observations coming from the robot's sensors. Since in an indoor environment, there are certain general places, which need to be recognised, successful implementation of localisation technique, i. e. the place recognition technique and path planning may result in an autonomous mobile 'tour-guide robot'. One example of such robot is the '*MIT Polly robot*' [Horswill, 1994]. Such general features may be doors, end-corridors, intersections, etc. If the path is appropriately modelled and the robot is trained adequately, it should be able to guide any newcomer to the environment, which can even be arbritary. The stochastic modeling and training methods for this purpose have seen enriched by a methodology, which is based on Hidden Markov Models (HMM) [Acyard, *et al.*, 1997]. Applying this approach seems promising in mobile robot navigation as HMM's are doubly embedded stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations [Rabiner, 1989]. A place-based positioning technique may enable a mobile robot to recognise places, which are not readily observable or are hidden giving it capabilities to navigate in dynamic environments. In this paper, we present a method of learning and recognising places based on second-order Hidden Markov Models (HMM2). We show how our place learning strategy can be effectively used for localisation of a mobile robot while it is moving in an indoor environment and finally path planning so that the mobile robot acts like a tour-robot. The tour-path is represented as a graph, where each node corresponds to a place and edges are paths to go from one place to another. At first, we define 10 distinct class of places, such as right door, right T-intersections, etc. (figure 1). We then plan the path from starting to the end and call this the 'state sequence' the robot will see on its complete run. Then we test run it several times to see how it recognises the sequence and compare the two. We record the results in the confidence matrix, and afterwards use it to update the model. We program the robot to take a certain action at each place. The goal is to develop a system that can enable the robot to localise itself automatically even if some places are not present in the actual run. That is to say, if there is a mismatch between the pre-planned path and actual environment, it can take appropriate actions and still can complete the tour successfully.

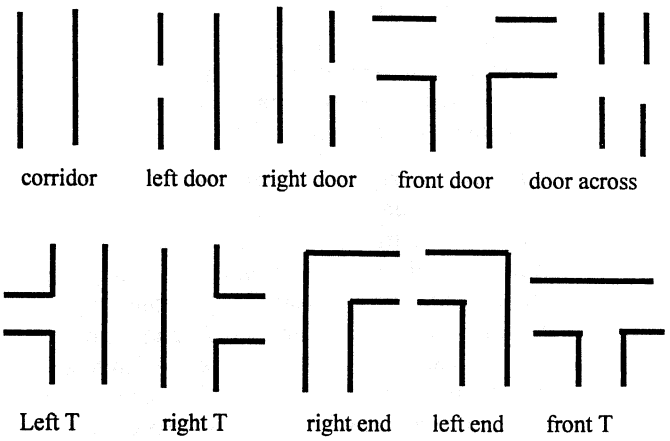
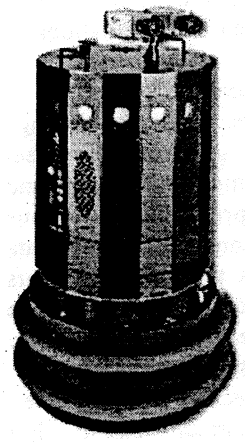


Figure 1: Ten Global Places

2 OUR ROBOT

The robot used for the purpose is a Nomad200 (www.robots.com/Nomad/). The robot is equipped with several sensors to enable it to estimate and correct its position as it follows a designated path. The sensors include infrared, laser, ultra sonic, vision, wheel encoders, and bump sensors. There is also a flux-gate compass on board, which is controlled by a microprocessor and is used to measure robot's orientation. A voice synthesiser enables the robot to 'talk'.



A vision system is used to monitor the tour from a remote location. Bump sensors detect collision with an obstacle and consist of two rings each with 10 independent pressure sensitive sensors located around the robot.

The laser sensor used to navigate the robot is the Sick LMS200 laser range finder (www.thomasregister.com olc/sick/home.htm).

Figure 2: 'Nomad' Robot.

The laser provides a two dimensional, 180° scan of the environment with up to 361 measured readings at a scan time of 40 ms.

3 OUR APPROACH

In our previous work, we used the mobile robot and its various sensors to use the robot as a tour-guide. It was programmed to perform the following tasks-

- Obstacle detection and avoidance
- Finding a wall
- Tracking one wall
- Tracking the mid-point of the corridor
- Turning at the end of the corridors
- Wandering the environment, and
- Going through doors

These behaviours are triggered either by the distance the robot has traveled, landmark detection or a combination of both. The distance travelled was measured using the wheel encoders, which give the x and y coordinates of robot's location in the environment. A predefined course of the tour was programmed into the Nomad, and was essentially a map-based navigation where the landmarks included the doors to the offices and the end of the corridors. Once the robot is in the approximate location of the landmark, the area is searched for the distinct features of the landmark. The Nomad was programmed to recognise obstacles in its path such as boxes or people. In these situations, the robot stops and asks the obstacle to be removed. It resumes its tour once the obstacle is removed.

The robot's environment consists mainly of straight corridors. These corridors are detected using a straight line-fitting algorithm on the data received by a laser range finder. From this algorithm, it was possible to determine the slope of the corridors with respect to robot's orientation. The robot uses this information in order to navigate its path through the environment. The robot tracks the walls by keeping to a pre-defined distance and the direction. A PD controller is used to keep the robot on course. This PD controller also reduces oscillations in Nomad's path as it follows a wall.

This method of navigation using 'dead-reckoning' introduces a number of problems. Due to slippage between robot's wheel and the ground, the wheel encoders could not always measure the wheel rotation and translation accurately. This produced erroneous information about the robot's position. The sensory data is cumulative and this compounds the effect of errors making it significantly inaccurate towards the end of the tour. And often too many unexpected obstacles in the environment also confuse the navigation algorithm.

To overcome these problems, we decided to incorporate the method described in [Acyard *et al.*, 1998]. For localisation purposes, we produce a confusion matrix to assign the level of confidence to a place that has been recognised. The idea is to give an initial and end-point to the robot and observe if it can derive the sequence of places according to the matrix. If some places are not visible or present (e.g. a door that is closed), by assessing the current state of the environment, the robot's position should be updated automatically. By comparing the original sequence and the places recognised, we can determine the highest probability of the robot to be at a particular place.

The problem falls into an HMM problem, namely given an observable sequence $O = O_1, O_2, \ldots\ldots, O_T$ and the model λ, how do we choose a corresponding state sequence $Q = q_1, q_2, \ldots\ldots, q_T$, which is optimal in some meaningful sense. This problem is usually solved using forward-backward algorithm [Rabiner and Juang, 1986]. We present our approach of solution and implementation of the method in the next sections.

4 HIDDEN MARKOV MODELS (HMMs)

A hidden Markov Model is characterised by-

- A set of states, N from beginning to end state interconnected such a way that any state can be reached from any other state, denote it as $S = \{s_1, s_2, s_3, \ldots\ldots\ldots, s_N\}$,
- A set of distinct observations per state, i. e. M,
- A matrix of state transition probability distribution $A = \{a_{ij}\}$ which is the probability of transition from state S_i to state S_j,
- A matrix of observation probability distribution at any state j, $B = \{b_j(k)\}$, and
- An initial state distribution $\pi = \{\pi_i\}$ which will represent the ideal state of the environment.

5 HMMs IN ROBOT NAVIGATION

In place-based mobile robot navigation, HMMs can be used for place learning and recognition. Using the observation probabilities of the model, it can generate the optimal state sequence in a dynamic environment and constantly update its position and localisation. First we need to choose the appropriate model and its elements before successfully implementing the method.

5.1 Model and State Transition Matrix

In a tour type of navigation, a model similar to left-right or Bakis model is most useful as it offers state transitions in a sequential manner (figure 3).

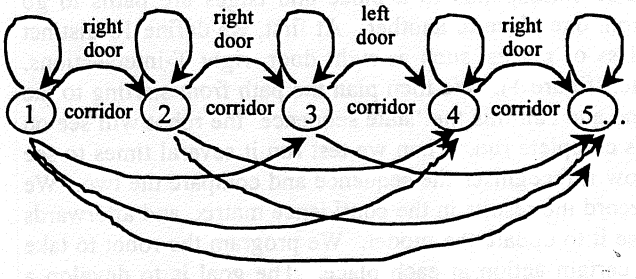


Figure 3: Topology of the Tour (taking into account some places are not visible)

In this model, the nodes may represent different places (1, 2, 3, 4, 5) in a navigation environment and the edges show the paths to go from one place to another. Assume that the robot can make the following mistakes [Aycard *et al.*, 1997] -

Insertions- robot recognises a non-existence place,
Deletions- robot misses a place, and
Substitutions- robot has confused a place with another.

Since the robot may insert a place by mistake, or skip one or more places, all of self, one-step and multi-step jumps are allowed. Then the model of navigation becomes very similar to figure 3.

This type of model has the desirable property that it can readily model situations whose property change over the time. This model also shows the state transition diagram of the tour. Since no backward transition is allowed in our robot tour (arrows going forward only), the model becomes simpler, i. e. no transitions are allowed to states whose indices are lower than current state-

$$A_{ij} = 0, \qquad j < i$$

And with multiple jump assumption, the state transition matrix looks as follows-

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

After constructing the state transition matrix for all the states, we can then find the optimal state sequence associated with a given observation sequence. Localisation of the robot at each state will be based on the observation of the past two states (second order) as recognised by the robot previously and local information of the current state. And this way the robot can learn about the environment. Once the learning and recognition phase of the model is done, an important assumption is made and that is, the state transition probabilities do not vary with time, i.e. the matrix is fixed for the entire tour.

5.2 The Confusion Matrix

Once the state transition matrix is determined, a confusion matrix is built, which contains the probabilities of the observable states given a particular hidden state. These probability distributions are represented in matrix form with the actual observation states in the environment as a function of what is recognised by the robot in its recognition runs. Subsequently the transition probabilites will be derived from the last line of the matrix. An example of such a matrix is shown in Table 1.

	c	dl	dr	da	tl	tr	re	le	fd	tf
Corridor	18	0	0	2	0	0	1	2	3	0
door left	0	19	0	1	2	0	0	2	0	0
door right	0	0	20	1	0	3	3	0	0	0
Door across	1	0	0	15	0	0	0	0	0	3
T-left	0	2	0	0	12	0	0	3	0	0
T-right	0	0	1	0	0	13	2	0	0	0
right end	1	0	1	0	0	1	14	0	0	1
left end	1	1	0	0	3	0	0	12	0	2
front door	1	0	0	0	0	0	0	0	16	3
T-front	1	0	0	0	0	0	1	2	2	11
Total	23	22	22	19	17	17	21	21	21	20
%recog.	78	86	91	79	70	77	67	57	76	55

Table 1: The Confusion Matrix

5.3 Localisation of the Robot

Once both the state transition and confusion matrix are derived, we use the forward-backward algorithm for the robot to localise itself in dynamic environments. We first define a variable as-

$$\gamma_t(i) = Pr\,(q_t = S_i | O, \lambda)$$

i.e., the probability of being in state S_i at time t, given the observation sequence O, and the model λ. This equation can be expressed in terms of forward-backward algorithm variables as-

$$\gamma_t(i) = \frac{\alpha_t(i)\,\beta_t(i)}{Pr\,(O|\lambda)} = \frac{\alpha_t(i)\,\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\,\beta_t(i)}$$

Since we are interested in the most likely state after each state, we only need to know the forward variable, making the equation simply-

$$\gamma_t(i) = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)}$$

$\alpha_t(i)$ accounts for the partial observation sequence O_1, O_2, ..., O_t and state S_i at t. It is possible to solve the state sequence that maximises the expected number of triple states (q_t, q_{t+1}, q_{t+2} – since it is a second order model). Instead of the exhaustive search through this process, we use a dynamic programming search known as *Viterbi algorithm* to find the forward variable $\alpha_t(i)$, which would give us the best state sequence, $Q = \{q_1, q_2, \ldots q_T\}$ given the observation sequence $O = \{O_1, O_2, \ldots O_T\}$. We compute the variable $q_t(i)$ recursively as follows-

Initialisation $\quad \alpha_1(i) = \pi_i b_i\,(O_1), \quad N \geq i \geq 1$

Recursion $\quad \alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)\,a_{ij}]\,b_j\,(O_{t+1}),$

$$N \geq j \geq 1;\ T\text{-}1 \geq t \geq 1$$

Here, π_i is the initial state probabilities at any state S_i, a_{ij} is the state transition probabilities from state S_i to S_j and

$b_j(O)$ is the probability of a place being resognised from the observation sequence O in a state i, S_i. Since, a_{ij} come from the state transition matrix and $b_j(O)$ come from confusion matrix, it is now easy to solve the problem of localisation.

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

Since the in-built sensors of NOMAD are not sufficiently reliable for scanning the environment, the laser range finder and the compass are used primarily to sense the direction of the robot and scanning environment. Other sensors are used only for obstacle detection and avoidance.

The laser is calibrated to collect data from a single scan of the environment and the points thus found are used to measure the angle of inclination of the robot to the floor. This angle is then used to find the free floor distance accurately in front of the robot. This distance measured this way is different for an obstacle or places like a door or an intersection. These differences in data will later distinguish different places such as corridors, doors, etc. The compass is used to give direction to the robot so that it can recognise which corridor it is in at any given time during the tour. At each place, the robot gives a brief description of the place as programmed into it using its voice synthesiser.

The robot is used to tour a floor of our building at the ANU. The floor consists of all the global places as defined and complex enough for a reasonable tour. Both trial and the actual tour are shown on figure 5.

6.2 Learning and Recognition Phases

First the robot is programmed to wander back and forth in the straight corridor shown above. From several trial runs, data is collected and compared to distinguish places. Thus the robot learned about different places.

Once the learning phase is over, it is run in the same setup to find out how it can recognise previously learned places. This recognition phase, in fact, produces the confusion matrix, which contains the probabilities of the robot's recognising a place correctly and localising itself at that place as a function of the actual environment.

We then develop the topology of the corridor (figure 3) and use it to derive the state transition probabilities.

Once both state transition and confusion matrix are derived, we can now test the actual tour using the *Viterbi algorithm*. This method is expected to produce all three types of errors to a considerable level in a reasonably noisy environment.

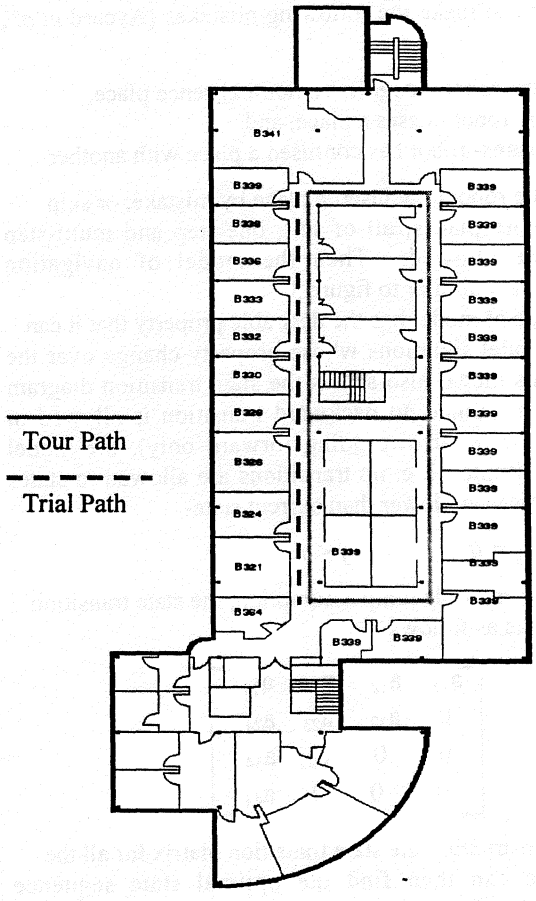


Figure 5: The Tour Path.

6.3 The Actual Tour

The actual tour of the building is the extended version of the test run. The probabilities distribution derived from the confusion matrix and state transition matrix are to be used and robot to give a complete tour of the building starting from the entrance and back. From the T-intersection in the middle, it is programmed to take decision based on date, i.e. to go straight on a Monday, right on a Tuesday, and so on.

Due to time constraints, the actual tour of the robot is not yet finished. As it seems, practical implementation of any theoretical research takes considerable amount of time. So far, the learning and recognition phases are being executed in order to develop rigorous understanding of the environment by the robot. Now, Nomad can recognise most common places, but still confuses some of them at times. Therefore, various algorithms to overcome this problem are being tested.

7. CONCLUSION

In this paper, we have presented the implementation of second order hidden Markov models in mobile robot navigation. The method is used for a tour-guide robot and should give successful tours of a planned path. It uses the crude idea of robot's position and the state of the environment to take appropriate actions and updates its tour graph accordingly.

The robot is controlled using a behaviour-based approach, and each of the behaviours is triggered throughout the tour depending on places. Although a HMM model enables the robot to recognise most places correctly, sometimes, if some places are not visible, it inserts a wrong place and thus ends up with wrong place information. Also the laser range finder of the robot gives erroneous information when it encounters big obstacles on its way. Currently these problems are being studied in the learning and recognition phases of the tour so that we can overcome them and then use the HMM model to have a successful 'Tour-Guide Robot'.

The next idea would be to use previously derived distance information along with a partially observable Markov decision process, which takes the robot displacement from the origin into account. This approach may help overcoming the above-named problems. In future, implementations by including this feature will allow the robot to start the tour anywhere in the environment and find its way to the end.

ACKNOWLEDGEMENTS

We acknowledge the help of Ali Gokt\gan and Dr. Peter Bartlett at the Systems Engineering Department, ANU. Also Peter Kittler, a final year engineering student, who implemented the initial tour program on the NOMAD robot.

REFERENCES

[Aycard *et al.*, 1998] Olivier Aycard, Jean-Fran| ois Mari, and Fran| ois Charpillet. *Second Order Hidden Markov Models for Place Recognition: New Results.* IEEE Press, NJ, to be appeared in ICTAI'98, 1998.

[Aycard *et al.*, 1997] Olivier Aycard, Dominique Fohr, and Jean-Fran| ois Mari. *Place Learning and Recognition Using Hidden Markov Models.* In proceedings of IEEE/IROS, pages 1741-1746, 1997.

[Rabiner, 1989] Lawrence R. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, Vol 77, No. 2, pages 257-286, February, 1989.

[Acyard *et al.*, 1998] Olivier Acyard, Pierre Laroche, and Fran| ois Charpillet. *Mobile Robot Localization in Dynamic Environments using Place recognition.* Proceedings of the IEEE/ICRA, pages 3135-3140, May 1998.

[Burgurd *et al.*, 1998] Wolfram Burgard, Andreas Derr, Dieter Fox, and Armin B. Cremers. *Integrating Global Position Estimation and Position Tracking for Mobile Robots: The Dynamic Markov Model Localization Approach.* Proceedings of IEEE/RSJ, pages 730-735, October 1998.

[Gutmann *et al.*, 1998] jens-Steffen Gutmann, Wolfram Burgurd, Dieter Fox, and Kurt Konolige. *An Experimental Comparison of Localization Methods.* Proceedings of IEEE/RSJ, pages 736-743, October 1998.

[Horswill, 1994] Ian Hoswill, *Visual Collision Avoidance by Segmentation.* Proceedings of IEE/RSJ, pages 902-909, May 1994.

[Rabiner and Juang, 1986] Lawrence R. Rabiner, and B. H. Juang. *An Introduction to Hidden Markov Models.* IEEE ASSP Magazine, Vol. 3, No. 1, pages 4-16, 1986.

[Kittler, 1998] Peter J. Kittler, *Tour-Guide Robot*, Final Year Thesis, FEIT, ANU, Canberra, 1998.