Map Representation Using Hidden Markov Models For Mobile Robot Localization

Jesus Savage, Oscar Fuentes, Luis Contreras and Marco Negrete

[1] *Bio-Robotics Laboratory, Universidad Nacional Autónoma de México (UNAM)*

Abstract. This paper describes a map representation and localization system for a mobile robot based on Hidden Markov Models. These models are used not only to find a region where a mobile robot is, but also they find the orientation that it has. It is shown that an estimation of the region where the robot is located can be found using the Viterbi algorithm with quantized laser readings, i.e. symbol observations, of a Hidden Markov Model.

Keywords: Hidden Markov Models, Robot localization, Viterbi Algorithm.

1 Introduction

It is the general case that Robots are conceived to perform a task, and the chosen world representation is highly correlated with the performance of the robot in such task [1]. The mapping process is restricted to the robot features, where a map is generated using a collection of sensor readings from a traversed trajectory in the scene given by the robot shape and its degrees of freedom, and this information might be useful to define the map [2] [3].

In the many years of research of robot navigation [4] and SLAM [5] [6] [7], several map representations have been proposed. From metric and topological to symbolic and probabilistic representations [8] [9].

In this work, we propose a probabilistic representation given by a Hidden Markov Model from a number of sensor readings and robot displacements; this approach has been proven useful for robot localization and navigation [10], [11]. We extend those works by incorporating the trajectory information in the Markov Model, allowing us to obtain the position and orientation related to a number of nodes on the trajectory.

The remaining of the paper is divided as follows: in Section 2 we introduce the VIRBOT system while in Section 3 we detail the cartographer module and develop the map representation using Hidden Markov Models based on the system characteristics. Then, in Section 4 we present a localization method using the proposed map representation. In Section 5 we detail the experiments and results and in Section 6 we conclude this work.

[1] ACKNOWLEDGMENT: This work was supported by PAPIIT-DGAPA UNAM under Grant IG-100818

2 The VIRBOT System

There are several architectures to control mobile robots, in our approach, the VIRBOT system [12], the mobile robot's operation it is divided into four general layers: Inputs, Planning, Knowledge Management and Execution, having each of them several subsystems, see Figure 1. Each subsystem has a specific function that contributes to the final operation of the robot.

This system has similar features presented in the INTERRAP agent architecture [13]

2.1 Inputs Layer

In this layer are the robot's internal and external sensors, as well as, the simulator that simulates these sensors when a simulated Robot is used.

2.2 Planning Layer

Action Planner: The objective of action planning is to find a sequence of physical operations to achieve the desired goals.

Motion Planner: This module receives, from the action planner, a set of locations that the robot needs to visit and it finds paths to reach them using a topological map and the A* Algorithm.

2.3 Knowledge Management Layer

Cartographer: This module creates and contains different types of maps for the representation of the environment. This section is explained in more detail in section 3

Knowledge Representation: A rule-based system, CLIPS, developed by NASA, is used to represent the robot's knowledge, in which each rule contains the encoded knowledge of an expert.

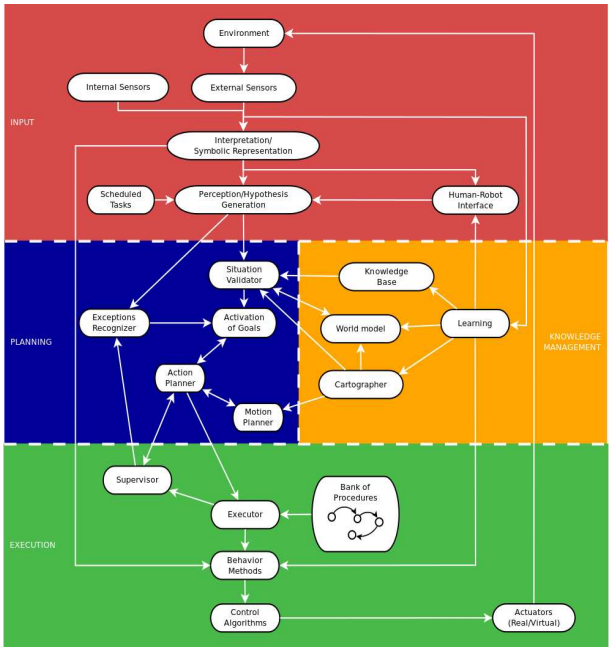

Figure 1. The VIRBOT System architecture consisting of several subsystems that control the operation of a mobile robot.

2.4 Execution Layer

This module executes the actions and movements plans.

3 Cartographer

The cartographer module is responsible for estimating the region where the robot is located. One of the problems of using only dead reckoning to estimate the robots' position x, y, θ is that, due to errors in the movement of the robot and errors in the sensors that measure them, after several movements, the real position of the robot can not be estimated accurately. In some robotics architectures, the ones based only on reactive behaviors [14], it is not necessary to know the exact pose of the robot but only an estimation of the region where is located. There have been several approaches based on Hidden Markov Models and Bayesian techniques to solve this problem [15]. In our system this task is done by a localization module, using laser readings, Hidden Markov Models and the Viterbi algorithm.

3.1 Hidden Markov Models

A Hidden Markov Model (HMM) is a two random variable process, in which X one of the random variables is hidden, and the other random variable O is observable [16]. The hidden random variable X represents the states, in a model that is a collection of states connected by transitions. In our system, the states represent the centroids of the regions in the free space where the robot navigates, see figure 2.

Each state has two sets of probabilities: an output probability, that provides the conditional probability that given that the system is in a particular state, a symbol is generated from a finite set of symbols, and a transition probability from going from one state to another.

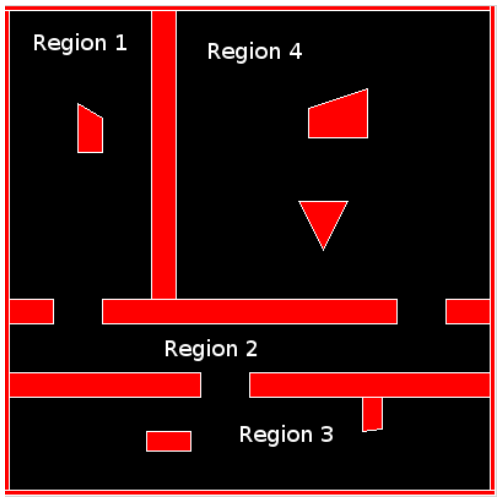

Figure 2. Navigation Regions.

3.1.1 Generation of the HMM Symbols

The symbols are obtained using clustering techniques like K Means or Vector Quantization (VQ). VQ techniques are used for data compression, given a set of vectors, $S_j = [s_1, s_2, ..., s_m]$ that represent lasers readings, see figure 3, a set of centroids are found which represent them.

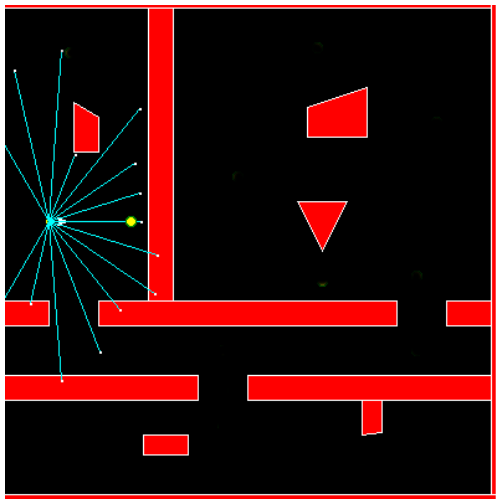

Figure 3. Laser data $S_j = [s_1, s_2, ..., s_m]$.

The collection of centroids is called a codebook. This codebook is designed from a long training sequence that is representative of all data, laser readings, to be encoded by the system. To generate the codebook, the Lynde-Buzo-Gray algorithm is used [17].

After the codebook is created, each vector S_t of the laser readings to be encoded is compared with each of the stored vectors C_i, and the vector S_t is coded by identifying the vector C_k that best represents S_t according to some distance measurement d.

$$d(S_t, C_k) = min(d(S_t, C_i)), \qquad i = 1, ..., L \qquad (1)$$

where L is the size of the codebook. The distance measurement used was the Euclidean distance.

3.1.2 HMM Model

The symbols are the indexes of a vector quantizer, for each laser reading we have L possible observations vectors that correspond to the L vectors of a quantizer. Then for a vector S_t the symbol O_t corresponds to the index of the centroid that best fits the laser readings, $O_t = k$.

And the probability of emitting symbol k being in state i, $b_i(k) = p(O_t = k|X_t = i)$, where $X_t = i$ means the Markov chain was in state i at time t, and $O_t = k$ means the output symbol at time t was k.

The transition probability defines the probability of taking the transition from a particular state to another state or itself. Where a_{ij} is the probability of taking a transition from state i to state j:

$$a_{ij} = p(X_{t+1} = j|X_t = i).$$

Also included is the initial state distribution probability:

$$\pi_i = P[x_1 = X_i], \qquad 1 < i < N$$

In the first order HMM, it is assumed that whether the Markov model will be in a particular state X at time $t + 1$ only depends on the present state at time t, and not on the past states.

$$P(X_{t+1} = x_{t+1}|x_i, ..., x_t)$$
$$= P(X_{t+1} = x_{t+1}|x_t).$$

Another assumption is that if a symbol will be emitted it only depends on the present state at time t, and not in the past states or emitted observations.

$$P(o_t = k|o_i, ..., o_{t-1}, x_i, ..., x_t)$$
$$= P(O_t = k|x_t).$$

3.2 Discrete Hidden Markov Models for Laser Readings

Each region R_j of the environment where the robot navigates is represented by a Markov model $\lambda_j = (A, B, \pi)$, where the matrices A and B are: $A = a_{ij}$ and $B = b_i(k)$ for all i, j and k, with N states and L observation symbols. The probabilities of each Markov model λ_j are found during training. For each region $j, 0 < j < N_{regions}$, the robot is set in region j with orientation of 0 degrees according to an global origin and then it follows the steps described in algorithm 1.

HMM states from 1 to 8, corresponds to the rotation process and from 9 to 16 to the navigation one, see figure 4. For each path that the robot takes in a closed region, see figure 5, we get the observations

Algorithm 1: Training Behavior for region R_j

Data:
Number of navigation steps NF
Training Vector Size TS
Result: Observations Vector $\vec{O}$
$n \leftarrow 0$
$aux \leftarrow 0$
for $n \leq TS$ **do**
  $\vec{O}_n \leftarrow$ Quantized(laser readings)
  if $aux > NF$ **then**
    The robot rotates 45°
    HMM state ← possible states (1,2,...,8)
    if $aux > NF + 8$ **then**
      $aux \leftarrow 0$
    end
  end
  else
    Robot in reactive navigation mode
    HMM state ← possible states (9,10,...,16)
  end
  $aux \leftarrow aux + 1$
  $n \leftarrow n + 1$
end

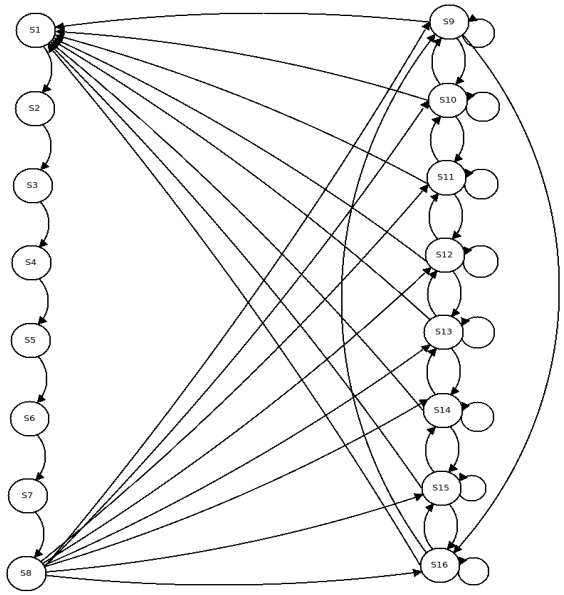

Figure 4. HMM for each region

$O^n(t_1), O^n(t_2), ..., O^n(t_i), ..., O^n(t_T)$, corresponding to the vectors of quantization that best fit each of the laser readings at t_i, where n corresponds to the nth-repetition.

For each laser reading, we have L possible observations that correspond to the L vectors of a quantizer. This quantizer was created using laser data during several paths that the robot followed. With the set of observations an estimation of the parameters of the HMM λ_j, that is A, B and π for each region j, was found using the BaumWelch algorithm.

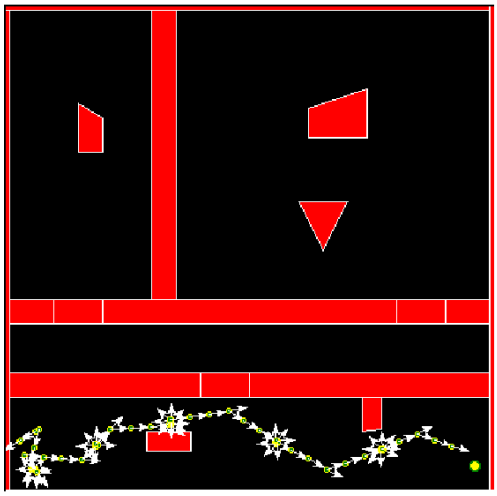

Figure 5. Training procedure for one region of the environment.

In the figure 6 we can see the complete HMM for the environment in figure 2. This HMM represents the concatenation of each region in the environment, where each node is the particular HMM λ_j for each region.

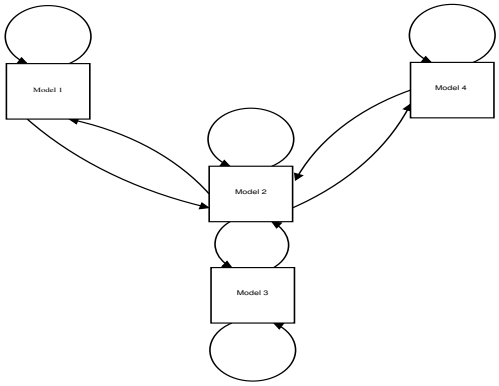

Figure 6. HMM generated from the concatenation of each model HMM of the regions shown in figure 2.

4 Robot Localization Using Discrete Hidden Markov Models

To recognize the region where the robot is and its orientation, first, the robot start rotating every 45 degrees, until it rotates 360 degrees, for each rotation it makes a laser reading, generating a vector of laser readings that is encoded through a vector quantizer of L vectors. This quantizer gives a set of observations

$$O = O(t_1), O(t_2), ..., O(t_i), ..., O(t_T),$$

that correspond to the VQ vectors that best fit the laser vector at time t during its rotation.

Then using the observation vector with all the HMM_j for each region, the probability of each model λ_j is found using the Forward algorithm [16], see figure 7. Then the robot is in $Region_j$:

$$Region_j = argmax_{1<i<N}[p(\lambda_i, O))], \quad (2)$$

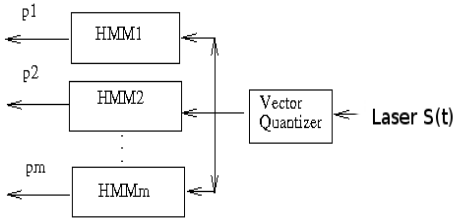

Figure 7. Probability evaluation for each region.

After it is found in which region is the robot as well its orientation the robot starts navigating to a defined destination, collecting again another observation vector O, and using this vector, the hidden variables, the states, are found using the Viterbi algorithm. This algorithm finds the best sequence of states in the network, and thus the best sequence of regions that the robot follows during its path. Figure 8 shows the best path followed by the robot going from one region to another.

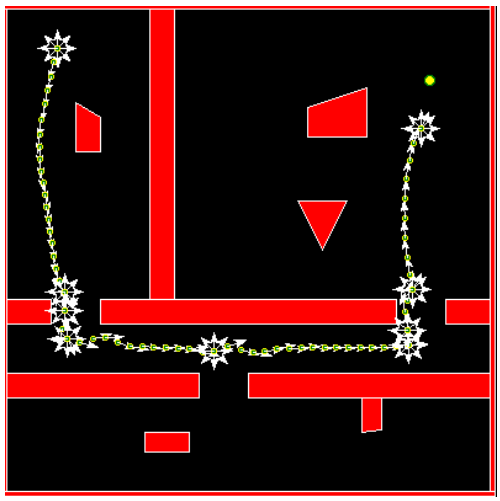

Figure 8. Best path followed by the robot.

The Viterbi algorithm is as follows [16]:

Given a HHM model $\lambda = (A, B, \pi)$ with N states and L observation symbols, and an observation sequence $O = O_1, O_2, ..., O_T$. The following equation gives the highest probability to reach state i along a single path at time t, given that the system has been in states $x_1...x_{t-1}, x_t$ and with observations $O_1, ..., O_t$:

$$\delta_t(i) = maxP[x_1...x_{t-1}, x_t = i, O_1, ..., O_t|\lambda].$$

In order to retrieve the state sequence, it is necessary for each t and j to keep track of the argument which maximize the previous equation. Using the array $\psi_t(j)$, the procedure is as follows:

2.1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \qquad 1 < i < N, \tag{3}$$

$$\psi_t(i) = 0,$$

2.2) Recursion:

$$\delta_t(i) = max_{1<i<N}[\delta_{t-1}(i)a_{ij}]b_j(O_t), \tag{4}$$

$$2 < t < T, \qquad 1 < j < N,$$

$$\psi_t(j) = argmax_{1<i<N}[\delta_{t-1}(i)aij], \tag{5}$$

$$2 < t < T, \qquad 1 < j < N.$$

2.3) Termination:

$$P^* = max_{1<i<N}[\delta_T(i)], \tag{6}$$

$$x_T^* = argmax_{1<i<N}[\delta_T(i)]. \tag{7}$$

2.4) State sequence backtracking:

$$x_t^* = \psi_{t+1}(q_{t+1}^*), \tag{8}$$

$$t = T-1, T-2, ..., 1.$$

Figure 9 shows the overall system to find the best sequence of regions that the robot visited.

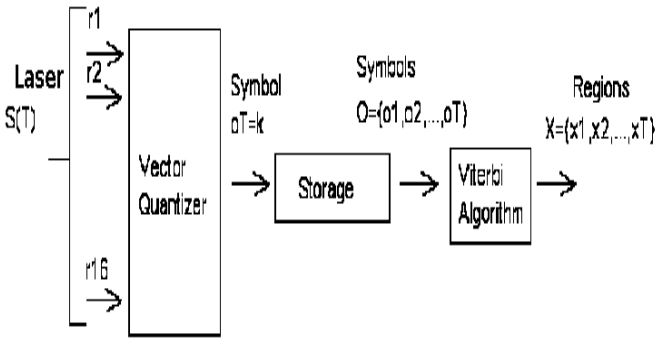

Figure 9. Overall System.

5 Experiments and Results

We tested the algorithm using the VIRBOT simulation module, in which laser readings can be simulated, see figure 3, as well as, with a TurtleBot robot, see figure 10. The simulator simulates laser readings by finding the distance of the objects in front of the robot, adding Gaussian noise to the values. The virtual mobile robot and the real one have a 180-degree laser, with a separation of 12 degrees each, making 15 laser readings in each sensing. The real robot uses a Hokuyo laser.

Figure 10. TurtleBot robot used in the experiments.

The algorithm was tested in an environment with 4 regions, shown in figure 2. The robot was put in any of the regions of each environment, and from this original position, it tried to reach a set of destinations using reactive behaviors. In figure 8, the robot starts in a particular region and to reach the goal destination, that is in another region, it needs to pass through the nodes that link the regions, when it reaches each of them it rotates to localize it shelf. With the symbols that were found using the vector quantizer, the start orientation of the robot, as well as, the best states were found on using the Viterbi algorithm. Table 1 shows the results with the robot's simulator, column $Tr.Size$ represents the number of vectors used to create the laser VQ; column $N.Symbols$ indicates the number L of centroids of the VQ, that is, the number of symbols of the HMM; column $N.Obs.$ indicates the number of observations T used in the Viterbi and Forward algorithms, and finally the column $Error$ shows the percentage of error for detecting in which region the robot is. As we can see the best performance was obtained with a training set of 8000 vectors, 512 observation symbols and using 64 observations each time the Forward and Viterbi algorithm was performed. From the estimated states we found 89% of the times the region where the simulated robot was, using only the laser readings without dead reckoning.

Tr.Size	*N. Symbols L*	*N. Obs. T*	*% Error*
4000	32	36	56%
8000	32	36	32%
4000	256	36	35%
8000	256	36	16%
4000	512	36	31%
8000	512	36	13%
4000	32	64	45%
8000	32	64	26%
4000	256	64	30%
8000	256	64	15%
4000	512	64	17%
8000	512	64	11%

Table 1. System performance with the simulated robot.

Table 2 shows the results with the TurtleBot robot. As we can see the best performance was obtained, as with the simulator, with a training set of 8000 vectors, 512 observation symbols and using 64 observations each time the Forward and Viterbi algorithm was performed. From the estimated states we found 85% of the times the region where the real robot was, using only the laser readings without dead reckoning.

Tr.Size	*N. Symbols L*	*N. Obs. T*	*% Error*
4000	256	36	36%
4000	512	36	29%
8000	256	64	24%
8000	512	64	15%

Table 2. System performance with the real robot.

6 Conclusions

We proposed a method to estimate the region where a robot is by using laser readings combined with Hidden Markov Models and the Viterbi algorithm. Using only laser reading without using dead reckoning, this technique was able to find the region where the robot was located. This approach worked well with our simulator, as well as, with a real mobile robot. The next step is to increase the number of regions, that is, to increase the accuracy of the estimated position.

References

[1] L. Contreras, W. Mayol-Cuevas, *O-POCO: Online POint cloud COmpression mapping for visual odometry and SLAM*, in *IEEE International Conference on Robotics and Automation* (????), p. 2017

[2] L.P. Berczi, T.D. Barfoot, *It's like deja vu all over again: Learning place dependent terrain assessment for visual teach and repeat*, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2016)

[3] M.W. M. Paton, K MacTavish, T.D. Barfoot, *Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat*, in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2016)

[4] D. Scaramuzza, F. Fraundorfer, IEEE Robotics and Automation Magazine 18(4) pp. 80–92 (2011)

[5] H. Durrant-Whyte, T. Bailey, IEEE Robotics and Automation Magazine 13(2) pp. 99–110 (2006)

[6] T. Bailey, H. Durrant-Whyte, IEEE Robotics and Automation Magazine 13(3) pp. 108–117 (2006)

[7] Thrun, J.J. Leonard, *Simultaneous Localization and Mapping* (Springer Handbook of Robotics, 2008)

[8] I.F.A.P.A.B.M.N.M.M. J. Savage, L. Contreras, C. Rivera, *Construction of Roadmaps Maps for Mobile Robots' Navigation Using RGB-D Cameras*, in *13th International Conference on Intelligent Autonomous Systems* (2014)

[9] L. Contreras, W. Mayol-Cuevas, *Towards CNN Map Compression for camera relocalisation*, in *arXiv* (2017)

[10] S. Shahriar, A. Zelinsky, *Mobile Robot Navigation based on localisation using Hidden Markov Models* (1999)

[11] S. Shahriar, A. Zelinsky, *Robot localization from minimalist inertial data using a Hidden Markov Model*, in *IEEE International Conference on Autonomous Robot Systems and Competitions* (2014)

[12] M.M. Jesus Savage, Marco Negrete, J. Cruz, *The Role of Robotics Competitions for the Development of Service Robots*, in *IJCAI, Workshop on "Autonomous Mobile Service Robots"* (2016)

[13] J. Muller, *The Design of Intelligent Agents: a layered approach* (Springer-Verlag, 1996)

[14] R. Arkin, R. Murphy, IEEE Transaction on Robotics and Automation 6(4) pp. 445–452 (1990)

[15] S. Thrun, Technical Report, CMU-CS-96-122, School of Computer Science, Carnegie Mellon University. (1996)

[16] L. Rabiner, Readings in Speech Recognition Morgan Kaufmann Publishers pp. 267–296 (1990)

[17] A.B. Y. Linde, R.M. Gray, IEEE Transactions on Communications, 84-95 (1980)