Robotica (2009) volume 27, pp. 1049–1061. © Cambridge University Press 2009
doi:10.1017/S0263574709005463

Automatic training method applied to a WiFi+ultrasound POMDP navigation system

M. Ocaña*, L. M. Bergasa, M. A. Sotelo, R. Flores, D. F. Llorca and D. Schleicher

Department of Electronics, Escuela Politécnica Superior, University of Alcalá, Campus Universitario s/n, 28871 Alcalá de Henares, Madrid, Spain.

(Received in Final Form: February 9, 2009. First published online: March 9, 2009)

SUMMARY

This paper presents an automatic training method based on the Baum–Welch algorithm (also known as EM algorithm) and a robust low-level controller. The method has been applied to the indoor autonomous navigation of a surveillance robot that utilizes a WiFi+Ultrasound Partially Observable Markov Decision Process (POMDP). This method uses a robust local navigation system to automatically provide some WiFi+Ultrasound maps. These maps could be employed within probabilistic global robot localization systems. These systems use *a priori* probabilistic map in order to estimate the global robot position. The method has been tested in a real environment using two commercial Pioneer 2AT robotic platforms in the premises of the Department of Electronics at the University of Alcalá. Some experimental results and conclusions are presented.

KEYWORDS: WiFi+Ultrasound robot navigation system; WiFi signal strength localization system; Partially Observable Markov Decision Process.

1. Introduction

For an indoor navigation system design, in which the main goal is the guidance of a robot to a destination room, the topological discretization of the environment is an appropriate representation to ease planning and learning tasks.[1] In a topological discretization, the environment is divided into some prior known nodes. For this approach, Partially Observable Markov Decision Process (POMDP) models provide solutions to localization, planning, and learning tasks. These models use probabilistic reasoning to deal with uncertainties, which is an essential feature in the case of WiFi (Wireless-Fidelity) localization sensors. The robot needs a low-level controller to move across the nodes and to perform local navigation.[2] The low-level controller allows the robot to reach the next node with the lowest positioning error, in such a way that observations are obtained with minimum error.

Over the last few years the interest in wireless networks has increased, and a large number of available mobile tools as well as other emerging applications are becoming more and more sophisticated. Wireless networks have become a critical component of the networking infrastructure and they are available in most corporate environments (universities, airports, train stations, tribunals, hospitals, etc.) and in many commercial buildings (cafes, restaurants, cinemas, shopping centers, etc.).

Many mobile robot platforms use wireless networking to communicate with off-line computing resources, human–machine interfaces, or others robots, since the advent of inexpensive wireless technology. These platforms usually have been equipped with 802.11b/g wireless Ethernet, thus having a cheap sensor from which position can be directly inferred without the computational overhead required by image processing or the expensive solution provided by laser systems. Many robotics applications would benefit from being able to use wireless Ethernet for both sensing position and communication without having to add new sensors in the environment.

The recent interest in location sensing and the rising demand on the deployment of such systems has made network researchers face a well-known problem in the field of robotics: localization. Finding a robot pose (position and orientation) from physical sensors is not a trivial problem. In fact, it is often referred to as "the most important problem to provide a mobile robot with autonomous capabilities".[3] Several systems for localization have been proposed and successfully developed for indoor environments. These systems are based on: infrared sensors,[4] computer vision,[5] ultrasonic sensors,[6] laser,[7] or radio frequency (RF).[8–13] Within the last group we can find the localization systems based on WiFi signal strength measure. WiFi location determination systems use the popular 802.11b/g network infrastructure to determine the device location without using any extra hardware. It makes these systems attractive for indoor environments where traditional techniques, such as Global Positioning System (GPS),[14] fail.

In order to estimate the robot global position, wireless Ethernet devices measure signal strength of received packets. Signal strength is a function of the distance and obstacles between wireless nodes and the robot. In practice, the system needs different radio paths coming from several base stations, or Access Points (APs), to measure the distance between the robot and the APs.

* Corresponding author. E-mail: mocana@depeca.uah.es

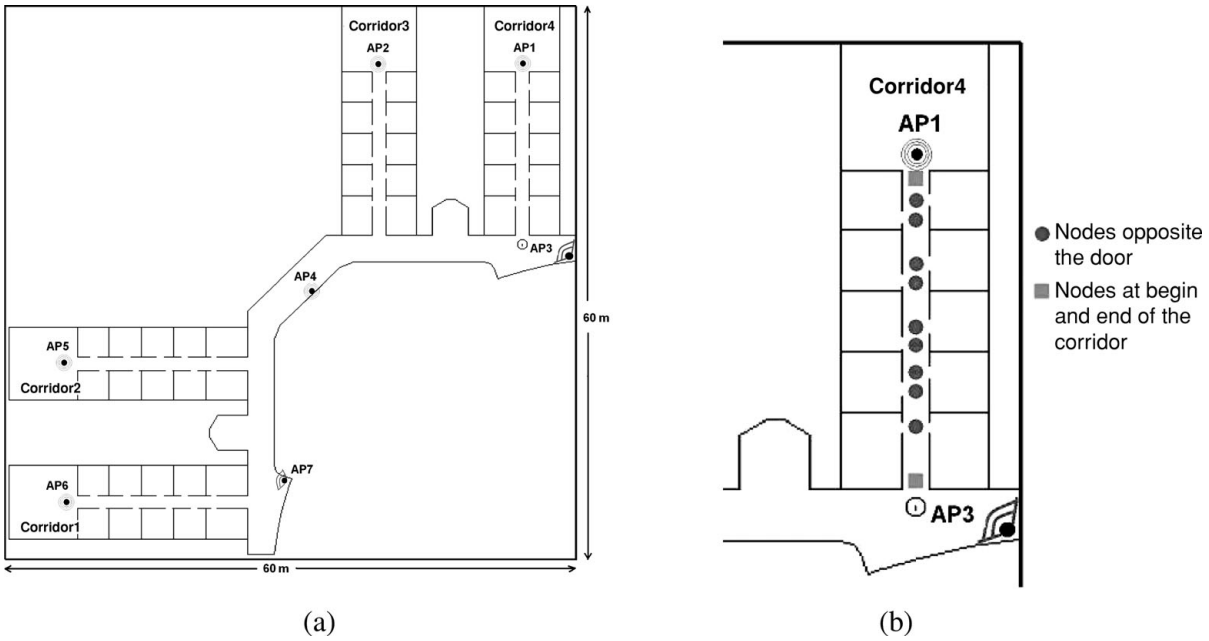


Fig. 1. (a) Third Floor of the UAH Electronics Department and (b) detailed view of nodes in corridor 4.

Unfortunately, a triangulation algorithm is not usually applied to infer the estimated position because wireless channel is very noisy in indoor environments and RF signal can suffer from reflection, diffraction, and multipath effect, which makes the signal strength a complex function of distance.[10] To solve this problem, several WiFi location determination systems use *a priori* radio map (wireless-map) that captures the signature of each AP at certain points in the area of interest. These systems work in two phases: training and estimation. During the training phase, the system constructs the WiFi map in a previous set-up, normally performed in manual mode or using an approximated propagation model.[15] In the estimation phase, the vector of samples received from each AP is compared to the wireless-map and the "nearest" match is returned as the estimated user location. The problem in the manual mode is that this method involves an enormous calibration effort because the observations are normally obtained in a manual way while in the approximated propagation model the error is larger than that in the former. In this paper we propose an automatic method to obtain the wireless map based on a robust local navigation system and the Baum–Welch algorithm. We compare our method with two other training methods existing in the literature.

The rest of the paper is organized in the following sections: Section 2 provides a brief description of the designed WiFi+Ultrasound POMDP Navigation System for indoor environments, as well as the system implementation and a description of our test bed. Section 3 describes the automatic training method developed in this work. Section 4 shows the experimental results. Finally, the conclusions and future work are described in Section 5.

2. WiFi+Ultrasound POMDP Navigation System

In this section we present the basic concepts of our WiFi+Ultrasound POMDP Navigation System, as well as a description of the test bed and the system implementation in order to identify the main problems derived from the use of WiFi sensors in the localization stage of a navigation system.

2.1. Test bed

The test-bed environment was established on the third floor of the Polytechnic School building, concretely in the Electronics Department at the University of Alcalá. The layout of this zone is shown in Fig. 1(a). It has a surface of 60 m $\times$ 60 m, with some 50 different rooms, including offices, labs, bathrooms, storerooms, and meeting rooms.

It is important to provide a complete WiFi coverage to the overall environment. We installed the APs to obtain a measure of at least three APs at any place of the environment. Therefore, seven Buffalo APs (WBRE-54G) were installed at the locations indicated in Fig. 1(a). Five APs were connected to omnidirectional antennas and two APs (AP3 and AP7) were connected to antennas of 120° of horizontal beam-width to achieve a maximum coverage. Then, we obtained a low error percentage in the localization stage in a previous work.[16]

We have tested our method in two corridors for simplifying the process. It can be noted that the environment is symmetric from the main diagonal. Thus, the obtained conclusions can be extrapolated to all corridors. Results presented in this paper were carried out at the third and fourth corridors in order to build *a priori* WiFi+Ultrasound map using the automatic training method that can then be used for navigation tasks. The environment was discretized into coarse-grained regions called nodes. At these corridors, 11 nodes were defined at the positions indicated in Fig. 1(b). We select the nodes in front of the offices, doors, at the beginning and at the end of corridors because these are the minimum necessary positions in order to plan some missions for guiding the robot from one room to another with the final goal of performing surveillance tasks.

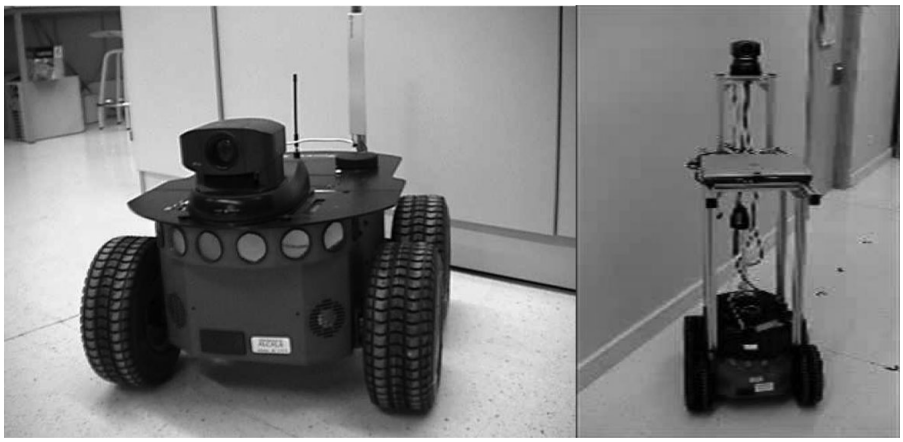


Fig. 2. Real robots Pioneer 2AT where the automatic training method has been tested.

Figure 2 depicts the real robotic platforms used for testing the proposed technique: two commercial Pioneer 2AT robots equipped with an embedded computer with an 850 MHz Celeron processor, WiFi interface and external antenna. The two robots have the same 2AT platform, but we have added an aluminum structure in one robot to carry a laptop and to increase the height of the camera for improving human–machine interface capabilities.

2.2. Main WiFi signal variations

This section shows a brief description of the main variations suffered by WiFi signal in indoor environments. These are useful for understanding the training algorithm. In ref. [21], authors identified the following three main causes for the variation of the signal strength in an indoor environment:

(1) Temporal variations: variations standing at a fixed position for a long time.
(2) Large-scale variations: the signal strength varies over a long distance due to attenuation.
(3) Small-scale variations: these variations happen when the robot moves over a small distance and they are due to the signal wavelength (at 2.4 GHz the wavelength λ is 12.5 cm, then, this effect will appear for distances below 12.5 cm).

We identified two additional variations for the signal strength measure that are directly related to the robot localization:

(4) Large orientation variations: these are the variations suffered by the WiFi signal when the robot is located at a node in the four basic orientations (North–South–East–West) respecting the longitudinal direction of the corridor.
(5) Small orientation variations: the signal strength varies when the robot is located at a node with slight angle differences with respect to the reference orientation (parallel to the walls). The small orientation variations are directly related with the small-scale variations, but in this application it is necessary to know what the robot positioning constraints are in order to achieve a robust localization system.

In ref. [2], we performed different tests to get these variations in our environment. We concluded that the WiFi signal strength measure is quite stable when there are not people in the environment. We demonstrated the pernicious effect of people moving and other wireless devices (Bluetooth keyboards, and mice), in the environment during the operation time.

We measured the large-scale variations and we concluded that the average signal strength was not linear with the distance due to the multipath effect. That is the reason why it is very difficult to build a propagation model for indoor environments and one of the main motivations of this work.

For demonstrating the small-scale variations we took several measures from all APs at different nodes moving the robot parallel to the walls in short distances ($<\lambda$). Slight variations were measured in a distance smaller than the half of the wavelength ($\lambda/2$) while the highest variations were measured in distances from $\lambda/2$ to $3\lambda/4$.

We also analyzed the robot orientation effect in our environment. We conclude that the histogram profiles were quite different for the basic orientations at each node, having a remarkable difference in the average signal.

For measuring the small orientation variations we placed the robot at several nodes of the environment. After that, slight variations were measured in orientations smaller than 9°.

We concluded that while the large scale and orientation variations are useful to localize the robot; the small scale and orientation ones, as well as the presence of people in the environment, can be pernicious for this purpose. Then, it is important to avoid small effects by accurate robot placing in the environment.

2.3. Description of the POMDP

A Markov Decision Process (MDP) is a model for sequential decision making. It is defined by a tuple $\{S, A, T, R\}$, where:

- S is a finite set of states ($s \in S$).
- A is a finite set of actions ($a \in A$).
- T is a state transition model which specifies a conditional probability distribution of posterior state s_{t+1} given prior state s_t and action executed $a_t (T = p(s_{t+1}|s_t, a_t))$.
- R is the reward function that determines the immediate utility of executing action a at state s, $(r(s, a) \forall (s \in S, a \in A))$.

POMDPs[1,17–19] are mathematical models that are defined by the same elements of a MDP as well as by the following ones:

- O is a finite set of observations ($o \in O$).
- ϑ is the observation model which specifies a conditional probability distribution over observations given the actual state $s(p(o|s) \; \forall \; (o \in O, s \in S))$.

As we have just explained, the environment is divided into certain discrete positions or nodes $s \in S$. These nodes are coarse-grained regions of variable size in accordance with the environment topology and its centers are separated by more than λ (in our case 80 cm). This simplification yields a reduction in the computation time and it is appropriate to minimize the WiFi small-scale effect.

The action set A has been selected to establish correspondences between transitions from one state to another and local navigation behaviors of the robot. We assume imperfect actions. Thus, the effect of an action can be different from the expected one (this is modeled by the transition matrix T).

The action set is very simple in our application due to the configuration of the states and the local navigation system. The action set is composed by the following actions: Follow corridor (a_{FC}), No operation (a_{NO}), Turn around (a_{TA}), Turn right (a_{TR}), and Turn left (a_{TL}). It is carried out by the local navigation system. Local navigation in corridors is based on ultrasound range measurements. The low-level controller keeps the robot on the center of the corridor while navigating, indicating to the POMDP the occurrence of a transition, such as a door detected or a corridor ending.[2]

We have selected two kinds of observations in our model: the WiFi signal strength measure observation ($o_{\mathrm{AP}n}$) and the ultrasound observation (o_{US}). The $o_{\mathrm{AP}n}$ is obtained as the average value of several samples of the signal strength, received in the WiFi robot interface from all APs, in order to minimize the high noise of the WiFi signal measures. Therefore, the $o_{\mathrm{AP}n}$ is divided into N different observations ($o_{\mathrm{AP1}}, o_{\mathrm{AP2}}, \ldots, o_{\mathrm{AP}N}$), where $n \in [1, \ldots, N]$ and N is the number of Aps in the environment.

The o_{US} is obtained from the ultrasound sensors. Four different observations are established: door on the left, door on the right, door on both sides, and wall on both sides. This way, the possible values are discrete and useful to index the observation. This observation, obtained from the ultrasound sensors, leads us to robust local observations that can be fused with global WiFi observations using Eq. (1).

The two kinds of observations ($o_{\mathrm{AP}n}$ and o_{US}) are complementary. The first one obtains an estimation of the global localization while the second one obtains a precise estimation of the local environment. The fusion of these observations will produce a good observability of states. POMDP provides a natural way for using multisensorial fusion in their observation models ($p(\vec{o}|s)$) by means of Bayes rule. For simplicity, we will assume that the observations are independent, then, the observation model can be simplified in the following way:

$$\begin{aligned} p(\vec{o}|s) &= p(o_{\mathrm{AP1}}, \ldots, o_{\mathrm{AP}N}, o_{\mathrm{US}}|s) \\ &= p(o_{\mathrm{AP1}}|s) \cdot \ldots \cdot p(o_{\mathrm{AP}N}|s) \cdot p(o_{\mathrm{US}}|s). \end{aligned} \quad (1)$$

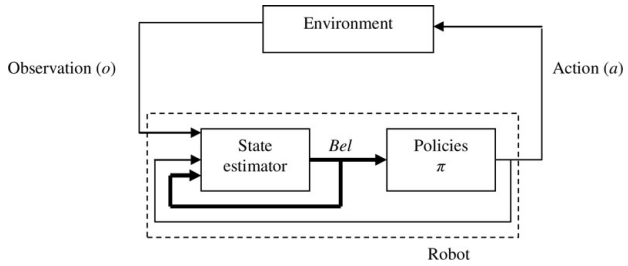


Fig. 3. POMDP structure.

In Eq. (1) $\vec{o}$ is a vector composed of the two kinds of observations: $o_{\mathrm{AP}n}$ (provided by the WiFi signal strength) and o_{US} (provided by the ultrasound sensors).

The observation's uncertainty model represents the real errors or failures of the sensor systems (ultrasound ring and WiFi interface). The observation's function ϑ incorporates this information to the POMDP. In this work, ϑ is a matrix for each observation (N WiFi observations: $\vartheta_{\mathrm{AP1}}, \ldots, \vartheta_{\mathrm{AP}N}$, and one for ultrasound observations: ϑ_{US}). The matrix dimensions are "$n_{\mathrm{S}} \times n_{\mathrm{O}}$", where n_{S} is the total number of states in the environment and n_{O} is the possible observation values in the current state.

In many real systems using POMDPs, the values of the transition (T) and observation matrices ($\vartheta_{\mathrm{AP}n}, \vartheta_{\mathrm{US}}$) are obtained with a simple deduction or using *a priori* expertise. In our case, we use the ability of our low-level controller and the Baum–Welch algorithm to build an autonomous training system, as we will describe in next section.

A POMDP does not provide a real state due to the observations uncertainty. A POMDP maintains a belief distribution called *Bel*(S) or *Belief Distribution* (*Bel*) over the states to solve. This distribution assigns a probability to a state s that indicates the possibility of being in the real state. This is the main reason to divide the control stage of a POMDP in the following two blocks, as can be seen in Fig. 3:

(1) *State estimator*: the input of this block is the current observations and its output is the *Bel* function. This block obtains the probability over all possible states.
(2) *Policies*: the input of this block is the current *Bel* and its output is the action to perform. This block obtains the optimal action to perform in the next execution step to maximize the reward function (R). This function determines the immediate utility of executing action a at state s.

The Belief Distribution must be updated whenever a new action or observation is carried out. When an action is executed and a new observation is taken, the new probabilities are obtained using Eq. (2).

$$Bel_{t+1}(s_{t+1}) = \eta \times p(o|s_{t+1}) \sum_{s \in S} p(s_{t+1}|s_t, a_t) \times Bel_t(s_t), \quad \forall s_{t+1} \in S. \quad (2)$$

There are different algorithms to solve the selection of the ideal action to execute in each state. In a POMDP the problem is more complex than in a MDP because the current state is unknown. Only a belief distribution is maintained in a

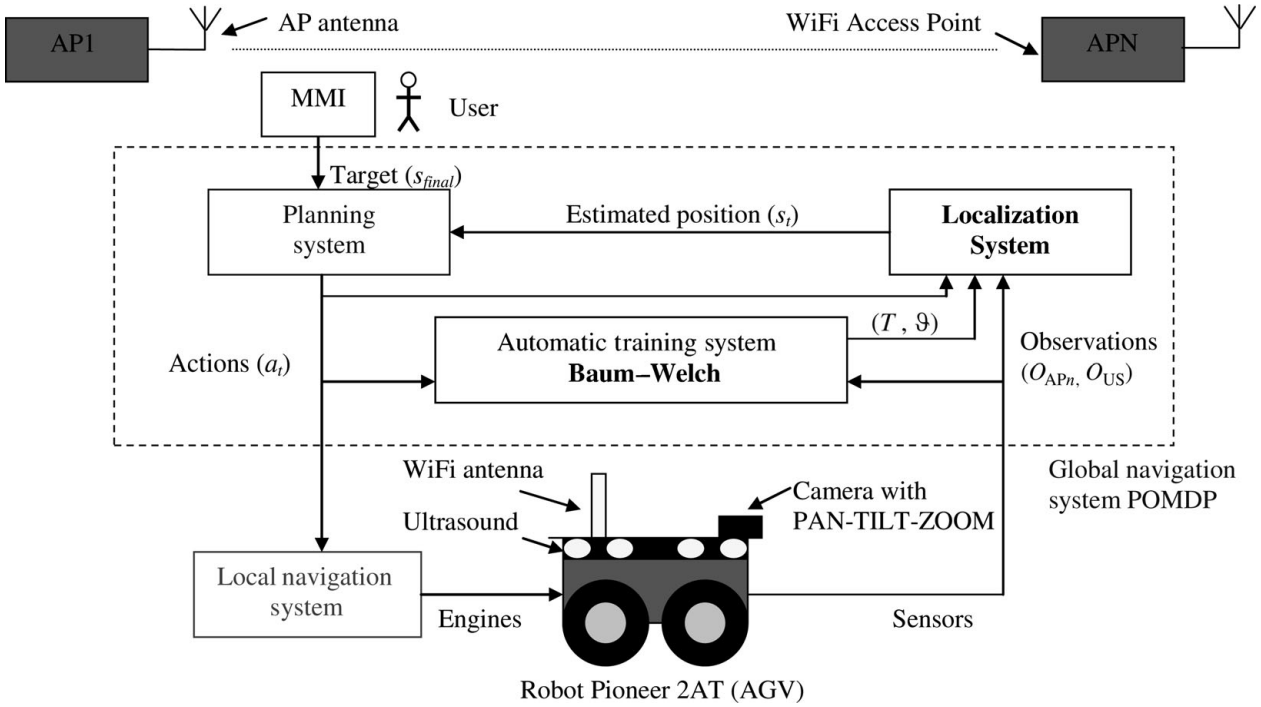


Fig. 4. Description of our WiFi POMDP navigation system.

POMDP. In this work we assume a simpler method, called *Most Likely State* (*MLS*), to select the optimal action, because the global observation provided by the WiFi sensor normally obtains a belief distribution that exhibits a maximum at the real state. This method selects the action associated to the most probable state of the Belief Distribution as shown in Eq. (3).

$$a = \pi_{\mathrm{MLS}}(Bel) = \pi * \left(\arg \max_{s}(s) \right). \tag{3}$$

2.4. Architecture of our WiFi+Ultrasound POMDP navigation system

The architecture of the global WiFi+Ultrasound POMDP navigation system that we have designed is shown in Fig. 4. The main blocks are as follows:

- The robot sensors provide two kinds of observations ($o_{\mathrm{AP}n}$ and o_{US}) and they are the inputs of the training block and localization system. The actions commanded by the planner are executed through the local navigation system by the actuators of the robot (the four engines connected two by two).
- The localization system uses the observations provided by the robot and the priori map to obtain the estimated position over all the states (*Bel*).
- The planning system has two inputs, the *Bel* over all the states and the commanded state, that is introduced by the final user. The commanded action is used as input to the training system and the local navigation system.
- The automatic training system is used to lead the WiFi observation matrix ($\vartheta_{\mathrm{AP}n}$), the ultrasound observation matrix (ϑ_{US}), and the POMDP transition matrix (T) from the training data set. These matrices will be used in the localization stage together with the observations (o_t) and actions (a_t).
- The Man–Machine Interface (MMI) is based on a friendly graphical interface built in GTK, in a typical client–server application. This interface works under Linux through a TCP-IP connection to the robot server (in our case Saphira[20]).
- The local navigation system is in charge of executing the actions commanded by the planning system.

3. Automatic Training Method

The existing WiFi training map methods involve an enormous calibration effort because, until now, observations at each node are manually carried out[10,11] or using a robot in teleoperated mode. To solve this problem, we propose an automatic WiFi training map method based on an autonomous and robust local navigation task and the Baum–Welch algorithm.

The autonomous local navigation task guides the robot along the center of the corridors and it is able to stop in the center of the nodes that are found by the robot in its route, using only ultrasound sensors. This task needs to know only the width of doors and corridors to navigate along the center of it and to detect doors on both sides (see previous author's work[2]).

The method provides the WiFi ($\vartheta_{\mathrm{AP1}}, \ldots, \vartheta_{\mathrm{AP}N}$) and Ultrasound observation matrix (ϑ_{US}), also called WiFi and US map. It provides one WiFi map for each AP of the environment and one US map. We will denote these matrices like ϑ, because the explanation is the same for all of them. In addition, the method provides the transition matrix T that will be used in the WiFi POMDP navigation system.

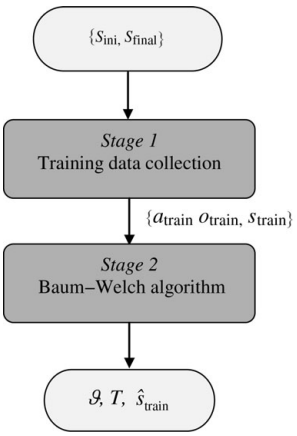


Fig. 5. Automatic training method.

The automatic training method, shown in Fig. 5, consists of following two different stages:

- In the first one, the training data is collected. The inputs of this stage are the start (s_{ini}) and target (s_{final}) states. This stage provides the executed actions a_{train} during the trail, the observations at each node o_{train}, and the visited states s_{train}.
- In the second stage, the Baum–Welch algorithm is executed over the stored data in order to yield the transition T and observation matrices ϑ. In addition, the algorithm provides the training estimated states $\hat{s}_{train}$ that will be compared to the visited states in order to obtain the localization error in the training stage. In this stage we use a novel setup process to minimize the convergence time. For that purpose we use a WiFi propagation model to find the possible convergence values within the WiFi observation matrices.

3.1. Stage 1: training data collection

For this first stage, based on the robustness of our low-level controller and after checking that the ultrasound measures have low error percentage in robot's local positioning, we assume that the POMDP works like a MDP using only local US observations (o_{US}). This means that the actions and observations have no uncertainty and that the robot knows where it is at each execution step. Then, we only need to provide the start s_{ini} and target s_{final} nodes to the MDP to achieve an automatic journey of the robot. If we chained several targets for the MDP, the process gives several training frames of actions and observations that represent the inputs for the second stage.

During this stage the robot travels through the environment from a known start node s_{ini} to another target node s_{final}. It travels in autonomous mode using only ultrasound information and executing local autonomous motion actions (a_t), and halting at the center of the nodes (s_t), where t represents the execution step. At each node, the robot stores the actions a_{train}, the observations of the WiFi signal strength and Ultrasound measures o_{train}, and the visited nodes s_{train}. The stored observations at each node and the executed actions through the different trials represent the training data set. These data constitute the inputs for the Baum–Welch algorithm that will be executed in the next stage. The visited nodes s_{train} will be used to evaluate the algorithm, and will be referred to real states.

The local navigation system that we use in this stage has following three main goals:

1. To execute the action commanded from the MDP.
2. To inform the MDP when a state transition is detected.
3. To place the robot in the optimal location to measure the WiFi signal.

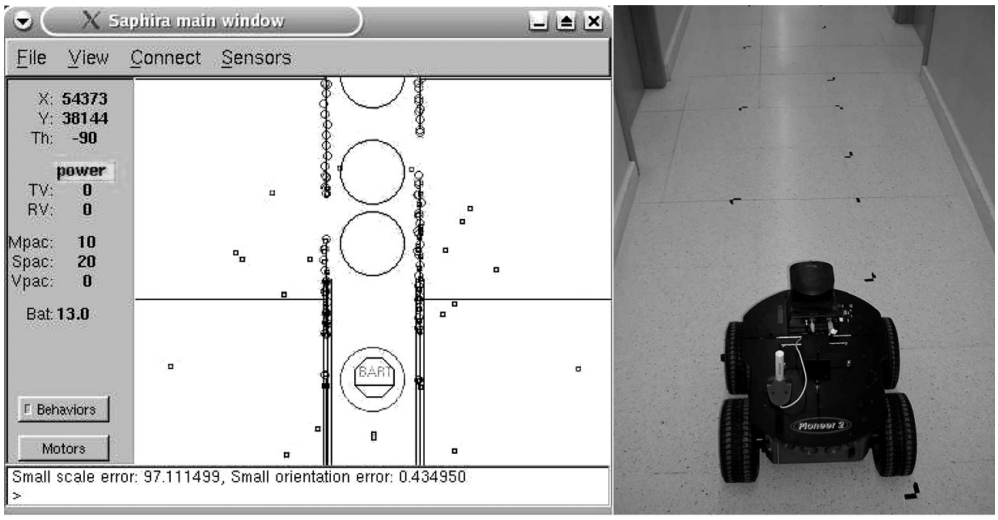
To carry out the state transition detection, the low-level controller uses only ultrasound range measurements. Two kinds of transitions can be detected: door detection transition and corridor ending transition. Using a door detection transition procedure instead of relying on dead reckoning more robust performance can be achieved, as demonstrated in practice, especially for large corridors. Door detection accuracy can be highly improved by using an H-shape model of corridor (see Section 4.1 in ref. [2]). Thus, deviations from the model can be regarded as doors. In more complex environments, when the corridor does not adjust to an H-shape model, it will be needed to use some others appropriate geometrical models.

The local navigation system is able to navigate the robot in a corridor, whatever the state of its doors and the position of persons walking around it. When the robot enters a corridor it needs to get the appropriate orientation in parallel with respect to the corridor walls. Then, using the H-shape model of corridor obtained from ultrasound range measurements, the robot gets the lateral and orientation error with regard to the center of the corridor.

In many buildings, like in our environment, the doors are depressed from the wall. For example, in our building the doors are depressed 8 cm from the wall, as can be seen in Fig. 6(b). The robot relies on this characteristic in a validation process in order to add a new measure in the model. This prevents perturbations in the environment. The distance between the new measured points and the previous corridor model must be below a validation distance d_v in order to validate the measurement and to include it in the estimation process. By using this process, door detection in the corridor walls turns out to be a simple task. Considering a $\pm\ d_v$ validation band, every door (either open, close, or ajar) will be detected as an open gap in the wall, as depicted in Fig. 6(a). In more complex environments, in which the doors are not depressed, it is possible to use a different door detection method. For example, a method based on visual information could be used (see previous author's work[1]).

When the robot reaches the end of the corridor a state transition is issued by detecting an obstacle in front of the robot for a long time (more than 1 min). This is a simple approximation but in practice it is demonstrated to be a robust solution. It only fails when some people or some objects stop in front of the robot for more than 1 min. Figure 6(a) depicts the robot control and door detection software working and Fig. 6(b) depicts the real robot at the corridor estimating the H-shape model.

In order to minimize the WiFi measure error, each AP observation is obtained using the mean value of several consecutive samples measured in the WiFi interface. The mean of the observed values are rounded off to integers



(a) (b)

Fig. 6. (a) Robot control software and (b) real robot at the corridor.

from 0 to 99. These will be used as matrix index. These values correspond to the 0 to −99 dBm measure range in the WiFi interface. Equation (4) shows the observation (o_{train}) and action (a_{train}) training set during n_T execution steps. Training data set is formed by the tuple $\{o_{\text{train}}, a_{\text{train}}\}$.

$$\begin{aligned} o_{\text{train}} &= \{o_1, \ldots, o_{n_T}\} \\ &= \{(o_{\text{AP1}}, \ldots, o_{\text{AP}N}, o_{\text{US}})_1, \ldots, \\ &\quad \times (o_{\text{AP1}}, \ldots, o_{\text{AP}N}, o_{\text{US}})_{n_T}\} \\ a_{\text{train}} &= \{a_1, \ldots, a_{n_T}\} \end{aligned} \tag{4}$$

3.2. Second stage: Baum–Welch algorithm

In the second stage, the Baum–Welch algorithm takes the training data set ($\{a_{\text{train}}, o_{\text{train}}\}$) and obtains the WiFi and US map ϑ for the different states or nodes. The map is formed by a matrix with "$n_{\text{S}} \times n_{\text{O}}$" dimensions, where n_{S} represents the total number of states or nodes and n_{O} is the total number of possible observation values, i.e. 100 values for WiFi observations and 4 for US observation. Figure 7 shows an example of the WiFi map or WiFi observation matrix $\vartheta_{\text{AP}n}$ for the Access Point n.

The algorithm used in this second stage is a particularization of the Baum–Welch or EM (Expectation–Maximization) algorithm, using WiFi and US observations and a novel setup process for the WiFi observation. The EM algorithm is a hill-climbing process that iteratively alternates two steps. The E-step (expectation step) calculates the state evolution. It estimates the robot localizations based on the currently available map parameters. The M-step (maximization step) uses the estimated trajectory, computed in the E-step, to recalculate the map ϑ in order to obtain the maximum likelihood parameters. The greater the improvement of the map, the easier it is to estimate the evolution of the states.

The E-step re-estimates, at each iteration, the robot trajectory, using a belief distribution *Bel* over all states, the training data set and the available map ϑ. The *Bel* parameter is carried out using two distributions: α_t(s) and β_t(s), as shown in Eq. (5).

$$\begin{aligned} Bel_t(s_t|o_{\text{train}}, a_{\text{train}}, \vartheta) &= \eta \cdot \underbrace{p(s_t|o_1, a_1, \ldots, o_t, \vartheta)}_{\alpha_t(s)} \\ &\cdot \underbrace{p(s_t|a_t, \ldots, o_{n_T}, a_{n_T}, \vartheta)}_{\beta_t(s)} \qquad \forall s_t \in S, \end{aligned} \tag{5}$$

where η is a normalization factor. The α_t(s) distribution is the probability of reaching a state when the robot has executed several actions and has moved through some several states. This distribution can be obtained by forward computation, as

Observation value / State	0	1	2	…	97	98	99
s_0	0.1	0.1	0.3	…	0.01	0.01	0.01
s_1	0.01	0.1	0.2	…	0.01	0.01	0.01
…	…	…	…	…	…	…	…
s_{ns-2}	0.02	0.1	0.3	…	0.01	0.01	0.01
s_{ns-1}	0.01	0.01	0.02	…	0.01	0.01	0.01

Fig. 7. Example of WiFi observation matrix $\vartheta_{\text{AP}n}$ generated with the automatic method.

expressed in Eq. (6).

$$\alpha_t(s) = p(s = s_t|o_1, a_1, \ldots, o_t) = \eta \cdot p(o_t|s_t) \cdot \sum_{\forall s \in S} p(s_t|s_{t-1}, a_{t-1}) \cdot \alpha_{t-1}(s) \quad \forall s_t \in S. \quad (6)$$

The β_t(s) distribution represents the probability of traveling through some several states when the robot has started at certain state and it has executed several actions. This distribution can be obtained by backward computation, as in Eq. (7).

$$\beta_t(s) = p\big(s_t = s|a_t, \ldots, o_{n_T}\big) = \eta \cdot \sum_{\forall s_t \in S} p(s_{t+1}|s_t, a_t) \cdot p(o_t|s_t) \cdot \beta_{t+1}(s_{t+1}) \quad \forall s_t \in S. \quad (7)$$

Finally, γ_t(s) distribution represents the best state estimation and it is computed as the product of $\alpha_t(s)$ and $\beta_t(s)$ distributions, as in Eq. (8).

$$\gamma_t(s) = \alpha_t(s) \cdot \beta_t(s). \quad (8)$$

The M-step re-adjusts the radio map parameters according to the previous map and the states evolution estimated in the E-step. This adjustment is carried out using a frequency count as shown in Eq. (9).

$$p'(o|s) = \frac{\sum_{t=1,\ldots,n_T|o_t=o} \gamma_t(s)}{\sum_{t=1,\ldots,n_T} \gamma_t(s)} \quad \forall s \in S. \quad (9)$$

In this step, the initial distribution is adjusted using Eq. (10) while the transition matrix is computed using Eq. (11). The transition matrix T defines the probability of reaching a state s_{t+1} when the prior state is s_t and the robot executes an action a_t.

$$p(s_0 = s) = \gamma_0(s) \quad (10)$$

$$p(s_{t+1}|s_t, a_t) = \frac{\sum_{t=1,\ldots,n_T-1|a_t=a} \gamma_t(s_t, s_{t+1})}{\sum_{t=1,\ldots,n_T-1|a_t=a} \gamma_t(s)} \quad \forall s_t, s_{t+1} \in S \quad \forall a \in A \quad (11)$$

One of the main parameters to determine in this training method is the optimal number of frames necessary to obtain a correct observation matrix. To determine this parameter we use the entropy and the divergence factor proposed by the authors in ref. [18] over the $\gamma_t(s)$ distribution, taking it like a training belief distribution (*Bel*). The entropy determines the uncertainty degree or scatter of the different states within the *Bel*, using the expression shown in Eq. (12).

$$H(Bel) = -\sum_{Bel(s)\neq 0} Bel(s) \cdot \log(Bel(s)). \quad (12)$$

The divergence factor determines the uncertainty degree of the *Bel* using Eq. (13).

$$\tilde{D} = 1 - \frac{(d_{\max} + p_{\max}) \cdot n_S - 1}{2 \cdot n_S - 1}, \quad (13)$$

where $d_{\max}$ is the difference between the first and the second maximum value in the *Bel* and $p_{\max}$ is the absolute value of the first maximum in the *Bel*.

The divergence factor will be zero if there is no uncertainty in *Bel*. It means that there is a state with the maximum probability. If there is a maximum uncertainty, the probability of all states is $1/n_S$, and the divergence factor will be one. Therefore, when entropy or divergence factor are close to zero, we will know that the training is about to finish and then we will select an optimal number of frames.

In this kind of algorithms there are some problems with the parameter initialization setup, especially when the WiFi observation matrices can take values from 0 to 99. Convergence time can be higher and even the algorithm might not converge. We have introduced an important improvement to avoid this problem. Initialization of the WiFi observation matrices will be carried out using a coarse radio propagation model.

WiFi radio signal propagates through the air following a radio propagation model. This model is very difficult to obtain for indoor environments due to the multipath effect and the temporal variations of the WiFi signal, as previously mentioned. Although an exact and general model does not exist, an approximated model can be used to initialize the WiFi observation matrices. Then, a reduced searching range can be used in the surroundings of the expected value calculated by the model. This searching range must be computed according to the model deviation with respect to the real propagation. In our case we use a generic log distance model[8] to obtain the expected value, as shown in Eq. (14).

$$\mathrm{RSL} = \mathrm{TSL} + G_{\mathrm{TX}} + G_{\mathrm{RX}} + 20\log(\lambda) - 20\log(4\pi) - 10 \cdot n_W \cdot \log(d) - X_a, \quad (14)$$

where RSL is the received signal level, TSL is the transmitted signal level, G_{TX} and G_{RX} are the transmitter and receiver antennas gain respectively, λ is the wavelength, n_W is a factor that depends on the walls effect, X_a is a normal random variable with a deviation, and d is the distance between the emitter and receiver. Figure 8 shows an example about how to calculate the distance at a certain node with respect to four APs. It is important to note that APs are placed in the ceiling of the environment while the WiFi antenna of the robot is close to the floor.

Once we have obtained the expected RSL value for the node, we extend the searching range (Δ_{search}) to the left and right of this value in the observation matrix. We select a searching range (Δ_{search}) according to the model deviation with respect to the real propagation. Then we use a uniform probability distribution over the whole searching range. The value of the probability in the searching range will be $p(o_{\mathrm{AP}n}|s) = 1/(2\Delta_{\mathrm{search}} + 1)$, while in the other cases will be 0.

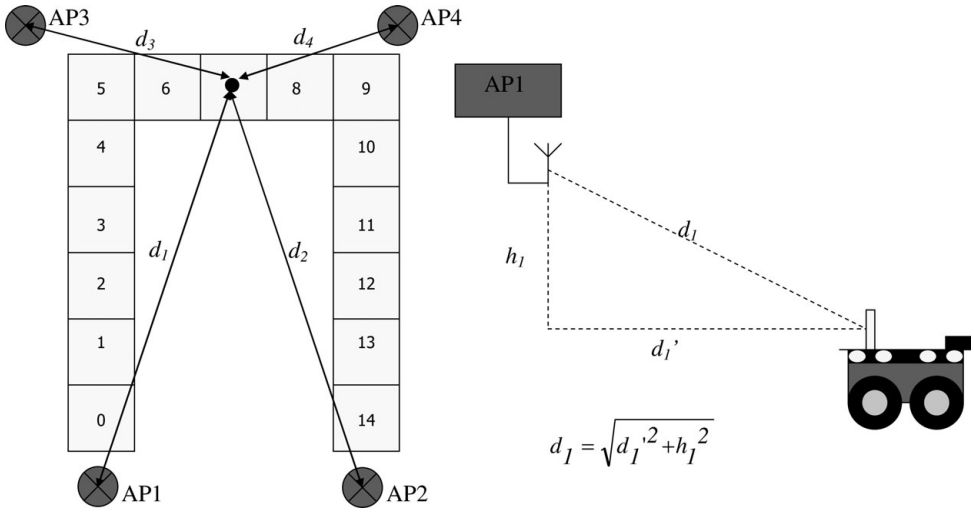


Fig. 8. Example of distance parameter calculation for the seven nodes and AP1.

In Fig. 9, we show an example of the observation matrix initialization for the states s_0 and s_1. In this example we obtained a maximum deviation lower than 30 dBm. Then, the selected range was RSL $\pm$ 30 dBm and the initialization value was $p(o_{\mathrm{AP}n}|s) = 1/61$.

4. Experimental Results

In order to obtain experimental results with this technique, we have used two robotics platforms shown in Section 2. These robots are called BART (Basic Agent for Robotics Tasks) and SIRA (Assistant Robotic System in Spanish acronym) and they have the following configuration: Orinoco PCMCIA Gold wireless card, Linux Red Hat 9.0 operating system, wireless tools by Jean Tourrilhes (http://www.hpl.hp.com/personal/Jean_Tourrilhes/), Orinoco driver patch by M. A. Youssef,[21] a 16 ultrasound sensor ring, and a SONY pan-tilt-zoom camera. We have modified the Lucent Wavelan driver for Linux applying the patch of M. A. Youssef to obtain the signal strength received from all access points in the robot range using active scanning.

We present two kinds of experimental results, firstly for testing the initialization method and, secondly, several experiments are carried out to compare the automatic training method with two other classical methods. All the experiments have been carried out in real conditions of the environment, with wireless devices on and people working in it.

First of all, we have measured the difference between the propagation model described in Eq. (14) and the real measures. We placed the robot in 67 positions of the third corridor (the first 24 positions), main corridor (the next 19 positions), and in the fourth corridor (the last 24 positions). Then, we measured the real received signal level (RSL) in the WiFi interface for all APs of the environment. We have used propagation model (14) with the following values: TSL = 20 dBm, $G_{\mathrm{TX}} = 5$ db, $G_{\mathrm{RX}} = 3$ dB, $n_{\mathrm{W}} = 5$ (indoor environment with obstructed path), and X_a with $a = 20$ (heavy building construction with a lot of partitions) in order to calculate the theoretical RSL to measure in these positions. We obtained a maximum difference from theoretical to real propagation greater than 30 dBm in AP3. Figure 10 shows the deviation between the real and theoretical received signal levels for AP1, AP2, AP3, and AP4 of the environment.

Therefore, we initially selected a searching range of $\Delta_{\mathrm{search}} = \pm 40$ dBm with respect to the value obtained from the model. To initialize the WiFi observation matrices, the values within the searching range were initialized to $p(o_{\mathrm{AP}x}|s) = 1/(2\Delta_{\mathrm{search}} + 1) = 1/81$ while the rest of observation values were initialized to 0. To show advantages of this searching method, we designed an experiment in

$$p(o_{\mathrm{AP}n} \mid s) = \frac{1}{2 \cdot \Delta_{\mathrm{search}} + 1} = \frac{1}{61}$$

$RSLs_0 \pm \Delta_{\mathrm{search}}$

$RSLs_1 \pm \Delta_{\mathrm{search}}$

	$o_{\mathrm{AP}n}$=0	$o_{\mathrm{AP}n}$=1	$o_{\mathrm{AP}n}$=2	...	$o_{\mathrm{AP}n}$=60	$o_{\mathrm{AP}n}$=61	...
s_0	0	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	0
s_1	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	$1/(2\Delta_{\mathrm{search}}+1)$	0	0

Fig. 9. Example of WiFi observation matrix initialization for the $o_{\mathrm{AP}n}$.

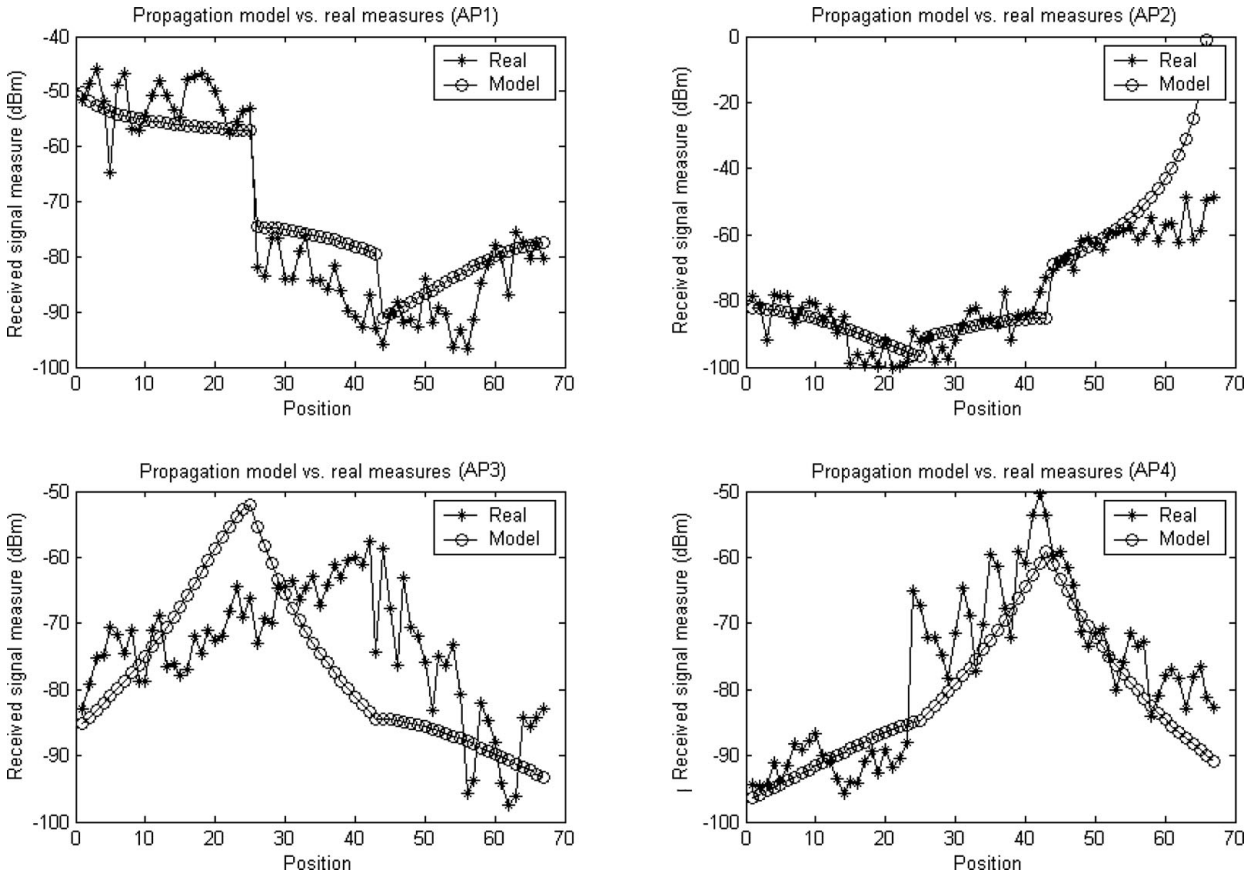


Fig. 10. Difference between real measures and estimated propagation model value.

which we checked following two parameters:

- Training error percentage: it is the percentage of training error (difference between reference and estimated state) with respect to all states in the training frames.
- Iteration number reduction: the percentage obtained between the number of iterations reduced with a certain searching range and the total iterations obtained with the original Baum–Welch algorithm.

We designed a test to validate the initialization method in which we varied the searching range from 100% (original Baum–Welch) to 10%. We used several frames with approximately 100 observations–actions stored and then we compared the training error percentage and iteration number reduction. Results are shown in Fig. 11.

Using this initialization method we obtained a reduction of approximately 20% in the number of iterations with respect to the original Baum–Welch algorithm in the range from 60 dBm to 40 dBm.

The training error percentage was maintained with almost the same value as in the original Baum–Welch algorithm in the range from 90 dBm to 40 dBm.

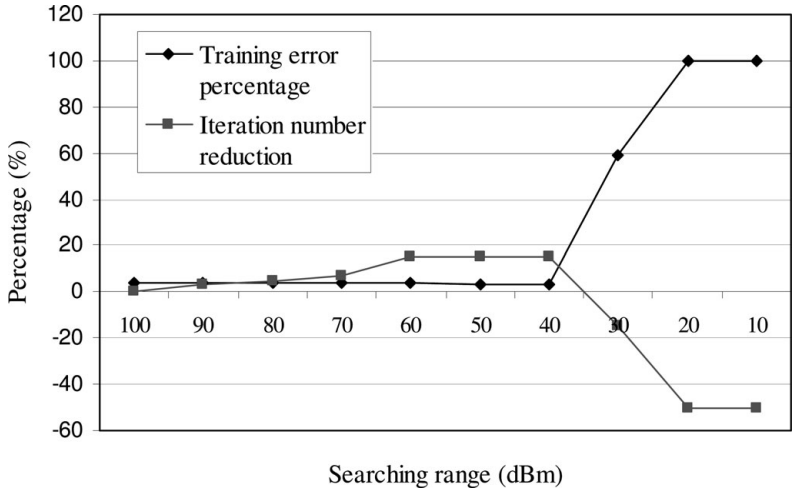


Fig. 11. Results of the searching range reduction.

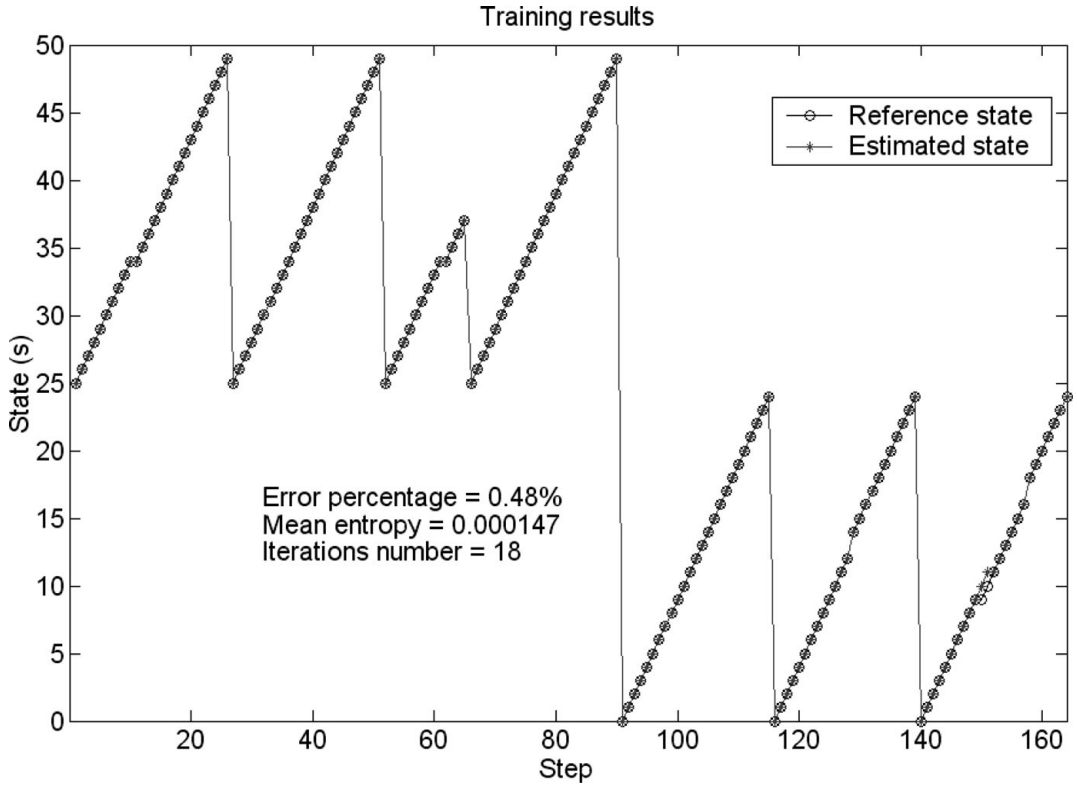
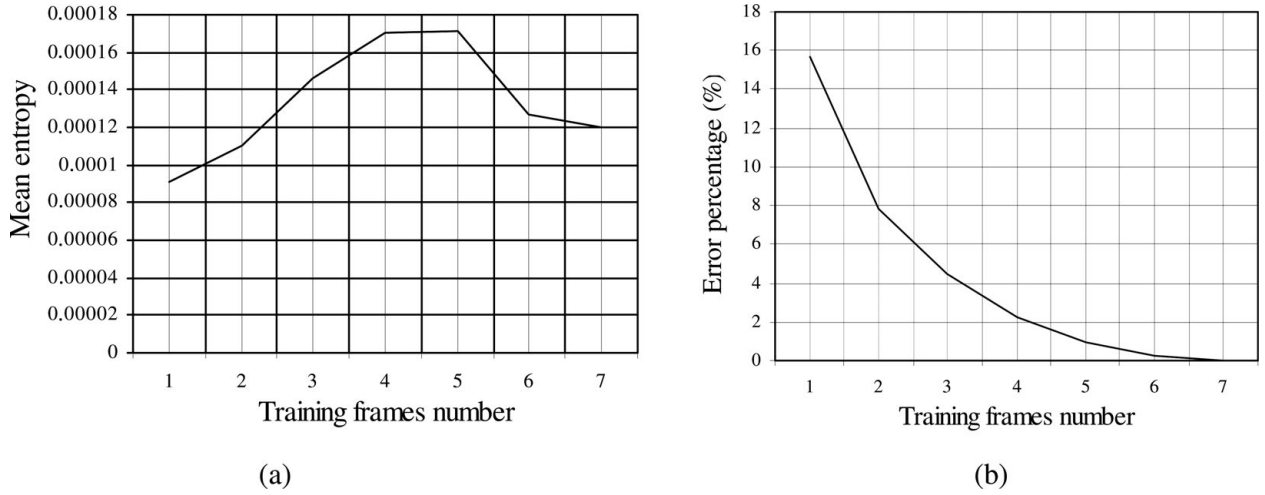


Fig. 12. Training results with seven frames.

When the search margin has a very small value (below 30 dBm) the effect of the reduction is negative. Then, with our initialization proposal, the algorithm converges with more iterations than Baum–Welch's algorithm and with higher percentage of error. Due to the searching range, it is lower than the deviation between the real propagation and the value predicted by the propagation model.

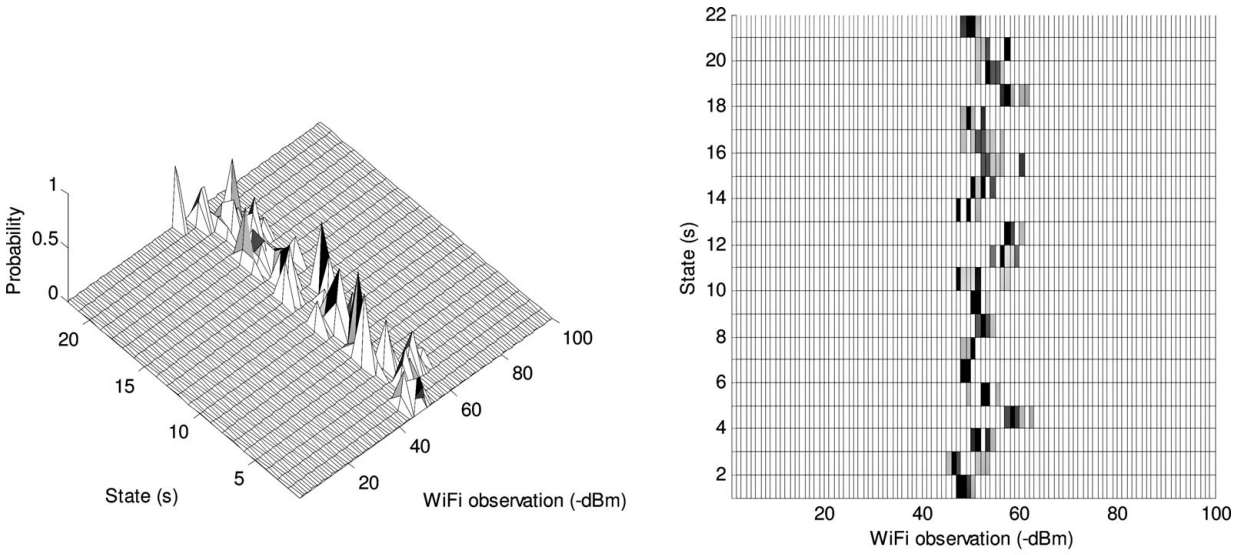
Once we have selected the searching range to use in the training method, we used the initialization process and evaluated the effect over the training percentage error and the mean entropy. Based on Fig. 11, we selected a 50 dBm searching range, as the average value between 60 dBm and 40 dBm. Figure 12 shows several training frames, with reference and estimated state by our algorithm. In this experiment we trained the robot along seven frames and we obtained a 0.48% error percentage and 0.0000147 value of mean entropy. This represents a low error and mean entropy for this stage.

Now, we focus our interest in obtaining the optimal number of frames needed for the WiFi and US map construction. It is important to select the minimum number of frames to reduce the computational cost. For this purpose we had the robot navigating around the environment for 2 h in an automatic way using our robust local navigation system. Then, we obtained the mean entropy of the training data set using from one to seven frames (Fig. 13(a)) and the localization error percentage during the training stage (Fig. 13(b)). We conclude that the number of frames greater than five in the training data set yields an average entropy value lower than



(a) (b)

Fig. 13. Optimal frames number for the training method.



(a) (b)

Fig. 14. (a) Automatic training method results for AP1 and (b) top view of the results.

0.00012 and a localization error percentage lower than 0.5%. Therefore six is the optimal number of frames to be used in the training data set.

Figure 14(a) shows an example of the automatic algorithm results for the WiFi map obtained from the AP1 along the 22 reference states in corridors 3 and 4. The black color indicates that the probability of obtaining an observation value is the highest, whereas the white color indicates the lowest probability. The top view of the result is depicted in the Fig. 14(b). As you can see, the WiFi map is a probabilistic representation that can be used in probabilistic localization system.

Finally, we compare our method with two other classical methods existing in the literature (manual[10,11] and model-based[15]). With this experiment we show that the automatic training method improves the manual and model-based methods. We compare three main parameters, the first two parameters are related to the training stage (man-work needed and time spent) and the last one is related to the localization stage (localization error percentage):

- Man-work needed: it means the necessity of having an expert to place the robot, to collect the data, to launch applications, etc., during the training phase. We evaluate this parameter like *LOW, MEDIUM* or *HIGH*, where *LOW* means that supervision of a man is not necessary and *HIGH* means the absolute necessity to have a person taking care of the training stage.
- Training time: it is the time spent in the training stage.
- Localization error percentage: it is the percentage of the localization error (difference between estimated and reference state) with respect to the total estimated states during the localization stage.

In the first classical method, the manual one, we trained the system by positioning the robot along the 100 states in a manual mode. When the robot was placed in a new state, an application was started to take several samples and to calculate the mean value. Our robot took 60 WiFi signal samples and the US observations and then the observation matrices were calculated as the average value at each state. This needed 9 h and a half of an intensive man-work.

Table I. Training methods comparison.

Method	Training time	Man-work needed	Localization error percentage (%)
Model-based	< 30 s	LOW	98
Manual	9 h 30 min	HIGH	24
Automatic	2 h	LOW	8

The model-based method uses the approximated propagation model given by Eq. (14) to obtain the observation matrices. The matrices are calculated by obtaining the expected RSL at each state. This observation is then fixed to 1 in the observation matrix. The rest of the possible observations in this state are set to 0.

In the localization stage, we collected the estimation done by the robot in 100 different cases, and then we have compared it with the reference positions to obtain the location error percentage. Table 1 shows a comparative between the three methods.

The lowest localization error percentage is achieved by the method proposed in our work, while the model-based method achieved the best training time. The man-work needed is low both in model-based and automatic methods.

We have applied this automatic training method to a POMD global navigation system based on WiFi observations in ref. [2]. When we gave the initial state to the navigation system, the POMDP algorithm yielded 100% of true locations during 50 experiments, and the navigation algorithm was able to recover from lost states. In cases where we did not give the initial state, the algorithm estimates 98% of the true locations. With these results we can conclude that the WiFi signal strength and ultrasound maps, obtained with the algorithm proposed in this work are very useful in this type of navigation systems.

5. Conclusions

To conclude, the next key points should be remarked.

- In this work we have presented an automatic training method for obtaining the WiFi ($\vartheta_{\mathrm{AP}n}$) and Ultrasound (ϑ_{US}) observation matrices, and transition matrix (T), in order to use them in a WiFi+Ultrasound POMDP navigation system.
- The method is based on a previously designed local navigation system and the Baum–Welch algorithm. We propose a novel process to initialize the Baum–Welch algorithm using an indoor radio propagation model. With this initialization process we achieve a 20% reduction in the number of iterations of the algorithm.
- The WiFi map achieved by using this technique is a probabilistic representation of the WiFi signal evolution in certain discrete positions of the environment along the time. In this way, it can be used either in a probabilistic or in a deterministic WiFi localization system simply selecting the most probable observation at each location.
- We have compared our method with two classical methods and we have concluded that our proposal achieves the best compromise between training time, man-work needed, and localization error.
- The method proposed in this work shows that calibration effort can be significantly reduced while maintaining enough localization accuracy for robots navigation missions. This effectively diminishes one of the most daunting practical barriers to wider adoption of this type of localization technique.
- We have used the automatic training method within a POMDP navigation system and we have observed that the robot is localized from the first execution step in most of cases. In addition, it is able to recover from localization losses.

In future, we will generalize our method to Department of Electronics environment. This is a more complex environment with unstructured areas where the H-shape model does not fit correctly and then it is necessary to use some others geometrical models. We will try to speed-up the algorithm by using an Ad-Hoc hardware. We also want to use this automatic method for constructing the WiFi map using the robot and then to use this map for localizing people having a PDA.

Acknowledgments

This work has been funded by grant S-0505/DPI/000176 (Robocity2030 Project) from the Science Department of Community of Madrid, and TRA2005-08529-C02-01 (MOVICOM Project) from the Spanish Ministry of Science and Technology (MCyT).

References

1. M. E. López, L. M. Bergasa, R. Barea and M. S. Escudero, "A navigation system for assistant robots using visually augmented POMDPs," *Autonom. Rob.* **19**(1), 77–87 (2005).
2. M. A. Sotelo, M. Ocaña, L. M. Bergasa, R. Flores, M. Marrón and M. A. García, "Low level controller for a POMDP based on WiFi observations," *Rob. Autonom. Syst.* **55**(2), 132–145 (2007).
3. I. Cox, "Blanche-an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Trans. Rob. Automat.* **7**(2), 193–204 (1991).
4. R. Want, A. Hopper, V. Falco and J. Gibbons, "The active badge location system," *ACM Trans. Info. Syst.* **10**, 91–102 (1992).
5. J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale and S. Shafer, "Multi-Camera Multi-Person Tracking for Easy Living," *Proceedings of Third IEEE International Workshop on Visual Surveillance* (IEEE Computer Society, Washington, DC, 2002) pp. 3–10.
6. N. B. Priyantha, A. Chakraborthy and H. Balakrishnan, "The Cricket Location Support System," *Proceedings of the Sixth ACM MobiCom* (ACM New York, 2002) pp. 155–164.
7. R. Barber, M. Mata, M. J. L. Boada, J. M. Armingol and M. A. Salichs, "A Perception System based on Laser Information for Mobile Robot Topologic Navigation," *Proceedings of 28th Annual Conference of the IEEE Industrial Electronics Society* (IEEE Industrial Electronics Society, Sevilla, Spain, 2002) pp. 2779–2784.
8. A. Bose and C. H. Foh, "A Practical Path Loss Model for Indoor WiFi Positioning Enhancement," *Proceedings of the International Conference on Information, Communications and Signal Processing ICICS'07* (Singapore, 2007) pp. 1–5.
9. V. Otsason, Accurate Indoor Localization Using Wide GSM Fingerprinting *Master's Thesis* (University of Tartu, 2005).
10. P. Bahl and V. N. Padmanabhan, "RADAR: A, In-building RF-based User Location and Tracking System," *Proceedings of the IEEE Infocom 2000*, vol. 2 (IEEE Computer and Communications Societies, Tel-Aviv, Israel, 2000) pp. 775–784.
11. A. Ladd, K. Bekris, A. Rudys, G. Marceu, L. Kavraki and D. Wallach, "Robotics Based Location Sensing Using Wireless Ethernet," *Proceedings of the International Conference on Mobile Computing and Networking* (ACM New York, NY, USA, Atlanta, GA, 2002) pp. 227–238.
12. A. Howard, S. Siddiqi and G. S. Sukhatme, "An Experimental Study of Localization Using Wireless Ethernet," *Proceedings of the International Conference on Field and Service Robotics* Lake Yamanaka, Japan (2003).
13. A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, B. Schilit, "PlaceLab: Device Positioning Using Radio Beacons in the Wild", **In**: *Pervasive Computing*, Lecture Notes in Computer Science, vol. 3468 (Springer-Verlag, Berlin, Germany, 2005) pp. 116–133.
14. P. Enge and P. Misra, "Special Issue on GPS: The Global Positioning System", *Proceedings of the IEEE* (IEEE, Piscataway, NJ, Jan. 1999) pp. 3–15.
15. V. Matellán, J. M. Cañas and O. Serrano, "WiFi localization methods for autonomous robots," *Robotica* **24**(4), 455–461 (2006).
16. M. Ocaña, WiFi Global Localization System applied to a Semiautonomous Robot Navigation System *Ph.D. Thesis* (Department of Electronics, Polytechnic School, University of Alcala, Spain, 2005).
17. E. Lopez, R. Barea, L. M. Bergasa and M. S. Escudero, "A human-robot cooperative learning system for easy installation of assistant robots in new working environments," *J. Intell. Rob. Syst.* **40**, 233–265 (2004).
18. M. E. López, Global Navigation System Based on Partially Markov Decision Process. Application in an Assistant Robot *Ph.D. Thesis* (Department of Electronics, Polytechnic School, University of Alcalá, Spain, 2004).
19. R. Simmons, R. Coodwin, K. Z. Haigh, S. Koenig and J. O'Sullivan, "A Layered Architecture for Office Delivery Robots," *Proceedings of the First International Conference on Autonomous Agents (Agents'97)* (ACM Press, Marina del Rey, CA, 1997) pp. 245–252.
20. K. Konolige, *Saphira Robot Control Architecture* (SRI International, 2002).
21. M. Youssef and A. Agrawala, "Small-Scale Compensation for WLAN Location Determination Systems," *Proceedings of the 2003 ACM workshop on Wireless security* (2003) pp. 11–20.