Global Navigation of Assistant Robots using Partially Observable Markov Decision Processes

María Elena López, Rafael Barea, Luis Miguel Bergasa, Manuel Ocaña & María Soledad Escudero
Electronics Department. University of Alcalá
Spain

1. Introduction

In the last years, one of the applications of service robots with a greater social impact has been the assistance to elderly or disabled people. In these applications, assistant robots must robustly navigate in structured indoor environments such as hospitals, nursing homes or houses, heading from room to room to carry out different nursing or service tasks.

Although the state of the art in navigation systems is very wide, there are not systems that simultaneously satisfy all the requirements of this application. Firstly, it must be a very robust navigation system, because it is going to work in highly dynamic environments and to interact with non-expert users. In second place, and to ensure the future commercial viability of this kind of prototypes, it must be a system very easy to export to new working domains, not requiring a previous preparation of the environment or a long, hard and tedious configuration process. Most of the actual navigation systems propose "ad-hoc" solutions that only can be applied in very specific conditions and environments. Besides, they usually require an artificial preparation of the environment and are not capable of automatically recover general localization failures.

In order to contribute to this research field, the Electronics Department of the University of Alcalá has been working on a robotic assistant called SIRA, within the projects SIRAPEM (Spanish acronym of Robotic System for Elderly Assistance) and SIMCA (Cooperative multi-robot assistance system). The main goal of these projects is the development of robotic aids that serve primary functions of tele-presence, tele-medicine, intelligent reminding, safeguarding, mobility assistance and social interaction. Figure 1 shows a simplified diagram of the SIRAPEM global architecture, based on a commercial platform (the PeopleBot robot of ActivMedia Robotics) endowed with a differential drive system, encoders, bumpers, two sonar rings (high and low), loudspeakers, microphone and on-board PC. The robot has been also provided with a PTZ color camera, a tactile screen and wireless Ethernet link. The system architecture includes several human-machine interaction systems, such as voice (synthesis and recognition speech) and touch screen for simple command selection.

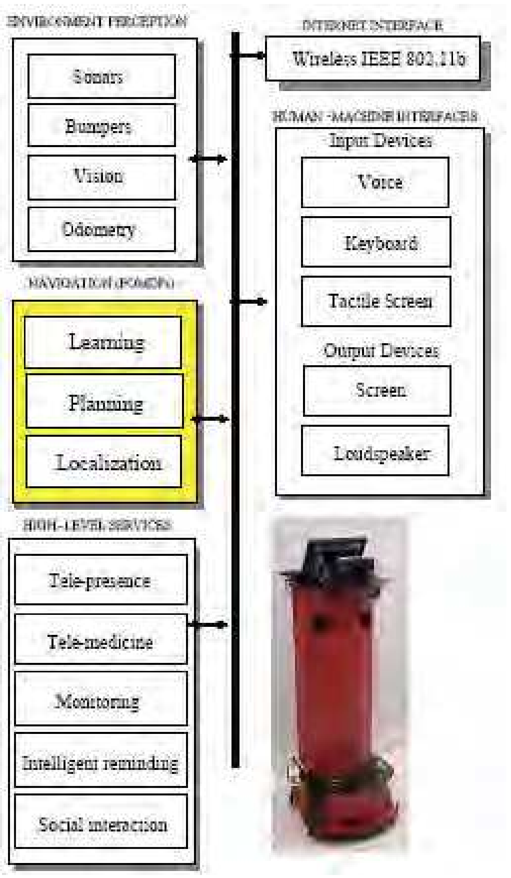


Fig. 1. Global architecture of the SIRAPEM System

This chapter describes the navigation module of the SIRAPEM project, including localization, planning and learning systems. A suitable framework to cope with all the requirements of this application is Partially Observable Markov Decision Processes (POMDPs). These models use probabilistic reasoning to deal with uncertainties, and a topological representation of the environment to reduce memory and time requirements of the algorithms. For the proposed global navigation system, in which the objective is the guidance to a goal room and some low-level behaviors perform local navigation, a topological discretization is appropriate to facilitate the planning and learning tasks.
POMDP models provide solutions to localization, planning and learning in the robotics context, and have been used as probabilistic reasoning method in the three modules of the navigation system of SIRA. The main contributions of the navigation architecture of SIRA, regarding other similar ones (that we'll be referenced in next section), are the following:

- Addition of visual information to the Markov model, not only as observation, but also for improving state transition detection. This visual information reduces typical perceptual aliasing of proximity sensors, accelerating the process of global localization when initial pose is unknown.

- Development of a new planning architecture that selects actions to combine several objectives, such as guidance to a goal room, localization to reduce uncertainty, and environment exploration.
- Development of a new exploration and learning strategy that takes advantage of human-machine interaction to robustly and quickly fast learn new working environments.

The chapter is organized as follows. Section 2 places this work within the context of previous similar ones. A brief overview of POMDPs foundations is presented as background in section 3. Section 4 describes the proposed Markov model while section 5 shows the global architecture of the navigation system. The localization module is described in section 6, the two layers of the planning system are shown in section 7 and the learning and exploration module are explained in section 8. Finally, we show some experimental results (section 9), whereas a final discussion and conclusion summarizes the chapter (sections 10 and 11).

2. Related Previous Work

Markov models, and particularly POMDPs, have already been widely used in robotics, and especially in robot navigation. The robots DERVISH (Nourbakhsh et al., 1995), developed in the Stanford University, and Xavier (Koenig & Simmons, 1998), in the Carnegie Mellon University, were the first robots successfully using this kind of navigation strategies for localization and action planning. Other successful robots guided with POMDPs are those proposed by (Zanichelli, 1999) or (Asoh et al., 1996). In the nursing applications field, in which robots interact with people and uncertainty is pervasive, robots such as Flo (Roy et al., 2000) or Pearl (Montemerlo et al., 2002) use POMDPs at all levels of decision making, and not only in low-level navigation routines.
However, in all these successful navigation systems, only proximity sensors are used to perceive the environment. Due to the typical high perceptual aliasing of these sensors in office environments, using only proximity sensors makes the Markov model highly non-observable, and the initial global localization stage is rather slow.
On the other hand, there are quite a lot of recent works using appearance-based methods for robot navigation with visual information. Some of these works, such as (Gechter et al., 2001) and (Regini et al., 2002), incorporate POMDP models as a method for taking into account previous state of the robot to evaluate its new pose, avoiding the teleportation phenomena. However, these works are focused on visual algorithms, and very slightly integrate them into a complete robot navigation architecture. So, the above referenced systems don’t combine any other sensorial system, and use the POMDP only for localizing the robot, and not for planning or exploring.
This work is a convergence point between these two research lines, proposing a complete navigation architecture that adds visual information to proximity sensors to improve previous navigation results, making more robust and faster the global localization task. Furthermore, a new Markov model is proposed that better adapts to environment topology, being completely integrated with a planning system that simultaneously contemplates several navigation objectives.
Regarding the learning system, most of the related works need a previous “hand-made” introduction of the Markov model of a new environment. Learning a POMDP involves two main issues: (1) obtaining its topology (structure), and (2) adjusting the parameters (probabilities) of the

model. The majority of the works deals with the last problem, using the well-known EM algorithm to learn the parameters of a Markov model whose structure is known (Thrun et al., 1998; Koenig and Simmons, 1996). However, because computational complexity of the learning process increases exponentially as the number of states increases, these methods are still time consuming and its working ability is limited to learn reduced environments. In this work, the POMDP model can be easily obtained for new environments by means of human-robot cooperation, being an optimal solution for assistant robots endowed with human-machine interfaces. The topological representation of the environment is intuitive enough to be easily defined by the designer. The uncertainties and observations that constitute the parameters of the Markov model are learned by the robot using a modification of the EM algorithm that exploits slight user supervision and topology constraints to highly reduce memory requirements and computational cost of the standard EM algorithm.

3. POMDPs Review

Although there is a wide literature about POMDPs theory (Papadimitriou & Tsitsiklis, 1987; Puterman, 1994; Kaelbling et al., 1996) in this section some terminology and main foundations are briefly introduced as theoretical background of the proposed work.

A Markov Decision Process (MDP) is a model for sequential decision making, formally defined as a tuple $\{\boldsymbol{S},\boldsymbol{A},\boldsymbol{T},\boldsymbol{R}\}$, where,

- $\boldsymbol{S}$ is a finite set of states ($s\in\boldsymbol{S}$).
- $\boldsymbol{A}$ is a finite set of actions ($a\in\boldsymbol{A}$).
- $\boldsymbol{T}=\{p(s'|s,a) \quad \forall\ (s,s'\in\boldsymbol{S} \quad a\in\boldsymbol{A})\}$ is a state transition model which specifies a conditional probability distribution of posterior state s' given prior state s and action executed a.
- $\boldsymbol{R}=\{r(s,a) \quad \forall\ (s\in\boldsymbol{S} \quad a\in\boldsymbol{A})\}$ is the reward function, that determines the immediate utility (as a function of an objective) of executing action a at state s.

A MDP assumes the Markov property, which establishes that actual state and action are the only information needed to predict next state:

$$p(s_{t+1} \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1} \mid s_t, a_t) \tag{1}$$

In a MDP, the actual state s is always known without uncertainty. So, planning in a MDP is the problem of action selection as a function of the actual state (Howard, 1960). A MDP solution is a policy $a=\pi(s)$, which maps states into actions and so determines which action must be executed at each state. An optimal policy $a=\pi^*(s)$ is that one that maximizes future rewards. Finding optimal policies for MDPs is a well known problem in the artificial intelligent field, to which several exact and approximate solutions (such as the "value iteration" algorithm) have been proposed (Howard, 1960; Puterman, 1994).

Partially Observable Markov Decision Processes (POMDPs) are used under domains where there is not certainty about the actual state of the system. Instead, the agent can do observations and use them to compute a probabilistic distribution over all possible states. So, a POMDP adds:

- $\boldsymbol{O}$, a finite set of observations ($o\in\boldsymbol{O}$)
- $\boldsymbol{\vartheta}=\{p(o|s) \quad \forall\ o\in\boldsymbol{O},\ s\in\boldsymbol{S}\}$ is an observation model which specifies a conditional probability distribution over observations given the actual state s.

Because in this case the agent has not direct access to the current state, it uses actions and observations to maintain a probability distribution over all possible states, known as the "belief distribution", *Bel(S)*. A POMDP is still a markovian process in terms of this probability distribution, which only depends on the prior belief, prior action, and current observation. This belief must be updated whenever a new action or perception is carried out. When an action *a* is executed, the new probabilities become:

$$\mathrm{Bel}_{\mathrm{posterior}}(\mathbf{S}=s') = K \cdot \sum_{s\in S} p(s'\,|\,s,a)\cdot \mathrm{Bel}_{\mathrm{prior}}(s) \qquad \forall s' \in \mathbf{S} \tag{2}$$

where K is a normalization factor to ensure that the probabilities all sum one. When a sensor report *o* is received, the probabilities become:

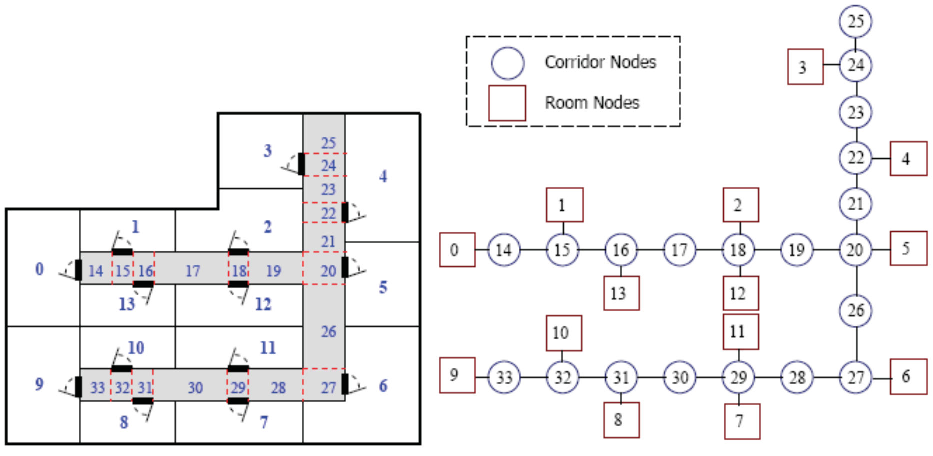
$$\mathrm{Bel}_{\mathrm{posterior}}(\mathbf{S}=s) = K \cdot p(o\,|\,s)\cdot \mathrm{Bel}_{\mathrm{prior}}(s) \qquad \forall s \in \mathbf{S} \tag{3}$$

In a POMDP, a policy $a=\pi(\boldsymbol{Bel})$ maps beliefs into actions. However, what in a MDP was a discrete state space problem, now is a high-dimensional continuous space. Although there are numerous studies about finding optimal policies in POMDPs (Cassandra, 1994; Kaelbling et al.,1998), the size of state spaces and real-time constraints make them infeasible to solve navigation problems in robotic contexts. This work uses an alternative approximate solution for planning in POMDP-based navigation contexts, dividing the problem into two layers and applying some heuristic strategies for action selection.

In the context of robot navigation, the states of the Markov model are the locations (or nodes) of a topological representation of the environment. Actions are local navigation behaviors that the robot can execute to move from one state to another, and observations are perceptions of the environment that the robot can extract from its sensors. In this case, the Markov model is partially observable because the robot may never know exactly which state it is in.

4. Markov Model for Global Navigation

A POMDP model for robot navigation is constructed from two sources of information: the topology of the environment, and some experimental or learned information about action and sensor errors and uncertainties.



a) Simplified map of an indoor environment

b) Topological graph for the environment of figure 2.a)

Fig. 2. Topological graph of an environment map.

Taking into account that the final objective of the SIRAPEM navigation system is to direct the robot from one room to another to perform guiding or service tasks, we discretize the environment into coarse-grained regions (nodes) of variable size in accordance with the topology of the environment, in order to make easier the planning task. As it's shown in figure 2 for a virtual environment, only one node is assigned to each room, while the corridor is discretized into thinner regions. The limits of these regions correspond to any change in lateral features of the corridor (such as a new door, opening or piece of wall). This is a suitable discretization method in this type of structured environments, since nodes are directly related to topological locations in which the planning module may need to change the commanded action.

4.1. The elements of the Markov model: states, actions and observations

States (***S***) of the Markov model are directly related to the nodes of the topological graph. A single state corresponds to each room node, while four states are assigned to each corridor node, one for each of the four orientations the robot can adopt.

The actions (***A***) selected to produce transitions from one state to another correspond to local navigation behaviors of the robot. We assume imperfect actions, so the effect of an action can be different of the expected one (this will be modeled by the transition model **T**). These actions are:

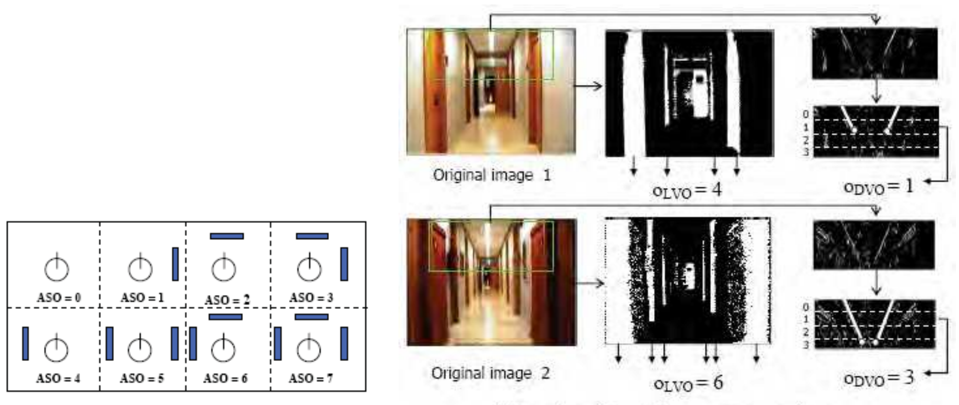
(1) *"Go out room" (a_O)*: to traverse door using sonar an visual information in room states,
(2) *"Enter room" (a_E)*: only defined in corridor states oriented to a door,
(3) *"Turn right" (a_R)*: to turn 90° to the right,
(4) *"Turn Left" (a_L)*: to turn 90° to the left,
(5) *"Follow Corridor" (a_F)*: to continue through the corridor to the next state, and
(6) *"No Operation" (a_{NO})*: used as a directive in the goal state.

Finally, the observations (***O***) in our model come from the two sensorial systems of the robot: sonar and vision. Markov models provide a natural way to combine multisensorial information, as it will be shown in section 4.2.1. In each state, the robot makes three kind of observations:

(1) *"Abstract Sonar Observation"* (o_{ASO}). Each of the three nominal directions around the robot (left, front and right) is classified as "free" or "occupied" using sonar information, and an abstract observation is constructed from the combination of the percepts in each direction (thus, there are eight possible abstract sonar observations, as it's shown in figure 3.a).
(2) *"Landmark Visual Observation"* (o_{LVO}). Doors are considered as natural visual landmarks, because they exist in all indoor environments and can be easily segmented from the image using color (previously trained) and some geometrical restrictions. This observation is the number of doors (in lateral walls of the corridor) extracted from the image (see figure 3.b), and it reduces the perceptual aliasing of sonar by distinguishing states at the beginning from states at the end of a corridor. However, in long corridors, doors far away from the robot can't be easily segmented from the image (this is the case of image 2 of figure 3.b), and this is the reason because we introduce a third visual observation.
(3) *"Depth Visual Observation"* (o_{DVO}). As human-interaction robots have tall bodies with the camera on the top, it's possible to detect the vanishing ceiling lines, and

classify its length into a set of discrete values (in this case, we use four quantification levels, as it's shown in figure 3.b). This is a less sensitive to noise observation than using floor vanishing lines (mainly to occlusions due to people walking through the corridor), and provides complementary information to o_{LVO}.

Figure 3.b shows two scenes of the same corridor from different positions, and their corresponding o_{LVO} and o_{DVO} observations. It's shown that these are obtained by means of very simple image processing techniques (color segmentation for o_{LVO} and edge detection for o_{DVO}), and have the advantage, regarding correlation techniques used in (Gechter et al., 2001) or (Regini et al., 2002), that they are less sensitive to slight pose deviations within the same node.



a) Abstract Sonar Observation (o_{ASO}) of the Markov model. b) Examples of visual observations: o_{LVO} and o_{DVO}.

Fig. 3. Observations of the proposed Markov model.

4.2. Visual information utility and improvements

Visual observations increase the robustness of the localization system by reducing perceptual aliasing. On the other hand, visual information also improves state transition detection, as it's shown in the following subsections.

4.2.1. Sensor fusion to improve observability

Using only sonar to perceive the environment makes the Markov model highly non-observable due to perceptual aliasing. Furthermore, the "Abstract Sonar Observation" is highly dependent on doors state (opened or closed). The addition of the visual observations proposed in this work augments the observability of states. For example, corridor states with an opened door on the left and a wall on the right produces the same abstract sonar observation (o_{ASO}=1) independently if they are at the beginning or at the end of the corridor. However, the number of doors seen from the current state (o_{LVO}) allows to distinguish between these states.

POMDPs provide a natural way for using multisensorial fusion in their observation models (*p(o|s)* probabilities). In this case, **o** is a vector composed by the three observations proposed in the former subsection. Because these are independent observations, the observation model can be simplified in the following way:

$$\mathrm{p}(o\,|\,s) = \mathrm{p}(o_{ASO}, o_{LVO}, o_{DVO}\,|\,s) = \mathrm{p}(o_{ASO}\,|\,s)\,\mathrm{p}(o_{LVO}\,|\,s)\,\mathrm{p}(o_{DVO}\,|\,s) \tag{4}$$

4.2.2. Visual information to improve state transition detection

To ensure that when the robot is in a corridor, it only adopts the four allowed directions without large errors, it's necessary that, during the execution of a "Follow Corridor" action, the robot becomes aligned with the corridor longitudinal axis. So, when the robot stands up to a new corridor, it aligns itself with a subtask that uses visual vanishing points, and during corridor following, it uses sonar buffers to detect the walls and construct a local model of the corridor. Besides, an individual "Follow Corridor" action terminates when the robot eaches a new state of the corridor. Detecting these transitions only with sonar readings is very critical when doors are closed.
To solve this problem, we add visual information to detect door frames as natural landmarks of state transitions (using color segmentation and some geometrical restrictions). The advantage of this method is that the image processing step is fast and easy, being only necessary to process two lateral windows of the image as it's shown in figure 4.

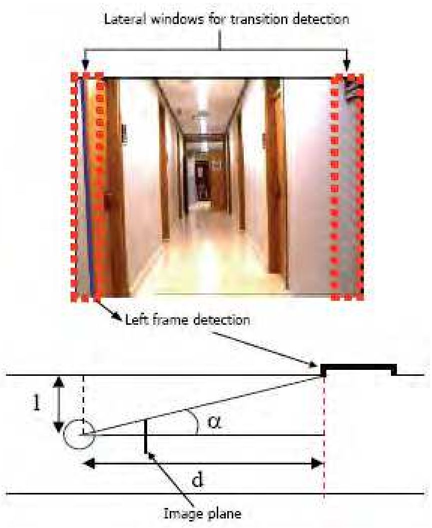


Fig. 4. State transition detection by means of visual information.

Whenever a vertical transition from wall to door color (or vice versa) is detected in a lateral window, the distance to travel as far as that new state is obtained from the following formula, using a pin-hole model of the camera (see figure 4):

$$d = \frac{l}{tg(\alpha)} = K \cdot l \tag{5}$$

where *l* is the distance of the robot to the wall in the same side as the detected door frame (obtained from sonar readings) and α is the visual angle of the door frame. As the detected frame is always in the edge of the image, the visual angle α only depends on the focal distance of the camera, that is constant for a fixed zoom (and known from camera specifications). After covering distance *d* (measured with relative odometry readings), the robot reaches the new state. This transition can be confirmed (fused) with sonar if the door is opened. Another advantage of this transition detection approach is that no assumptions are made about doors or corridor widths.

4.3. Action and observation uncertainties

Besides the topology of the environment, it's necessary to define some action and observation uncertainties to generate the final POMDP model (transition and observation matrixes). A first way of defining these uncertainties is by introducing some experimental "hand-made" rules (this method is used in (Koenig & Simmons, 1998) and (Zanichelli, 1999)). For example, if a "Follow" action (a_F) is commanded, the expected probability of making a state transition (F) is 70%, while there is a 10% probability of remaining in the same state (N=no action), a 10% probability of making two successive state transitions (FF), and a 10% probability of making three state transitions (FFF). Experience with this method has shown it to produce reliable navigation. However, a limitation of this method is that some uncertainties or parameters of the transition and observation models are not intuitive for being estimated by the user. Besides, results are better when probabilities are learned to more closely reflect the actual environment of the robot. So, our proposed learning module adjusts observation and transition probabilities with real data during an initial exploration stage, and maintains these parameters updated when the robot is performing another guiding or service tasks. This module, that also makes easier the installation of the system in new environments, is described in detail in section 8.

5. Navigation System Architecture

The problem of acting in partially observable environments can be decomposed into two components: a state estimator, which takes as input the last belief state, the most recent action and the most recent observation, and returns an updated belief state, and a policy, which maps belief states into actions. In robotics context, the first component is robot localization and the last one is task planning.
Figure 5 shows the global navigation architecture of the SIRAPEM project, formulated as a POMDP model. At each process step, the planning module selects a new action as a command for the local navigation module, that implements the actions of the POMDP as local navigation behaviors. As a result, the robot modifies its state (location), and receives a new observation from its sensorial systems. The last action executed, besides the new observation perceived, are used by the localization module to update the belief distribution *Bel(S)*.

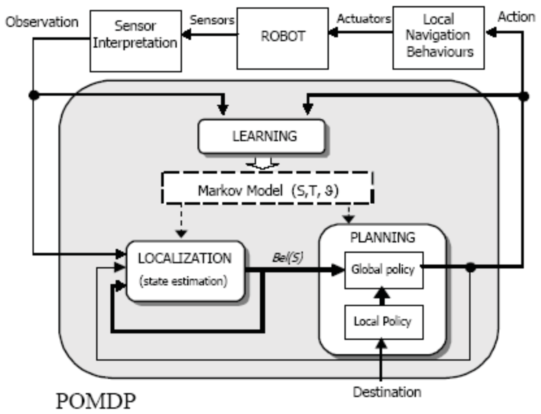


Fig. 5. Global architecture of the navigation system.

After each state transition, and once updated the belief, the planning module chooses the next action to execute. Instead of using an optimal POMDP policy (that involves high computational times), this selection is simplified by dividing the planning module into two layers:

- A local policy, that assigns an optimal action to each individual state (as in the MDP case). This assignment depends on the planning context. Three possible contexts have been considered: (1) guiding (the objective is to reach a goal room selected by the user to perform a service or guiding task), (2) localizing (the objective is to reduce location uncertainty) and (3) exploring (the objective is to learn or adjust observations and uncertainties of the Markov model).
- A global policy, that using the current belief and the local policy, selects the best action by means of different heuristic strategies proposed by (Kaelbling et al., 1996).

This proposed two-layered planning architecture is able to combine several contexts of the local policy to simultaneously integrate different planning objectives, as will be shown in subsequent sections.

Finally, the learning module (López et al., 2004) uses action and observation data to learn and adjust the observations and uncertainties of the Markov model.

6. Localization and Uncertainty Evaluation

The localization module updates the belief distribution after each state transition, using the well known Markov localization equations (2) and (3).

In the first execution step, the belief distribution can be initialized in one of the two following ways: (a) If initial state of the robot is known, that state is assigned probability 1 and the rest 0, (b) If initial state is unknown, a uniform distribution is calculated over all states.

Although the planning system chooses the action based on the entire belief distribution, in some cases it´s necessary to evaluate the degree of uncertainty of that distribution (this is, the locational uncertainty). A typical measure of discrete distributions uncertainty is the entropy. The normalized entropy (ranging from 0 to 1) of the belief distribution is:

$$\mathrm{H}(\mathbf{Bel}) = -\frac{\sum_{s\in\mathbf{S}} Bel(s)\cdot\log(Bel(s))}{\log(n_s)} \tag{6}$$

where n_s is the number of states of the Markov model. The lower the value, the more certain the distribution. This measure has been used in all previous robotic applications for characterizing locational uncertainty (Kaelbling, 1996; Zanichelli, 1999).

However, this measure is not appropriate for detecting situations in which there are a few maximums of similar value, being the rest of the elements zero, because it's detected as a low entropy distribution. In fact, even being only two maximums, that is a not good result for the localization module, because they can correspond to far locations in the environment. A more suitable choice should be to use a least square measure respect to ideal delta distribution, that better detects the convergence of the distribution to a unique maximum (and so, that the robot is globally localized). However, we propose another approximate measure that, providing similar results to least squares, is faster calculated by using only the two first maximum values of the distribution (it's also less sensitive when uncertainty is high, and more sensitive to secondary maximums during the tracking stage). This is the normalized divergence factor, calculated in the following way:

$$\mathrm{D}(\boldsymbol{Bel}) = 1-\frac{n_s\left(d_{\max}+p_{\max}\right)-1}{2\cdot n_s-1} \tag{7}$$

where d_{max} is the difference between first and second maximum values of the distribution, and p_{max} the absolute value of the first maximum. Again, a high value indicates that the distribution converges to a unique maximum. In the results section we'll show that this new measure provides much better results when planning in some kind of environments.

7. Planning under Uncertainty

A POMDP model is a MDP model with probabilistic observations. Finding optimal policies in the MDP case (that is a discrete space model) is easy and quickly for even very large models. However, in the POMDP case, finding optimal control strategies is computationally intractable for all but the simplest environments, because the beliefs space is continuous and high-dimensional.
There are several recent works that use a hierarchical representation of the environment, with different levels of resolution, to reduce the number of states that take part in the planning algorithms (Theocharous & Mahadevan, 2002; Pineau & Thrun, 2002). However, these methods need more complex perception algorithms to distinguish states at different levels of abstraction, and so they need more prior knowledge about the environment and more complex learning algorithms. On the other hand, there are also several recent approximate methods for solving POMDPs, such as those that use a compressed belief distribution to accelerate algorithms (Roy, 2003) or the 'point-based value iteration algorithm' (Pineau et al., 2003) in which planning is performed only on a sampled set of reachable belief points.
The solution adopted in this work is to divide the planning problem into two steps: the first one finds an optimal local policy for the underlying MDP ($a^*=\pi^*(s)$, or to simplify notation, $a^*(s)$), and the second one uses a number of simple heuristic strategies to select a final action ($a^*(\mathbf{Bel})$) as a function of the local policy and the belief. This structure is shown in figure 6 and described in subsequent subsections.

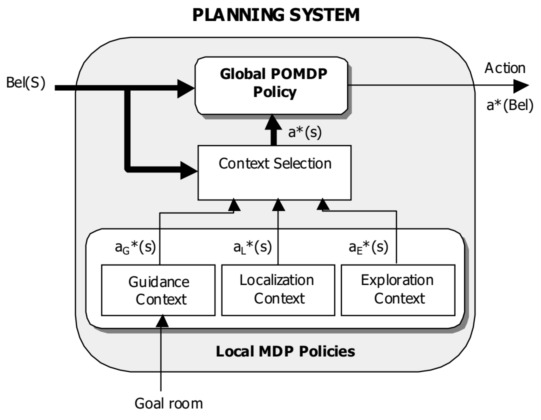


Fig. 6. Planning system architecture, consisting of two layers: (1) Global POMDP policy and (2) Local MDP policies.

7.1. Contexts and local policies

The objective of the local policy is to assign an optimal action ($a^*(s)$) to each individual state s. This assignment depends on the planning context. The use of several contexts allows the

robot to simultaneously achieve several planning objectives. The localization and guidance contexts try to simulate the optimal policy of a POMDP, which seamlessly integrates the two concerns of acting in order to reduce uncertainty and to achieve a goal. The exploration context is to select actions for learning the parameters of the Markov model.
In this subsection we show the three contexts separately. Later, they will be automatically selected or combined by the 'context selection' and 'global policy' modules (figure 6).

7.1.1. Guidance Context

This local policy is calculated whenever a new goal room is selected by the user. Its main objective is to assign to each individual state *s*, an optimal action ($a_G*(s)$) to guide the robot to the goal.
One of the most well known algorithms for finding optimal policies in MDPs is 'value iteration' (Bellman, 1957). This algorithm assigns an optimal action to each state when the reward function *r(s,a)* is available. In this application, the information about the utility of actions for reaching the destination room is contained in the graph. So, a simple path searching algorithm can effectively solve the underlying MDP, without any intermediate reward function.
So, a modification of the A* search algorithm (Winston, 1984) is used to assign a preferred heading to each node of the topological graph, based on minimizing the expected total number of nodes to traverse (shorter distance criterion cannot be used because the graph has not metric information). The modification of the algorithm consists of inverting the search direction, because in this application there is not an initial node (only a destination node). Figure 7 shows the resulting node directions for goal room 2 on the graph of environment of figure 2.

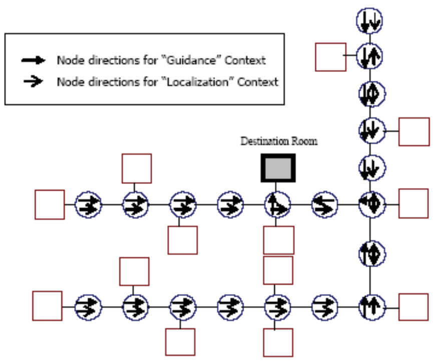

Node directions for "Guidance" Context
Node directions for "Localization" Context
Destination Room

Fig. 7. Node directions for "Guidence" (to room 2) and "Localization" contexts for environment of figure 2.

Later, an optimal action is assigned to the four states of each node in the following way: a "follow" (a_F) action is assigned to the state whose orientation is the same as the preferred heading of the node, while the remaining states are assigned actions that will turn the robot towards that heading (a_L or a_R). Finally, a "no operation" action (a_{NO}) is assigned to the goal room state.
Besides optimal actions, when a new goal room is selected, Q(s,a) values are assigned to each (s,a) pair. In the MDPs theory, Q-values (Lovejoi, 1991) characterize the utility of executing each action at each state, and will be used by one of the global heuristic policies shown in next subsection. To simplify Q-values calculation, the following criterion has been used: Q(s,a)=1 if action a is optimal at state s, Q(s,a)=-1 (negative utility) if actions a is not

defined at state s, and Q(s,a)=-0.5 for the remaining cases (actions that disaligns the robot from the preferred heading).

7.1.2. Localization Context

This policy is used to guide the robot to "Sensorial Relevant States" (SRSs) that reduce positional uncertainty, even if that requires moving it away from the goal temporarily. This planning objective was not considered in previous similar robots, such as DERVISH (Nourbakhsh et al., 1995) or Xavier (Koenig & Simmons, 1998), or was implemented by means of fixed sequences of movements (Cassandra, 1994) that don't contemplate environment relevant places to reduce uncertainty.
In an indoor environment, it's usual to find different zones that produce not only the same observations, but also the same sequence of observations as the robot traverses them by executing the same actions (for example, symmetric corridors). SRSs are states that break a sequence of observations that can be found in another zone of the graph.
Because a state can be reached from different paths and so, with different histories of observations, SRSs are not characteristic states of the graph, but they depend on the starting state of the robot. This means that each starting state has its own SRS. To simplify the calculation of SRSs, and taking into account that the more informative states are those aligned with corridors, it has been supposed that in the localization context the robot is going to execute sequences of "follow corridor" actions. So, the moving direction along the corridor to reach a SRS as soon as possible must be calculated for each state of each corridor.
To do this, the "Composed Observations" (COs) of these states are calculated from the graph and the current observation model ϑ in the following way:

$$\begin{aligned} &CO(s) = 100 \cdot o_{DVO}(s) + 10 \cdot o_{LVO}(s) + o_{ASO}(s) \\ &\text{with } \; o_{DVO}(s) = \underset{o_{DVO}}{\arg\max}\left(p(o_{DVO} \mid s)\right) \\ &\qquad\quad o_{LVO}(s) = \underset{o_{LVO}}{\arg\max}\left(p(o_{LVO} \mid s)\right) \\ &\qquad\quad o_{ASO}(s) = \underset{o_{ASO}}{\arg\max}\left(p(o_{ASO} \mid s)\right) \end{aligned} \tag{8}$$

Later, the nearest SRS for each node is calculated by studying the sequence of COs obtained while moving in both corridor directions. Then, a preferred heading (among them that align the robot with any connected corridor) is assigned to each node. This heading points at the corridor direction that, by a sequence of "Follow Corridor" actions, directs the robot to the nearest SRS (figure 7 shows the node directions obtained for environment of figure 2). And finally, an optimal action is assigned to the four states of each corridor node to align the robot with this preferred heading (as it was described in the guidance context section). The optimal action assigned to room states is always "Go out room" (a_O).
So, this policy ($a^*_L(s)$) is only environment dependent and is automatically calculated from the connections of the graph and the ideal observations of each state.

7.1.3. Exploration Context

The objective of this local policy is to select actions during the exploration stage, in order to learn transition and observation probabilities. As in this stage the Markov model is unknown (the belief can't be calculated), there is not distinction between local and global policies, whose common function is to select actions in a reactive way to explore the environment. As this context is strongly connected with the learning module, it will be explained in section 8.

7.2. Global heuristic policies

The global policy combines the probabilities of each state to be the current state (belief distribution Bel(*S*)) with the best action assigned to each state (local policy *a*(s)*) to select the final action to execute, *a*(**Bel**)*. Once selected the local policy context (for example guidance context, $a^*(s)=a_G^*(s)$), some heuristic strategies proposed by (Kaelbling et al., 1996) can be used to do this final selection.
The simpler one is the "Most Likely State" (MLS) global policy that finds the state with the highest probability and directly executes its local policy:

$$a^*_{MLS}(\boldsymbol{Bel}) = a^*\left(\arg\max_s(Bel(s))\right) \tag{9}$$

The "Voting" global policy first computes the "probability mass" of each action (V*(a)*) (probability of action *a* being optimal) according to the belief distribution, and then selects the action that is most likely to be optimal (the one with highest probability mass):

$$\begin{aligned} V(a) &= \sum_{s\ |_{a^*(s)=a}} Bel(s) \qquad \forall a \in \boldsymbol{A} \\ a^*_{vot}(Bel) &= \arg\max_a(V(a)) \end{aligned} \tag{10}$$

This method is less sensitive to locational uncertainty, because it takes into account all states, not only the most probable one.
Finally, the Q_{MDP} global policy is a more refined version of the voting policy, in which the votes of each state are apportioned among all actions according to their Q-values:

$$\begin{aligned} V(a) &= \sum_{s \in S} Bel(s) \cdot Q^a(s) \quad \forall a \in \boldsymbol{A} \\ a^*_{Q_{MDP}}(Bel) &= \arg\max_a(V(a)) \end{aligned} \tag{11}$$

This is in contrast to the "winner take all" behavior of the voting method, taking into account negative effect of actions.
Although there is some variability between these methods, for the most part all of them do well when initial state of the robot is known, and only the tracking problem is present. If initial state is unknown, the performance of the methods highly depends on particular configuration of starting states. However, MLS or Q_{MDP} global policies may cycle through the same set of actions without progressing to the goal when only guidance context is used. Properly combination of guidance and localization context highly improves the performance of these methods during global localization stage.

7.3. Automatic context selection or combination

Apart from the exploration context, this section considers the automatic context selection (see figure 6) as a function of the locational uncertainty. When uncertainty is high, localization context is useful to gather information, while with low uncertainty, guidance context is the appropriate one. In some cases, however, there is benign high uncertainty in the belief state; that is, there is confusion among states that requires the same action. In these cases, it's not necessary to commute to localization context. So, an appropriate measure of uncertainty is the "normalized divergence factor" of the probability mass distribution, D(V(*a*)), (see eq. 7).

The "thresholding-method" for context selection uses a threshold ϕ for the divergence factor D. Only when divergence is over that threshold (high uncertainty), localization context is used as local policy:

$$a^*(s) = \begin{cases} a_G^*(s) & if \quad D < \phi \\ a_L^*(s) & si \quad D \geq \phi \end{cases} \tag{12}$$

However, the "weighting-method" combines both contexts using divergence as weighting factor. To do this, probability mass distributions for guidance and localization contexts ($V_G(a)$ and $V_L(a)$) are computed separately, and the weighted combined to obtain the final probability mass V(*a*). As in the voting method, the action selected is the one with highest probability mass:

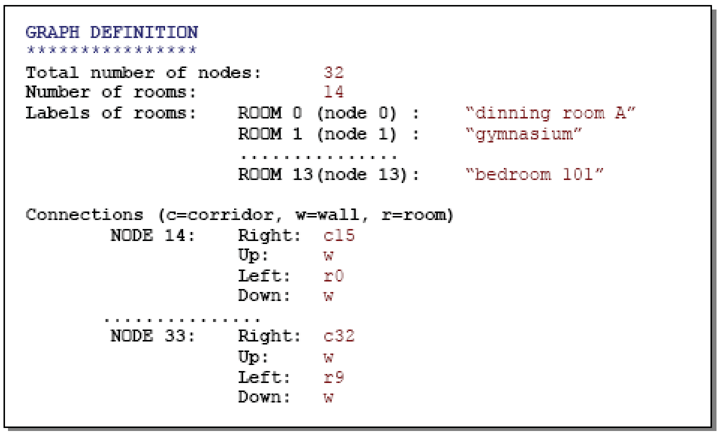
$$\begin{aligned} V(a) &= (1-D)\cdot V_G(a) + D\cdot V_L(s) \\ a^*(Bel) &= \arg\max_a (V(a)) \end{aligned} \tag{13}$$

8. Learning the Markov Model of a New Environment

The POMDP model of a new environment is constructed from two sources of information:

- The topology of the environment, represented as a graph with nodes and connections. This graph fixes the states ($s \in \mathbf{S}$) of the model, and establishes the ideal transitions among them by means of logical connectivity rules.
- An uncertainty model, that characterizes the errors or ambiguities of actions and observations, and together with the graph, makes possible to generate the transition T and observation ϑ matrixes of the POMDP.

Taking into account that a reliable graph is crucial for the localization and planning systems to work properly, and the topological representation proposed in this work is very close to human environment perception, we propose a manual introduction of the graph. To do this, the SIRAPEM system incorporates an application to help the user to introduce the graph of the environment (this step is needed only once when the robot is installed in a new working domain, because the graph is a static representation of the environment).



```
GRAPH DEFINITION
****************
Total number of nodes:        32
Number of rooms:              14
Labels of rooms:     ROOM 0 (node 0)  :     "dinning room A"
                     ROOM 1 (node 1)  :     "gymnasium"
                     ...............
                     ROOM 13(node 13):      "bedroom 101"

Connections (c=corridor, w=wall, r=room)
        NODE 14:     Right:  c15
                     Up:     w
                     Left:   r0
                     Down:   w
       ...............
        NODE 33:     Right:  c32
                     Up:     w
                     Left:   r9
                     Down:   w
```

Fig. 8. Example of graph definition for the environment of Fig. 2.

After numbering the nodes of the graph (the only condition to do this is to assign the lower numbers to room nodes, starting with 0), the connections in the four directions of each corridor node must be indicated. Figure 8 shows an example of the "Graph definition" application (for the environment of figure 2), that also allows to associate a label to each room. These labels will be identified by the voice recognition interface and used as user commands to indicate goal rooms.
Once defined the graph, the objective of the learning module is to adjust the parameters of the POMDP model (entries of transition and observation matrixes). Figure 9 shows the steps involved in the POMDP generation of a new working environment. The graph introduced by the designer, together with some predefined initial uncertainty rules are used to generate an initial POMDP. This initial POMDP, described in next subsection, provides enough information for corridor navigation during an exploration stage, whose objective is to collect data in an optimum manner to adjust the settable parameters with minimum memory requirements and ensuring a reliable convergence of the model to fit real environment data (this is the "active learning" stage). Besides, during normal working of the navigation system (performing guiding tasks), the learning module carries on working ("passive learning" stage), collecting actions and observations to maintain the parameters updated in the face of possible changes.

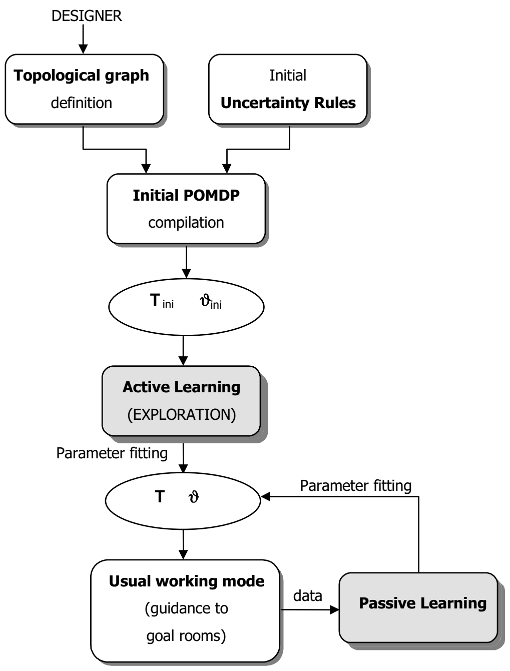

DESIGNER
Topological graph definition
Initial Uncertainty Rules
Initial POMDP compilation
T_{ini} ϑ_{ini}
Active Learning (EXPLORATION)
Parameter fitting
T ϑ
Parameter fitting
Usual working mode (guidance to goal rooms)
data
Passive Learning

Fig.9. Steps for the introduction and learning of the Markov model of a new environment.

8.1. Settable parameters and initial POMDP compilation

A method used to reduce the amount of training data needed for convergence of the EM algorithm is to limit the number of model parameters to be learned. There are two reasons because some parameters can be excluded off the training process:

- Some parameters are only robot dependent, and don't change from one environment to another. Examples of this case are the errors in turn actions (that are nearly deterministic due to the accuracy of odometry sensors in short turns), or errors of sonars detecting "free" when "occupied" or vice versa.
- Other parameters directly depend on the graph and some general uncertainty rules, being possible to learn the general rules instead of its individual entries in the model matrixes. This means that the learning method constrains some probabilities to be identical, and updates a probability using all the information that applies to any probability in its class. For example, the probability of losing a transition while following a corridor can be supposed to be identical for all states in the corridor, being possible to learn the general probability instead of the particular ones.

Taking these properties into account, table 1 shows the uncertainty rules used to generate the initial POMDP in the SIRAPEM system.

TRANSITION MODEL UNCERTAINTIES (F=Follow, L=Left, R=Right, O=Out, E=Enter,N=No action)				
Command	**Effect of Command (% probabilities)**			
a_F	N = 10	F = 70	FF = 10	FFF= 10
a_L	N = 5	L = 90	LL = 5	
a_R	N = 5	R = 90	RR = 5	
a_O	N = 5	O = 85	OF = 10	
a_E	N = 10	E = 90		
OBSERVATION MODEL UNCERTAINTIES				
ASO Model				
Open door probability (for all doors)	50 %			
Prob. of detecting something being nothing	10 %			
Prob. of detecting nothing being something	5 %			
LVO Model				
Room states	Uniform distribution over $o_{LVO}=\{0,1,2\}$			
Corridor states perpendicular to corridor direction and oriented to a door	Uniform distribution over $o_{LVO}=\{0,1,2\}$			
Corridor states perpendicular to corridor direction and oriented to a wall	Uniform distribution over $o_{LVO}=\{0,1\}$			
Corridor states aligned with corridor direction	Uniform distribution over o_{LVO}			
DVO Model				
Room states	Delta distribution in $o_{DVO}=0$			
Corridor states perpendicular to corridor direction	Delta distribution in $o_{DVO}=0$			
Corridor states aligned with corridor direction	Uniform distribution over o_{DVO}			

Table 1. Predefined uncertainty rules for constructing the initial POMDP model.

Figure 10 shows the process of initial POMDP compilation. Firstly, the compiler automatically assigns a number (n_s) to each state of the graph as a function of the number of the node to which it belongs (n) and its orientation within the node (head={0(right), 1(up), 2(left), 3(down)}) in the following way (n_rooms being the number of room nodes):