A Service Robot Named Markovito

Hector Avilés
Department of Information Technologies
Universidad Politécnica de Victoria
Cd. Victoria, Tamaulipas, Mexico
Email: hector_hugo_aviles@hotmail.com

E. Corona, A. Ramírez, B. Vargas, J. Sánchez,
L.E. Sucar and E.F. Morales
Department of Computer Science
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
Tonantzintla, Puebla, Mexico
Email: {esucar,emorales}@inaoep.mx

***Abstract*—The idea of service robots to assist humans in everyday life has been around for many years. Although there is much work in developing different abilities for this kind of robots, little attention has been paid for the integration of these behaviors into a complete functional system. In this paper we present our service robot called Markovito. Markovito integrates different skills to interact with humans and its environment in order to: i) deliver messages and objects between people, ii) search for an object, iii) navigate to different destination points, and iv) follow a person under user requests. Several experiments in real scenarios have shown that Markovito is reliable in the execution of these tasks, while keeping a simple and intuitive robot software architecture able to incorporate new abilities in the near future.**

***Index Terms*—Service mobile robot, MDP, navigation, perception, human-robot interaction.**

I. Introduction

Service robots are mobile robots that help people in different activities, such as helping elderly people in their home, serving as hosts and guides in museums or shopping malls, aiding in hospitals, etc. Such robots need different capabilities to perform their tasks, such as navigation, mapping, localization and obstacle avoidance to move around in an uncertain and changing environment. They also need clear, simple and natural interactive interfaces to exchange information between robots and humans. Developing applications for service robots requires a considerable effort, however, most of them share a core of basic capabilities. It is then natural to develop different modules that can perform such common capabilities and combine them into a general architecture to produce different applications. In this paper we present a service mobile robot called Markovito. Markovito uses a general architecture for building different service robots applications. We have developed several components for performing common tasks, such as, navigation and localization, human tracking, and human-robot interaction. These modules are coordinated with a decision–theoretic controller that serves as an orchestra director. Different service robot applications can be easily constructed by using different configurations of modules and solving a Markov decision problem. Our framework is illustrated in four RoboCup@Home [1] tasks: navigation, follow a human, lost & found and the Open Challenge.

II. Related work

Building service robots to help people has been the subject of recent research. The challenge is to achieve reliable systems that operate in highly dynamic environments and have easy to use interfaces.

RHINO [2] was one of the most successful service robots ever built, designed as a museum tour guide. RHINO successfully navigated a very dynamic environment using laser sensors and interacted with people using pre-recorded information. A person could select a specific tour of the museum by pressing one of many buttons on the robot. RHINO's task planning was specified using an extension to the GOLOG language called GOLEX; GOLEX is an extension of first order calculus, but with the added ability to generate hierarchical plans and a run-time component monitoring the execution of those plans. MINERVA [3] was the successor of RHINO. It differed from RHINO in that it could generate tours of exhibits in real-time as opposed to choosing one of several pre-determined tours. MINERVA also improved on the interaction by incorporating a steerable head capable of displaying different emotional states. The GOLOG language was combined with decision theoretic planners in DTGOLOG, used in the implementation of a service delivery robot [4]. More recently, the robot PEARL escorted elderly people around an assisted living facility [5]. Its navigation and localization used probabilistic techniques with laser sensors. PEARL is more focused on the interaction side with an expressive face and a speech recognition engine. One of PEARL's contributions is the use of a hierarchical partially observable Markov decision process (HPOMDP), which is an extension of hierarchical MDPs (HMDPs) [6] to model uncertain observations. HMDPs use an specified hierarchical decomposition of the domain, and introduce *abstract actions* in higher level MDPs which invoke the policies of lower-level MDPs. HOMER [7] is a messenger robot also based in MDPs. The authors introduce Multiply Sectioned Markov Decision Processes (MS-MDPs) as a mechanism for efficient task coordination. The robot's task is partitioned into a number of subtasks, each assigned to an MDP, such that each one can be specified and solved independently; they all are executed concurrently, coordinated implicitly by common state variables.

Contrary to previous approaches, that have focused on solving a single application; in this work a 3-layer architecture, several software modules with novel ideas to perform general purpose tasks, and an MDP framework to act as coordinator are proposed to efficiently build different service robots applications.

III. Software Architecture

Crucial to the design of a human-interactive mobile robot is the ability to rapidly and easily modify the robot's behavior. This requires for a mobile robot to have a modular software architecture, with a planning module that coordinates the different software modules to achieve the goal.

Our software architecture is based on a Behavior-based architecture [8]. A behavior is an independent software module that solves a particular problem, such as navigation or face detection. In this paper behaviors are also refereed as modules. Behaviors exist at 3 different levels:

- Functional level: The lowest level behaviors interface with the robot's sensors and actuators, relaying commands to the motors or retrieving information from the sensors.
- Execution level: Middle level modules perform the main functions, such as navigation, localization, speech recognition, etc. These interface with the lowest level through a shared memory mechanism. Each middle level module computes some aspect of the state of the environment. The outputs of these modules are typically reported to the highest level modules.
- Decision level: The highest level coordinates the middle level modules based on a global planner. The planner is based on a Markov decision process (MDP). The MDP is solved to obtain an optimal policy according to the tasks' objectives, which is used to command the other modules.

This architecture can be implemented in a distributed platform, such that each level and each module within a level could be on a different processor. A transparent communication mechanism permits different configurations without need to modify the modules. Thus, some processing could be done on board the robot (lower level modules) and other off board (high level modules). Also a module can be changed without affecting the rest of the system.

IV. Coordinator

Markov decision processes (MDPs) have become the semantic model of choice for decision theoretic planning (DTP) in the AI community [9]. They are simple for domain experts to specify, or can be learned from data. They have many well studied properties including approximate solution and learning techniques. An MDP is a tuple $\{\mathcal{S}, \mathcal{A}, \Pr, R\}$, where $\mathcal{S}$ is a finite set of states and $\mathcal{A}$ is a finite set of actions. Actions induce stochastic state transitions, with $\Pr(s, a, t)$ denoting the probability with which state t is reached when action a is executed at state s. $R(s, a)$ is a real-valued reward function, associating with each state s and action a. Solving an MDP is finding a mapping from states to actions. Solutions are evaluated based on an optimality criterion such as the expected total reward. An optimal solution is one that achieves the maximum over the optimality measure, while an approximate solution comes to within some bound of the maximum.

The space and time complexity of MDPs increases with the number of states. This problem can be reduced by using factored representations [10], in which the state is decomposed in a set of variables or factors, and the transition functions are represented using a factored representation (a two-stage dynamic Bayesian network). In this paper we use a factored representation to specify the MDPs and SPUDD [11] to solve them.

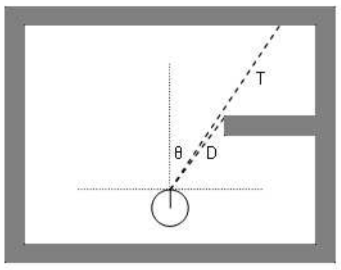


Fig. 1. An example of a discontinuity extracted from laser scanner data. Their is a *large* difference between on laser reading (T) and the next (D); B denotes the angle between the robot's front and the discontinuity.

The combination of a coordinator based on MDPs with a 3-level architecture can reduce the development costs of different service robots applications. By simply re-arranging the modules, changing the goal (reward), and solving a new MDP, different applications can be developed. Different software modules of common tasks can be incorporated in a transparent way. In the following sections, some of the general modules that have been developed are briefly described.

V. Map Building and Localization

A mobile robot requires a model or map of its environment to perform tasks. Our map building module uses information from a laser scan, its odometer and a sonar ring to construct an occupancy cells map using particle filters. Due to space constraints, this module is not described in detail (see [12] for more information).

The ability for mobile robots to locate themselves in an environment is not only a fundamental problem in robotics but also a pre-requisite to many tasks such as navigation. There are two types of localization problems: local and global. Local localization techniques aim to compensate for odometric errors during navigation and require information about the initial position of the robot. Global localization aims to locate the robot's position without prior knowledge of its current location. These problems are particularly hard in dynamic environments where new obstacles can corrupt the robot's sensor measurements [2].

In order to locate itself either during navigation or globally, this module uses natural landmarks that have the advantage that the environment does not need to be transformed. In this work, discontinuities are used, that can be easily extracted from laser scanner data with high accuracy, to solve the local and global localization problem. A discontinuity is defined as an abrupt variation in the measured distance of

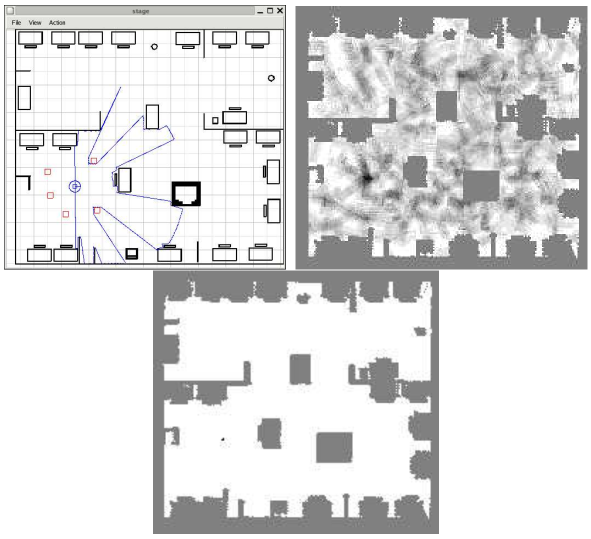


Fig. 2. Global localization. Top left: a simulated environment, showing the robot and some discontinuities. Top right: results of the 1st. stage, where potential locations are shown as gray dots in the map. Bottom: the robot is localized after the 2nd. stage, its position is depicted as a black point near the center of the lower-left “room” in the map.

two consecutive readings of the laser, as shown in Figure 1. Given a set of landmarks (discontinuities), a triangulation process is performed between all the visible landmarks to estimate the robot’s position. The information from all the visible landmarks is combined considering the angle between landmarks, the distance between the robot and its farthest landmark, and if there are landmarks at both sides of the robot or only one, to give more accurate estimates.

For the global localization problem, a ray tracing approach is used to simulate laser readings on the map. Each cell is associated with all its visible landmarks and their values. This process is performed off-line once a map is constructed. To match a cell with the current readings of the robot, an initial stage filters out a large number of candidate positions with a fast algorithm. It counts the number of discontinuities obtained from the laser data that matches the distance, depth, and orientation of the previously stored discontinuities associated to each cell. A modified discrete relaxation algorithm is used in a second stage to determine the similarity of each cell with the observations of the robot, considering the distances between the discontinuities in this stage. Our global localization algorithm is able to locate the robot even with new obstacles, as can be seen in Figure 2, where five new obstacles are added to the environment (see [13] for more details).

VI. Navigation

We have developed different navigation modules. Initially, we implemented a navigation module that uses a dynamic programming algorithm, with exponential costs near obstacles, to find the least expensive path. In order to avoid new obstacles, the robot is sensing its environment while moving.

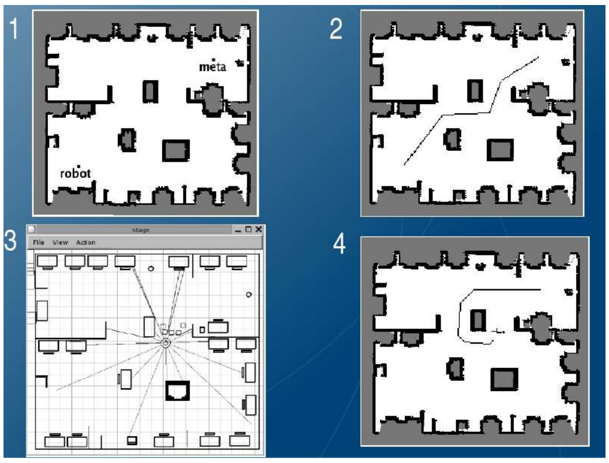


Fig. 3. Re-planning with obstructed paths. (1) shows the start position (robot) and the goal (meta). (2) depicts the initial route obtained by the navigator. (3) shows the simulated robot (near the middle) encountering that a door has been blocked. In (4) we see the new route to the goal.

In case a new obstacle is placed in front of the robot the module finds an alternative path, as shown in Figure 3 (see [14] for more details).

In this paper, a novel navigation strategy using machine learning techniques is presented. To illustrate more clearly the motivation behind this work, imaging going to a new place, e.g., a conference site. You would normally ask for directions of places of interest, like the registration desk or a toilet, and you will get general directions, like “at the end of the aisle to your right” or possibly “in room 203”. You will have then to navigate without collisions in an unknown and dynamic environment to a particular destination point through well known natural marks like walls and doors and expected dynamic conditions, like walking people. Imagine you want your robot to learn how to perform a similar skill. You place your robot in an unknown environment and you want it to navigate to a particular point, like a charging station, but the robot is only given the general direction of its destination point. The robot has to learn how to perform simple skills, like obstacle avoidance and orientation towards a goal, and use them to safely go to a particular goal in a dynamic and unknown environment. This is the approach follow in this paper for the navigation module.

The application of machine learning techniques with expressive representation languages to domains like robotics have received little attention due to the huge amount of low level and noisy data produced by the sensors. The use of relational representations in this area are novel, and their advantages have just recently being addressed (e.g., [15]). In this paper, first-order logic relations are learned to build the navigation module. This module consists of a set of reactive rules that sense the environment continuously and apply actions whose continuous execution eventually satisfy a goal condition. These rules are known as TOPs (Teleo-OPerators) [16], an effective framework to achieve goals when unexpected events occur.

The objective of our learning process is to provide a mobile robot with abilities to move through indoor environments and to accomplish goals. In order to learn TOPs, we combine three machine learning techniques: (i) behavioural cloning, (ii) inductive logic programming (ILP), and (iii) a simple grammar learning algorithm. Behavioural cloning is a technique to learn skills from examples [17]. The key idea of this method is to show the robot what to do instead of how to do a task. For instance, in order to learn how to navigate without collisions, the robot is presented with human traces steering the robot and avoiding obstacles, simplifying the programming effort.

We introduced a two phase learning process to learn TOPs for mobile robots in indoor environments: (i) learning of basic TOPs, and (ii) learning of complex TOPs. In the first phase, a system called TOPSY uses human-guided traces and the natural landmark identification process, described in the previous section, to reduce the information from the sensors into a small set of ground predicates (landmarks) suitable for an ILP system called ALEPH [18]. A small set of background knowledge is given to the system from which other predicates are learned and used in the induction of simple TOPs.

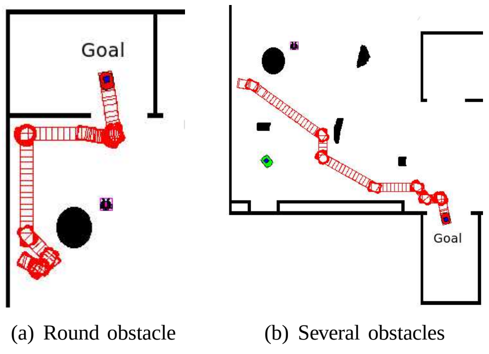
TOPSY was able to learn the following TOPs from human traces:

- *avoid(State,Action)*: to wander around without collisions. *Action* can take the following values: *go_forward*, *turn_left*, and *turn_right*.
- *orient(State,Action)*: Given a target point, the robot has to turn until it is oriented towards the goal, only if it is located in a *safe_turn* zone. *Action* can take the following values: *turn_right*, *turn_left* and *nothing*.

In order to learn how to combine TOPs to perform more complex tasks (e.g., the goto TOP), previously learned TOPs that apply to states in traces are identified, returning high-level traces of applicable TOPs. The goal is to learn a grammar with TOPs able to reproduce a set of traces. With this purpose, FOSeq (First Order learning from Sequences), an algorithm to learn grammars from sequences based on association rule learning is introduced and applied to induce the goto TOP. This algorithm is in its initial stage, however, the learned TOPs were used for navigation tasks on different environments with dynamic objects. Figure 4 shows some examples of the *Goto* TOP under different scenarios.

A. Planning

The previous TOPs are used with a probabilistic road map module that returns collision free paths. A probabilistic road map (PRM) [19], [20] is built using a random generation of points in the (free) configuration space. Each point is joined to its neighbors if there is a straight free path between them, and the information is stored in a graph, G. Given an initial configuration s and a goal configuration g, the idea for path planning is to connect s and g by finding a path in G. Given the random nature of the algorithm, some paths in G may be too long and irregular, so a smoothing algorithm, trying to find shortcuts, is applied. This process is illustrated in figure 5. The points (landmarks) in the path are given as intermediate goals



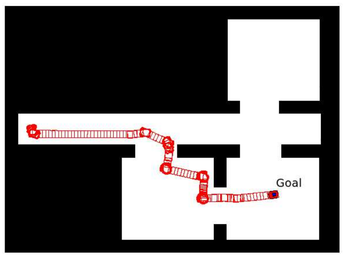


Fig. 4. The “Goto” TOP guides the robot to the goal position in different simulated scenarios.

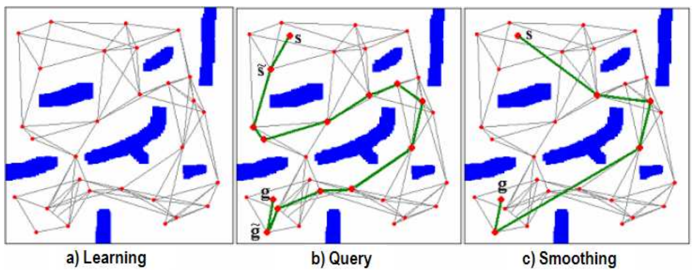


Fig. 5. The different stages of the probabilistic road map (PRM). (a) shows the learning phase, in which the graph is bulit. (b) depicts an initial path between the start position (s) and the goal (g). (c) shows the fi nal path after the smoothing phase.

to the navigation algorithm. In this way, the combination of PRMs and TOPs produces a robust navigation module.

VII. Perception

Service robots require a sophisticated perception to operate in complex and dynamic environments. A multimodal approach is used, combining several types of sensors, such as vision, sonar, laser, and sound. Some perception capabilities are integrated within specific modules, while others are developed as specific modules such that they could be used by different subsystems. Next we describe the human tracking and object recognition modules.

A. Human Tracking

Human body tracking is important for the interaction between humans and service robots. In particular, several tasks require the robot to follow a human to a particular destination point. In this module we extract torso boundaries using a histogram and the back projection image [21] coupled with

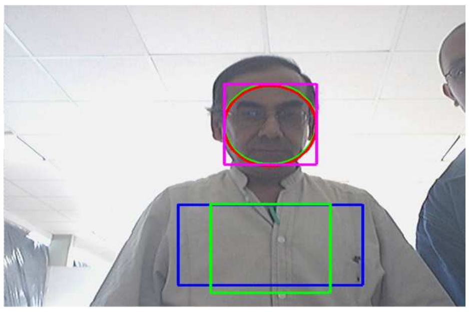


Fig. 6. Face and torso detection. The face of the person is shown as a white circle and the torso as a black rectangle.

Haar functions [22] with a monocular camera. We estimate the user position relative to the robot applying a distance transform. Our torso detection and tracking system is divided in two stages. The first stage is the torso localization process, that uses a face detection algorithm based on color histograms in RGB. Once the face is detected, the torso position is estimated based on human biometry [23]. The color histogram of the torso is registered by this module. The second stage consists of tracking the torso using the color histogram obtained at the first stage, coupled with detectors based on motion and appearance information. Finally, a distance transform is applied, considering a pinhole camera model.

Based on the torso's color histogram obtained in the first images, the application of the Camshift algorithm [21] with this histogram as region of interest, produces a torso area estimation. Knowing the distance l between the user and the camera, the focal length f and a real torso area estimation X, the pixel size p (typically 4-12 μm) is determined, using Equation (1):

$$p = \frac{f \cdot X}{(f - l) \cdot x} \tag{1}$$

where, x is the image area of the torso.

Once the torso region has been detected in an image sequence, the next step consists of performing torso tracking. For torso tracking we have to consider if a region detected using the Camshift algorithm [21] is coincident with a region detected using Haar detectors [22]. In this case, we obtain its camera distance, using Equation (2), we calculate its center point and finally its position by simple geometry.

$$z = f \cdot \left(1 - \frac{X}{x \cdot p}\right) \tag{2}$$

Figure 6 shows the torso recognition system.

B. Object Recognition

For object recognition we have implemented a module based on the SIFT algorithm (see [24] for more details). This module can be easily trained to recognize any object, by just showing the robot the object in front if its camera for a few seconds. The object model, based on SIFT features, is stored in an object database; and, if it is later seen by the robot, it can be recognized.

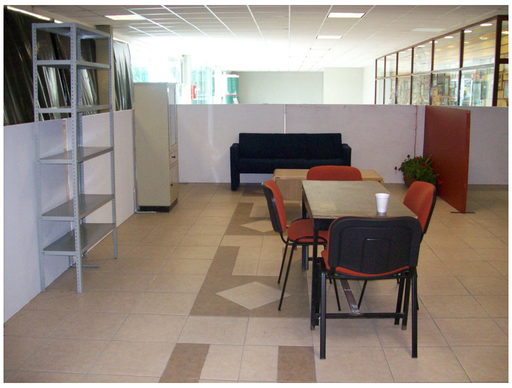


Fig. 7. The environment used for the RoboCup@Home tasks.

VIII. Speech Recognition and Synthesis

For speech synthesis, the Festival System [25], with a Spanish dictionary, was used. For the speech recognition a language model was used to restrict the combination of words according to the vocabulary for the different tasks. This language model was obtained via a series of "Wizard of Oz" experiments, where different persons talked to a simulated robot (the Wizard). The phrases were recorded, and from them the statistics to build a probabilistic language model were estimated. The robot recognizes keywords of the phrases which are given as inputs to the coordinator.

IX. Applications

The previously described modules can be combined to perform different service robot tasks. As mentioned in Section IV, the different modules are coordinated with an MDP. In particular, Markovito was applied to four tasks of the RoboCup@Home competition: navigate, lost & found, follow a human, and the open challenge (deliver messages and objects). Markovito first built a map of the environment (see Figure 7)[1]. Next we describe each one of the 4 tasks, in particular the MDP that coordinates the modules to perform the task.

A. Navigate through three places and return to the starting position

In this task, Markovito has to navigate safely through three places, that are given by a user at the beginning, and then return to its starting position. As soon as the robot reaches a destination point it has to announce it. All the interaction is made through natural language.

Six variables were defined for this MDP: *localize* indicates when the robot's position is know, *get goals* when the robot has obtained the three places to visit, *arrived at destination point*, *has trajectory* for navigation, *end position* when it returns to the starting position.

[1]This is the *home* environment used at the Mexican RoboCup@Home competition that took place in Puebla, Mexico, August 2007, see: http://www.upaep.mx/cmr2007/

TABLE I
THE SIX ACTIONS USED TO COORDINATE THE MODULES FOR THE NAVIGATION TASK.

Action	Module	Description
Localize	Navigation	Global localization
Wait for goals	Speech	Obtain the places to visit
Generate trajectory	Navigation	Path to a destination place
Go to next goal	Navigation	Navigate to reach the next place
Arrival to goal i	Speech	Announce that the robot has reached a new place
Announce finish	Speech	Announce that the robot has returned to the initial position

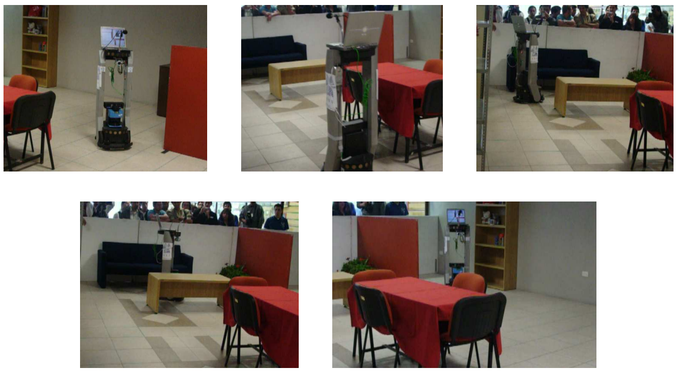


Fig. 8. Markovito navigating through the environment reaching different destination points.

The MDP has six actions to coordinate the modules which are described in Table I. Each action calls one of the previously described modules.

Figure 8 shows a sequence of Markovito navigating through the environment reaching different destination points.

B. Lost & Found

In this task, Markovito is shown an object which later has to search for. The user places several objects, one by one, in front of the camera. After wards the user indicates the robot which object to search for and a destination area. Markovito has to navigate to the indicated area and search for the object. It sends a recognition message when the object is recognized.

This MDP has 11 variables: *localize, has order, forget object, confirm forgetting, learn object, confirm learning, object in data base, found, reached destiny, is near*, and *confirm object*.

The actions consider in this MDP are: *global localization, wait for order, forget object, learn object, confirm learn object, confirm forget object, go to search area, look for object, get close to object*, and *confirm found object*. All the interactions in this task are in natural language. This task uses the localization, navigation, planning, speech and perception modules.

Figure 9 shows Markovito learning an object, in this case a Teddy bear (left) and then searching for it (right).

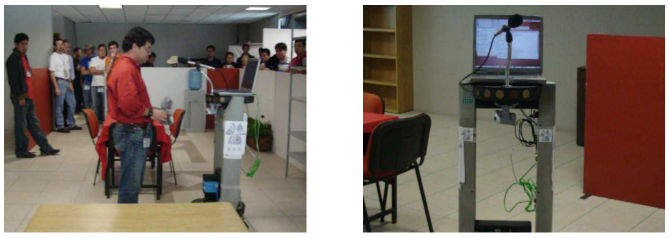


Fig. 9. Markovito learning to recognize an object (left) and later searching for it (right).

TABLE II
THE ACTIONS USED IN THE TASK TO FOLLOW A PERSON AND THE DIFFERENT MODULES USED IN THIS TASK.

Action	Module	Description
Localize	Navigation	Global localization
Wait order to calibrate	Speech	Wait for a calibration order
Calibrate	Vision	Get information about the person
Wait order to follow	Speech	Wait for a follow order
Follow	Navigation	Follow a person
Announce lost person	Speech	Announce that the robot lost the person
Search the person	Vision	Search for a person
Announce found person	Speech	Announce that the robot found the person
Wait order to stop	Speech	Wait for a stop order
Generate trajectory	Navigation	Give a path to destination place
Return	Navigation	Navigate to the starting position
Announce arrive	Speech	Announce arriving to the starting position

C. Follow a person under user request

For this task, Markovito has to follow a human through an unknown track in a home-like environment. After reaching the end position, the robot has to return to its starting position. The task consisted of two stages. In the first step, the human stands in front of the robot at a distance of one meter for about one minute for calibration. At this state the torso detection module was used. In a second step, the human starts walking towards the end position, passing through a number of places. This task used the navigation module based on TOPs to go dynamically to the different places given by the torso tracking module.

This MDP has 13 variables: *localize, person, calibrate order, follow order, stop order, calibrate, follow, searching, announce following, announce searching, announce finish, has trajectory*, and *get destination*. All the interaction is also in natural language. Table II shows the actions that are used to coordinate the different modules.

D. Deliver messages and objects between people

The goal in this task is to receive and to deliver a message, an object or both, under user requests. The interaction again is through natural language. The user gives an order to send a message/object and the robot asks for the name of the sender

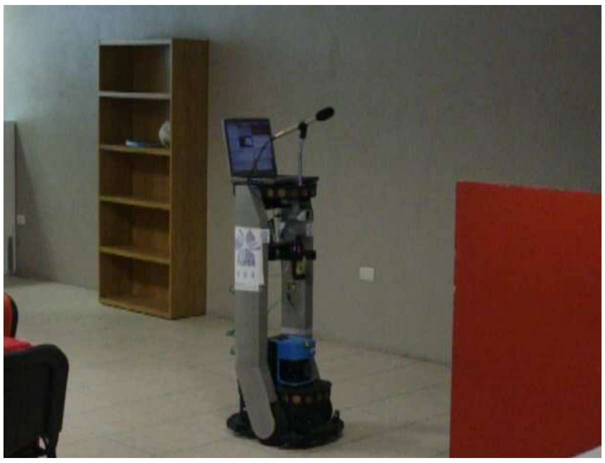


Fig. 10. Markovito delivering a beer in the RoboCup@Home scenario.

and the receiver. Markovito either records a message or uses its gripper to hold an object, and navigates to the receiver's place and deliver the message/object.

An MDP with 12 binary variables was used for this task: *localized, greetings, sender, receiver, message, object, trajectory, reached destiny, found receiver, object/message delivered, message*, and *object*. Two multi-valued variables were also used for the battery level and for the type of delivery.

Thirteen actions were defined for this task: *localize globally, generate path to destination place*, and *execute trajectory*, as part of the navigation module. *Wait for greeting, request type of delivery, request sender's name, request receiver's name, record message, deliver message* and *confirm delivery*, as part of the speech module. *Grasp object* and *release object* as part of the delivery module. *Detecting a person* as part of the vision module. Figure 10 shows Markovito delivering a beer to a person in the home scenario.

X. Conclusions and Future Work

A general framework for creating service robots applications has been described. This framework uses several general modules to perform common service robot tasks Some of these modules include some novel ideas, in particular the localization, navigation and human tracking modules. The architecture uses an MDP framework to coordinate these behaviors. Different service robot applications can be easily constructed by combining the basic modules, and defining a new MDP to obtain an optimal policy for each task. We have illustrated the capabilities of this framework with four different applications defined in the RoboCup@Home competition.

There are several research directions to follow. In particular, we plan to incorporate a module that can be used to teach the robot by example, to incorporate a face recognition module, and to combine gesture recognition with the natural language module to enhance our human-robot interface. We are also improving the coordinator to deal with conflicting situations. Finally, we plan to use this framework for developing other applications.

References

[1] "The robocup@home webpage." [Online]. Available: http://www.ai.rug.nl/robocupathome/

[2] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Proceedings of the Fifteenth National Conference on Artifi cial Intelligence (AAAI '98)*, Madison, Wisconsin, July 1998.

[3] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'99)*, Detroit, Michigan, May 1999.

[4] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun, "Decision-theoretic, high-level agent programming in the situation calculus," in *Proceedings of the AAAI National Conference on Artifi cial Intelligence (AAAI '00)*, 2000.

[5] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma, "Experiences with a mobile robotic guide for the elderly," in *Proceedings of the AAAI National Conference on Artifi cial Intelligence (AAAI '02)*, Edmonton, Canada, 2002.

[6] T. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Journal of Artifi cial Intelligence Research*, vol. 13, pp. 227–303, 2000.

[7] P. Elinas, L. Sucar, J. Hoey, and A. Reyes, "A decision-theoretic approach for task coordination in mobile robots," in *Proc. IEEE Robot Human Interaction (RoMan)*, Japan, September 2004.

[8] R. Arkin, *Behavior-based Robotics*. MIT Press, 1998.

[9] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY.: Wiley, 1994.

[10] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artifi cial Intelligence*, vol. 101, no. 1-2, 1998.

[11] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier, "Spudd: Stochastic planning using decision diagrams," in *Proc. of the 15th Conf. on Uncertainty in AI, UAI-99*, 1999, pp. 279–288.

[12] V. Jaquez, "Construcci´on de mapas y localizaci´on simult´anea con robots m´oviles," Master's thesis, ITESM Campus Cuernavaca, Mexico, 2005.

[13] S. Hern´andez and E. Morales, "Global localization of mobile robots for indoor environments using natural landmarks," in *Proceedings IEEE Conference on Robotics, Automation and Mechatronics (RAM-2006)*, 2006.

[14] S. Hern´andez, "Navegaci´on de un robot m´ovil en ambientes interiores usando marcas naturales del ambiente," Master's thesis, ITESM Campus Cuernavaca, Mexico, 2005.

[15] A. Cocora, K. Kersting, C. Plagemann, W. Burgard, and L. De Raedt, "Learning relational navigation policies," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[16] S. Benson and N. J. Nilsson, "Reacting, planning, and learning in an autonomous agent," *Machine Intelligence*, vol. 14, pp. 29–62, 1995.

[17] D. Michie and C. Sammut, "Behavioral clones and cognitive skill models," *Machine Intelligence*, vol. 14, pp. 395–404, 1995.

[18] A. Srinivasan, "The aleph 5 manual http://web.comlab.ox.ac.uk/oucl/research/areas /machlearn/ aleph," 2005.

[19] J. Barraquand and J. C. Latombe, "Robot motion planning: A distributed representation approach," *Journal of Robotics Research*, 1991.

[20] J. C. Latombe, *Robot motion planning*. Kluwer Academics Publishers, 1991.

[21] G. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, 1998.

[22] D. Snow, M. Jones, and P. Viola, "Detecting pedestrians using patterns of motion and appearance," in *Ninth IEEE International Conference on Computer Vision (ICCV 2003)*, 2003, pp. 734–741.

[23] H. Aviles, L. E. Sucar, and C. Mendoza, "Visual recognition of similar gestures," in *The 18th International Conference on Pattern Recognition (ICPR-2006)*, 2006.

[24] D. Lowe, "Distint image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[25] "The festival speech synthesis system." [Online]. Available: http://www.cstr.ed.ac.uk/projects/festival/