COACHES
Cooperative Autonomous Robots in Complex and Human Populated Environments

L. Iocchi[1] ✉, M. T. Lázaro[1], L. Jeanpierre[2], A.-I. Mouaddib[2], E. Erdem[3], H. Sahli[4]

[1] DIAG, Sapienza University of Rome, Italy
[2] GREYC, University of Caen Lower-Normandy, France
[3] Sabanci University, Turkey
[4] Vrije Universiteit Brussel, Belgium

Abstract. The deployment of robots in dynamic, complex and uncertain environments populated by people is gaining more and more attention, from both research and application perspectives. The new challenge for the near future is to deploy intelligent social robots in public spaces to make easier and safer the use of these spaces. In this paper, we provide an overview of the `COACHES` project which addresses fundamental issues related to the design and development of autonomous robots to be deployed in public spaces. In particular, we describe the main components in which Artificial Intelligence techniques are used and integrated with the robotic system, as well as implementation details and some preliminary tests of these components.

1 Introduction

Public spaces in large cities are becoming increasingly complex and unwelcoming environments because of the overcrowding and complex information in signboards. It is in the interest of cities to make their public spaces easier to use, friendlier to visitors and safer to increasing elderly population and to the people with disabilities. In the last decade, we observe tremendous progress in the development of robots in dynamic environments populated by people. There are thus big expectations in the deployment of robots in public areas (malls, touristic sites, parks, etc.) to offer services to welcome people in the environment and improve its usability by visitors, elderly or disabled people.

Such application domains require robots with new capabilities leading to new scientific challenges: robots should assess the situation, estimate the needs of people, socially interact in a dynamic way and in a short time, with many people, the navigation should be safe and respects the social norms. These capabilities require new skills including robust and safe navigation, robust image and video processing, short-term human-robot interaction models, human need estimation techniques and distributed and scalable multi-agent planning.

Fig. 1. COACHES environment and robot.

The main goal of the COACHES project (October 2014 - September 2017) is to develop robots that can suitably interact with users in a complex large public environment, like a shopping mall. Figure 1 shows the *Rive de l'orne* shooping mall in Caen (France) where the experimental activities of the project will be carried out, as well as a prototype of the robot that will be used.

Previous work on social robotics and human-robot interaction mostly focused on one-to-one human-robot interaction, including elderly assistance (e.g., GiraffPlus project [5]) and interaction with children (e.g., MOnarCH project [6]). Robots acting as museum tour-guides have also been successfully experimented. One of the first robot interacting with many non-expert users was RHINO deployed at the "Deutsches Museum" in Bonn, Germany [2]. In this work, the main focus was in the mapping, localization and navigation abilities in crowded environment, while human-robot interaction was limited to buttons on the robot, a remote Web interface and pre-recorded sentences issued by the robot.

As shown in the figure, in contrast with previous work, the COACHES environment is very challenging, because populated by many people. Moreover, we aim at a more sophisticated interaction using multiple modalities (speech, gesture, touch user interfaces) and dialog generated on-line according to the current situation and the robot's goals. Consequently, the required level of "intelligence" of the COACHES robots in order to adequately perform complex and effective tasks in this environment in presence of people is much higher than in previous projects.

The proposed methodology to reach the project goals is based on integration of Artificial Intelligence and Robotics. In this paper, we describe the main overall architecture of the system (Section 2) and the components related to Artificial Intelligence and Robotics (Section 3): 1) knowledge representation, 2) planning under uncertainty, 3) hierarchical plan execution and monitoring. Section 4 provides some examples and Section 5 concludes the paper.

2 Software Architecture

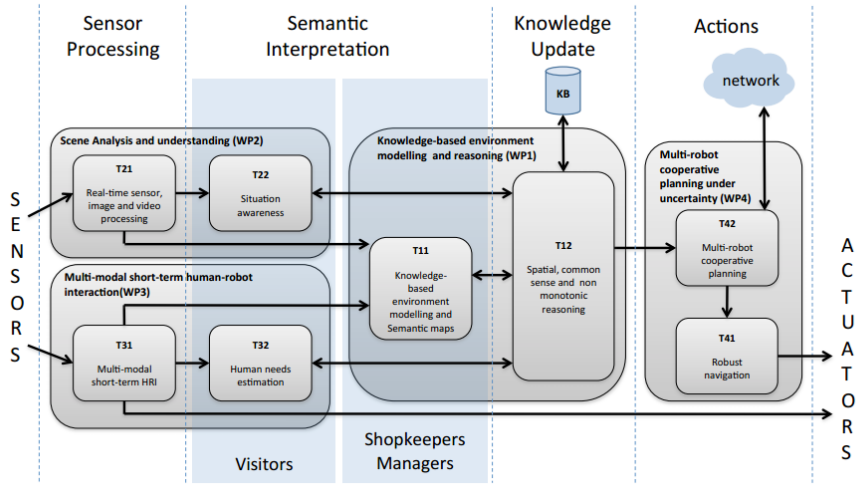

Fig. 2. COACHES software architecture

The software architecture of the COACHES robots is shown in Figure 2). An open architecture (hard/soft) and standard technologies available will be used, so that it will be easy to extend and/or adapt the capabilities of the system during the whole length of the project (especially to integrate and test various algorithms and/or sensors). Such an open architecture will also simplify and optimize integration efficiency as well as re-use of assets in other projects or products.

The main software components that will be developed for control, reasoning and interaction functionalities of the system are listed below.

- *Scene analysis*, including sensor processing procedures for both on-board robot devices and static sensors in order to determine the current situation and understand events that are of interest for the system.
- *Multi-modal HRI*, defining a set of modalities for human-robot interaction, including speech recognition and synthesis, touch interaction, graphical interface on a screen mounted on the robot and Web interfaces.
- *Knowledge-based representation and reasoning*, defining the formalism and the procedure to represent and reason about the environment and the task of the robots.
- *Planning and execution monitoring*, for generating the plans to achieve the desired goals and monitor their execution for robust behaviors.
- *Safe navigation*, for guaranteeing safety operations of the robot in a populated environment.

3 Artificial Intelligence components

While the overall software architecture described before integrates several components that are all important for the development of the project, in this paper we focus on the modules that implement a proper integration between Artificial Intelligence and Robotics techniques. Thus, in this section, we will describe the three main components that allow the robots to: 1) represent and reason about the environment, 2) generate the plan to reach their goals; 3) monitor the execution to overcome failures.

3.1 Knowledge base representation and reasoning

The knowledge base (KB) is used to model both static knowledge (e.g., the semantic map of the environment and the common sense information) and the dynamic knowledge (e.g., human activities) coming from different units, such as the perception modules of the architecture, particularly the multi-modal HRI interface and the image processing modules. From these information, the reasoning module is able to infer the list of possible tasks to accomplish. This list of tasks is then sent to the decision module (described in the following section) that will compute the policy to accomplish them.

Although there are many existing approaches to semantic representations of the environment (see [7] for a survey), a standard formalism does not exist. In this section, we thus define the main features of the knowledge base used in the project, based on experience in previous work [1, 3]. We first introduce the semantic labels used to describe elements of the world, then predicates that determine relations among these labels, and finally its application to the use case in the project.

Semantic labels. In order to refer to objects and classes of objects in the knowledge base, we introduce a set of labels that will be associated to semantic meanings.

A first set of these labels are called *Concepts*. Concept labels are associated to classes of objects and in this paper they are denoted with an uppercase initial letter. For example, *Restaurant* is a concept used in the semantic map to denote the class of restaurants in the shopping mall. These concepts are organized in a hierarchical way according to the "is-a" relation. In this way, we can express, for example, that the concept *FrenchRestaurant* is a *Restaurant*.

A second set of labels will be used to denote objects. Each object belongs to a concept implementing the relation "instance-of". Object labels are denoted with lowercase letters. Thus, a particular restaurant in the mall will be denoted with an object label *caféMarcel* that will be an instance of the concept *FrenchRestaurant*.

Predicates. Predicates are used to describe relations among the semantic labels. For example, the "is-a" and the "instance-of" relations can be represented by the corresponding predicates **is-a** and **instance-of**. Predicates are also used to

denote properties of objects or locations (e.g., the status of a door or the presence of air-conditioned in a shop).

For representing the *Rive de l'orne* shopping mall, we consider different types of areas: shops, restaurants, halls, corridors, rest areas, offices, toilettes, etc. For shops, services and restaurants we consider different categories:

- *Shop categories*: dress shop, women dress shop, kid dress shop, men dress shop, makeup store, store perfume, sport store, etc.
- *Restaurant categories*: French, Japanese, Chinese, Italian, Oriental, African, fast-food, etc.
- *Service categories*: security, information, health-care, etc.

All these areas are represented as concepts that are grouped in a more general concept *Area*. The hierarchy of these areas will be defined through the "is-a" relation of the semantic labels described before.

Some examples of predicates for representing the shopping mall are:

is-a(FrenchRestaurant, Restaurant)
instance-of(caféMarcel, FrenchRestaurant)
connect(door12, hall, caféMarcel)
open(door12)
airconditioned(caféMarcel)

Reasoning. The KB is used by reasoning processes that define the goals for the `COACHES` robots. To this end, the reasoning engine takes into account the available information in the KB related to: semantic map, common-sense knowledge, and dynamic knowledge coming from the scene analysis and HRI modules. With this input, this module determines which goals for the system are consistent with the current situation. Then these goals are passed to the Planning module described below.

A further function of reasoning on the KB is to determine conditions for plan execution that are derived from direct perception. In this way, plan execution can consider properties not directly observable from perception, but coming from a reasoning process that interpret perception with common-sense reasoning.

3.2 Planning under uncertainty

In this section we describe the Markov Decision Process (MDP) used to model the `COACHES` planning domain and the algorithm implemented for computing the optimal policy.

Task structure. The result of the reasoning module (KB module) is a set of goals $G = \{g_1, g_2, \ldots, g_k\}$ concerning advertisement, patrolling, assisting and escorting. We note also that advertising goals could be performed in parallel with the moving ones. Consequently, the task structure is a hierarchy of modules to execute. This structure is inspired by progressive processing units [4], that we

name PRU+. In our application, we define four PRU+. Each PRU+ is composed of levels where the first level concerns the execution of the subtask GOTO SITE X, the second level concerns the advertisement at a location (x, y) and the third level consists of DO TASK X where X could be the assistance, the patrolling, the escorting or the surveillance. With such task structures we can also define some joint goals requiring joint PRU+. For example, escorting a people from one location in a building to another location in the other building requires a cooperation between robots. Indeed, the first robot executes a policy of PRU+ for escorting a user to the exit of the first building, provide him/her information to reach the other building and then send information to the other robots in the other building to continue the escorting task at the second building. The structure of tasks we propose for single robot tasks is {GOTO x, ADVERTISEMENT, DO x}, while for the joint task is {GOTO x, ADVERTISEMENT, INFORM PEOPLE, SEND MESSAGE TO THE OTHER ROBOTS}. The task DO x concerns different tasks of assistance.

More formally, a PRU+ is defined by a set of levels $\{l_1, l_2, \ldots, l_k\}$, where each level l_i is composed by a set of modules $\{m_i^1, m_i^2, \ldots, m_i^{p_i}\}$ and each module m_i^j is defined by different execution outcomes that we name options $\{\alpha_i^j, \beta_i^j, \ldots\}$.

MDP definition and planning. The planning procedure consists of formalizing the robot activities as an MDP using the PRU+ task definition. This procedure is based on two steps: 1) generating an MDP from a PRU+, 2) compute the optimal policy for the generated MDP. In the following we define the $MDP =< S, A, R, T >$ where :

- S is a set of states defined by $x = [l, m, o, v]$ where l is the level of the PRU, m is a module of the level l, o is an option of module m and v are state variables defining the execution context representing the subset of variables to be considered for the option o.
- A is the set of actions consisting of execution of one module of the next levels E or skipping the level S.
- T is the transition function defined as follows :
  - $Pr([l+1, m', o', v']|[l, m, o, v], E) = p(o')$, this means when execution module m' at state $[l, m, o, v]$ we move to state $[l', m', o', v']$ with probability p(o') representing the probability to get the outcome o'.
  - $Pr([l+2, m', o', v']|[l, m, o, v], S) = 1$, this transition is deterministic because we skip level $l+1$ and we move to level $l+2$.
- R is the reward function related to the options assessing the benefit to get the outcome;

From this definition, the Bellman equation for our model becomes

$$V(x) = R(o) + max_{E,S} \sum_{x'} Pr(x'|x, a)V(x')$$

The optimal policy π is computed by a standard MDP solving algorithm based on value-iteration. Moreover, in this algorithm, a tabu-list of actions is

used to choose or drop actions to be inserted in the policy. This tabu-list is built and updated by the Model updater module, described below in this section, representing the actual feedback coming from the execution layer.

3.3 Plan execution and monitoring

Plan execution monitoring and interleaving planning and execution are crucial features for an autonomous robot acting in a real environment, specially when human interaction is involved, as for the `COACHES` robots. Indeed, in complex scenarios, it is not possible to model and foresee all the possible situations that may occur, consequently plans generated off-line (i.e., before the actual execution of the task), when several information about the real environment are not known, may not be optimal or feasible at execution time.

It is thus necessary to explicitly model and consider possible plan failures and to devise a mechanism that is able to properly react to these failures. Moreover, on-line replanning (i.e., planning after plan failures) may not be feasible when the model itself is inaccurate, since the same cause of the plan failure (typically a non-modeled feature of the environment) will likely occur also in next executions.

To this end, we have defined a plan execution and monitoring framework composed by three modules: a planner (as described in the previous section), an executor, and a model updater. The three modules cooperate during the execution of a complex task for a robot and provide for a feedback mechanism from execution to planning. More specifically, the following interactions are devised: 1) the planner notifies on-line to the executor the best plan (policy) to be executed according to the current model of the world; 2) the executor executes this plan (policy) and determines success or failures of the actions; 3) each failure is reported to the model updater that will follow some rules (either automatic domain dependent or manual domain dependent) to modify the current model, so that the planner can generate a new plan that is more suitable for the current situation as detected by the executor.

The execution module is based on the Petri Net Plan (PNP) formalism[1] [8]. PNP is a formalism to represent high-level plans for robot and multi-robot systems. Being based on Petri Nets, it is very expressive and can thus represent durative ordinary and sensing actions, and many constructs such as sequence, loop, interrupt, fork/join, and several multi-robot synchronization operators. PNPs are used to model the behavior (i.e., the policy) that is generated by the planner module and to execute it using the PNP-ROS implementation that allows ROS actions[2] to be executed under the control of a PNP.

The two main components of this process will be described in the rest of this section: 1) Policy to PNP transformation; 2) Model updater.

Policy to PNP transformation. The policy generated by the MDP planner is automatically transformed in a PNP. For this process, the MDP planner

[1] pnp.dis.uniroma1.it
[2] wiki.ros.org/actionlib

produces the following information: the initial state, one or more goal states, state-action pairs implementing the policy and the conditions to be checked when non-deterministic actions are executed. States, actions and conditions are represented just as unique labels. With this input, the algorithm for generating the corresponding PNP is based on a recursive procedure for building the graph corresponding to the policy, starting from the initial state to the goal states, applying the state-action pairs for each state and adding a sensing operator for every non-deterministic effect of an action.

The labels in the policy and in the PNP referring to actions correspond to implemented actions, while labels referring to conditions correspond to sensor processing procedures that evaluate their truth based on the current information available to the system.

The PNP generated with this process does not contain a representation of action failures. Action failures are considered by defining a set of execution conditions for each action and by automatically adding action interrupts when these conditions are not valid. In this way the new PNP will be able to actually check execution conditions of actions and to interrupt the plan whenever these conditions are not valid. For example, an execution condition of a communication action is that a person is in front of the robot. While executing the action, the condition of a person being in front of the robot is checked and, if it becomes false, the action is interrupted.

When an interrupt is activated, the flow of execution of the plan can follow one of the two following lines: 1) *internal recovery procedure*[3], when the current plan itself contains a recovery behavior (i.e., a sub-plan or portion of the plan) for dealing with this failure; 2) *plan failure*, when the current plan is not able to deal with this kind of failure.

In the latter case, the executor sends to the Model updater module the following information: 1) action failed, 2) condition that was checked to determine action failure, 3) status of the plan (that can contain additional conditions useful for diagnosis of the failure). Given this input, the Model updater module (described in the next paragraph) modifies the MDP model of the domain and activates a new planning procedure to generate a new plan (policy) that aims at avoiding at least the failure cause just occurred.

Model update. The problem of updating a planning model, given the feedback of the execution of the plan, is very relevant for actual application of planning techniques to real problems, but, to the best of our knowledge, a general solution suitable for our needs does not exists.

At this moment, we have implemented a simple method that builds and maintains a tabu list of actions to be selected in the MDP planning process. More specifically, whenever an action fails, the action is inserted in the tabu list and thus it will not be selected in the next generation of the policy. This mechanism is also tied to a time decay mechanism, so that the presence of an

[3] At this moment, the *internal recovery procedures* are manually written, while some automatic technique could be devised.

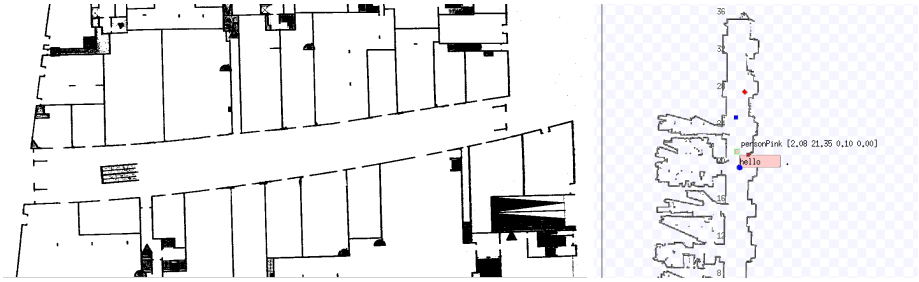

Fig. 3. 2D map of the *Rive de l'orne* shopping center and Stage simulator snapshot of the DIAG example.

action in the tabu list decreases over time, making the action available some time after the action failed, in order to try it again in the future.

For example, if the action of moving to a particular shop fails because there are too many people in that area, the robot will avoid to generate a new plan that will include going to that shop for a while, avoiding thus the main cause of the current failure.

Although not optimal, this strategy allows the robot to generate new plans that will possibly avoid the causes of failure of the previous plans.

4 Implementation and tests

Before experimenting the robots in the actual environment, it is necessary to develop and test the solutions in a simulator and in a more controlled environment. To this end, we report here the development of a simulated environment for the project and some preliminary tests made with the robot in the DIAG Department.

`COACHES` architecture is implemented within the ROS framework[4]. ROS includes several ready-to-use modules for basic functionalities of the robots, hardware drivers, simulation, and debugging. Moreover, the ROS environment guarantees an easy porting from simulation to real robots and in particular, our software architecture is implemented in such a way to remain unchanged when passing from simulation to robots.

4.1 2D Simulator environment

The simulation environment in `COACHES` is 2D and is based on Stage, in particular on its ROS version[5]. The choice of a 2D simulator (instead of a 3D one) is motivated by: 1) the need of modeling and testing high-level behaviors of the robots that do not involve 3D perception, 2) the possibility of using the simulator for multiple robots and other moving elements representing people in the

[4] www.ros.org
[5] wiki.ros.org/stage

environment, 3) the possibility of using the simulator on standard laptops, thus not requiring advanced graphical cards for running 3D simulations.

We have extended the original Stage simulator by adding a characterization of people in the environment and simple forms of HRI: i) the words spoken by the robot appears in the simulation window; ii) a GUI can be used by an operator to simulate human-robot inputs.

In the Stage simulator maps of the *Rive de l'orne* shopping center (Fig. 3 left) and of the DIAG Department (Fig. 3 right) have been realized. The Stage environment models one or more robots that have the same 2D sensor and actuator configurations as the real ones and some additional mobile obstacles that represent people moving in the environment. Several behaviors can be tested in this simulated environment such as: 2D perception of human behaviors, human-robot social navigation (e.g., following a person or guiding a person), safe navigation in the environment.

Several tests have been performed on the simulator, showing that it is a suitable tool for developing high-level robot behaviors.

4.2 Preliminary tests at DIAG

In order to test the developed modules on a robot, we have defined a task (similar to the `COACHES` use cases) that can be run in an office environment. We consider a robot assisting users in an office. The robot welcomes people at the entrance and tell them about the latest news. It may also offer assistance for the printer: bringing the printed document to some other person, or informing technicians about printer troubles.

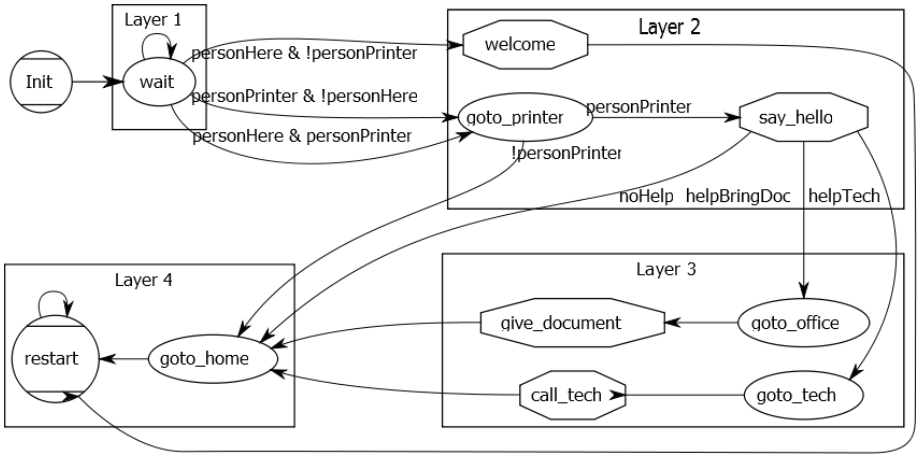

Fig. 4. PRU+ for printer-assistance and welcome

The mission is described in the PRU+ depicted in Figure 4. It has 4 layers: 1) waiting for people, 2) welcoming people and offering assistance, 3) bringing documents and fetching for technicians, and 4) returning to home position. The expected behavior is the following: from action 'wait' in Layer 1 four outcomes are possible: nobody is there, somebody has been detected near the entrance, near the printer, or both. When the 'wait' action completes, the robot might decide to wait again. If somebody is close to it, the robot can welcome and announce news. If somebody is at the printer, the robot can go there. Tasks 'welcome', 'say_hello', 'call_tech' and 'give_document' are also granting the robot a reward. They are represented by octagons in Figure 4.

From this PRU+, a 16-states MDP is built. Once solved, it produces the following policy:

- (Init): wait
- (1,wait,both): goto_printer
- (1,wait,entry): welcome
- (1,wait,print): goto_printer
- (2,goto_printer,err): goto_home
- (2,goto_printer,ok): say_hello
- (2,say_hello,bring): goto_office
- (2,say_hello,done): goto_home
- (2,say_hello,help): goto_tech
- (2,welcome,done): restart
- (3,goto_office,done): give_document
- (3,goto_tech,done): call_tech
- (3,give_document,done): goto_home
- (3,call_tech,done): goto_home
- (4,goto_home,done): restart
- (4,restart,done): restart

The policy is denoted by state-action pairs, where states are represented as $[l, m, v]$ (i.e., level, module and state variables, as described in the previous section) and actions correspond to the tasks defined in the PRU+. This policy is then translated into a PNP and executed by the robot.

Figure 5 shows some snapshots of plan execution, in the situation where the robot is asked to bring a document to a person. The interaction with the two persons involved and a few intermediate snapshots are reported. Notice that, although in a simplified setting[6], with these tests we have verified suitability and effectiveness of most of the components of our software architecture and their interconnection.

5 Conclusions

In this paper, we have described the main concepts of the COACHES project and in particular its Artificial Intelligence and Robotics components and their integration. More specifically, we have described a framework for integrating knowledge representation and reasoning, MDP planning and PNP execution, allowing a feedback from execution to reasoning in order to update and improve the current model of the world. Implementation and preliminary tests of such an integration have been performed to assess the suitability of the proposed architecture.

[6] At this moment HRI and perceptions modules are not fully implemented and thus we replaced them with the remote control of an operator.

Fig. 5. Example of plan execution.

Many interesting results are expected from the COACHES project, since the environment and the challenges considered here are very ambitious. Among the several performance evaluation procedures, we aim at including extensive user studies that will be used to validate the effective development of intelligent social robots performing complex tasks in public populated areas.

We believe that deploying robots in public spaces populated by non-expert users is a fundamental process for the actual design, development and validation of integrated research in Artificial Intelligence and Robotics. Consequently, we envision many significant contributions to this research area from the COACHES project.

Acknowledgements

COACHES is funded within the CHIST-ERA 4^{th} Call for Research projects, 2013, Adaptive Machines in Complex Environments (AMCE) Section.

References

1. E. Bastianelli, D. D. Bloisi, R. Capobianco, F. Cossu, G. Gemignani, L. Iocchi, and D. Nardi. On-line semantic mapping. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–6, Nov 2013.
2. W. Burgard, A. B. Cremers, D. Fox, D. Hhnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proc. of AAAI*, 1998.
3. R. Capobianco, J. Serafin, J. Dichtl, G. Grisetti, L. Iocchi, and D. Nardi. A proposal for semantic map representation and evaluation. In *Proc. of the European Conference on Mobile Robots (ECMR)*, 2015.
4. S. Cardon, A.-I. Mouaddib, S. Zilberstein, and R. Washington. Adaptive control of acyclic progressive processing task structures. In *Proc. of IJCAI*, pages 701–706, 2001.
5. S. Coradeschi et al. GiraffPlus: A system for monitoring activities and physiological parameters and promoting social interaction for elderly. In *Human-Computer Systems Interaction: Backgrounds and Applications 3 Advances in Intelligent Systems and Computing*, volume 300, pages 261–271, 2014.
6. I. Ferreira and J. Sequeira. Designing a Social [Robot] for Children and Teens: Some Guidelines to the Design of Social Robots. *International Journal of Signs and Semiotic Systems*, 3(2), 2014. Special Issue on The Semiosis of Cognition.
7. I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 2014.
8. V. A. Ziparo, L. Iocchi, P. U. Lima, D. Nardi, and P. F. Palamara. Petri net plans - A framework for collaboration and coordination in multi-robot systems. *Autonomous Agents and Multi-Agent Systems*, 23(3):344–383, 2011.