A Cooperative Agent Based Architecture for Environmental Exploration and Knowledge Sharing by Vision

Andrea Ruisi[1], Massimo Cossentino[2], Ignazio Infantino[2], Antonio Chella[1] and Roberto Pirrone[1]

[1]*DINFO, University of Palermo, Palermo, Italy, [ruisi, chella, pirrone]@unipa.it*
[2]*ICAR - CNR, sez. di Palermo, Palermo, Italy,[cossentino, infantino]@cere.pa.cnr.it*

Abstract

A comprehensive approach to the design and implementation of multi-robots cooperative systems is described. It focuses on a design process that uses the Unified Modeling Language and on a detailed ontology description with the goal of sharing the knowledge on environments that robots can acquire through the use of their vision sub-system. We base the implementation of our robotics vision system on agents inserted in a generic multi-level architectures. The first objective of this work is to provide a framework to perform a rigorous agent-based design process for scenarios where many robots are involved in different operations. Then we introduce a system for describing, upgrading and sharing knowledge about operating environments of a cooperative robot fleet. As a consequence we design a multi-level, agent-based vision architecture that takes advantage of the distribution over several different robots. Details of the methodology, of the ontological approach, of system implementation using FIPA-OS environment, along with real experiments are reported.

1. Introduction

In recent years, mobile robots have been involved in more and more complex tasks often requiring the collaboration among several individuals that in general differ in their skills, and in the way they perceive the external environment. In such a context, the research activity in the field of robotics has been mainly focused on the development of complex algorithms to accomplish the specific robotic tasks like path-planning, vision, localization, and so on. From the architectural point of view, two different philosophies have been carried on: the reactive and the behaviour-based paradigms. Our approach starts from these experiences and elaborates an architecture based on the cooperation of different levels of abstractions addressing three main robot's features: perception, cognitive components and actuators. The presence of a fleet of robots, each one deploying several agents, realizes a huge system whose management raises at least two kinds of problems: the complexity of the software needs a specific design methodology [4] while the ontology design and sharing needs a particular attention throughout all the phases of the process.

We consider the sharing of the ontology as one of the key issues in cooperative robotics because different robots can coordinate their work towards a common goal only if they can share the representation of the concepts and actions of their operating environment.

Starting from the previous considerations, we apply a novel methodology to the design of multi-agent robotic systems using the Unified Modeling Language. From a robotic point of view we refer to the behavior-based approach. Particularly, the proposed methodology uses behavior-based philosophy as a part of a wider process which begins with the requirements analysis for the whole system, identifies agents, and then defines behaviors [2]. The agents defined in such a way are deployed on the required hardware platforms, thus allowing both single robot and multi-robot scenarios.

For the ontology design, the architecture we adopt is based on the fundamental assumption that robots can obtain environmental experience from three different and conceptually divided channels: (i) the metric channel, giving quantitative information about the environment (lasers, sonars, odometers); (ii) the visual channel, giving snapshots of the environment (cameras); (iii) the semantic channel, giving support for the association of a *semantic* valence (a category) to spatial entities. The last channel is introduced as a way to face the well known *anchoring* problem [20]; in order to deal with the symbolic representation of the environment, a suitable ontology model has been devised which relates the couples *(symbol, entity)* that are present/discovered in the environment itself. We start from a meta-ontology (Ontology Identification Phase) from which we identify the ontological description suitable for the specific problem (Ontology Description Phase). In describing the structure of the knowledge we use UML in one of the phases of the design process (Domain Ontology Description). The proposed architecture can be extended towards the definition of the objectives of the different agents using methodologies like desirability functions [16,17] or generally behavior-based architectures [18].

The identified structures are then used to model the communications in the Communication Ontology Description phase. The paper is arranged as follows: Section 2 deals with the overall description of the agent based architecture; section 3 explains the design methodology; section 4 deals with the description of the domain ontology; section 5 reports the multi-level vision architecture with experimental results, and finally in section 6 some conclusions are drawn.

2. Description of the Robotic Architecture

From the cognitive point of view, in our approach we refer to the architecture of fig. 1a. In this structure it is possible to devise three main components: the perception, which is responsible to map the stream of raw data in a symbolic form, that in turn is provided to the cognitive component where the symbolic data computation and, in general, deliberative behaviors of the system are located. The cognitive part can also support perception with some hints aimed to refine the perceptive process, and focuses the attention on those external stimuli that are judged to be more useful for the current task completion. The third component is the actuation one, which communicates with the other two, in order to drive the robot hardware during perception tasks, and in attention focusing. The perception-action link allows also reactive behaviors. Some of the authors already presented this architectural structure [3,7,10]. Its main goal is to go beyond the classical behavior-based model, and to provide the robot with true "symbol grounding" capabilities due to the intermediate representation of sensory data, that is used to instantiate pieces of knowledge at the symbolic component. Through this mechanism the robot is able to act more effectively in a deliberative fashion. The aim of this work is to provide a framework for our architecture allowing us to define a rigorous design methodology relying on the agent-based software paradigm. In particular, the scheme reported in figure 1a can be regarded as a categorization of the possible agents typologies both if we look at the single robot architecture and if we consider a multi-robot scenario. In the second case we address the interaction between the external actors, and the whole team in order to perform cooperative tasks. In other words figure 1 is the highest level of abstraction in the system design, without taking into consideration the implementation details. Our approach suggests a possible abstraction from the single robot architecture to a multi robot team: the robot that is itself a multi-agent system, can be viewed as a single agent in the multi robot context in which it cooperates with the others in order to reach the goals of the entire system. Each robot can be thought as containing several agents; some of them interact with the external environment, some others process the knowledge to plan a strategy of reaching the goal, and at the end, other agents issue commands to the robot's hardware. At the same time it is also possible to zoom in the single robot representation and to see it as composed of several agents logically classifiable in the same three types (Perception, Cognitive and Actuator). Furthermore we can zoom in each single agent and find a perception capability (necessary to be aware of the external environment), a cognitive part (where the knowledge is processed) and some actuator features (to realize the decisions taken in order to reach the goal). It is simple to identify these elements in a vision agent. It accesses to an image using the driver of an hardware or through some kind of interaction with another agent (for example a message exchange), it processes the image accordingly to its objective and at the end it communicates the result to one or more agents interested in further steps. In our experiments we refer to the FIPA (Foundation for Intelligent Physical Agents) architecture [1]. In this approach, each agent is composed by a colony of tasks as described in fig. 2 and can play different roles that can be put into relation with one of the three areas reported in the general architecture of fig. 1. We suppose that there is a one-to-many relation between each one of these three areas and the agents of the system as depicted in fig. 1b.

3. The Design Methodology

In conceiving our design methodology we followed one specific guideline: the use of standards whenever possible. This justifies the use of UML as modeling language, the use of the FIPA architecture for the implementation of our agents and the use of XML in order to represent the knowledge exchanged by the agents in their messages.

Our methodology, called PASSI (Process for Agent Societies Specification and Implementation) [5] is a step-by-step requirement-to-code method for developing multi-agent software that integrates design models and philosophies from both object-oriented software engineering and MAS using UML notation. It has evolved from a long period of theory construction and experiments in the development of embedded robotics applications (see [6],[7],[9]). It is composed of five models (System Requirements, Agent Society, Agent Implementation, Code Model and Deployment Model) which include several distinct phases (Fig. 2).

The code production phase is also strongly supported by the automatic generation of a great amount of code thanks to a library of reusable patterns of code and pieces of design. The models and phases of PASSI are:

1. System Requirements Model. A model of the system requirements in terms of agency and purpose.

It is composed of four phases: (a) Domain Description

(D.D.): A functional description of the system using conventional use-case diagrams. (b) Agent Identification (A.Id.): The phase of attribution of responsibility to agents, represented as stereotyped UML packages. (c) Role Identification (R.Id.): A series of sequence diagrams exploring the responsibilities of each agent through role-specific scenarios. (d) Task Specification (T.Sp.): Specification of the capabilities of each agent with activity diagrams.

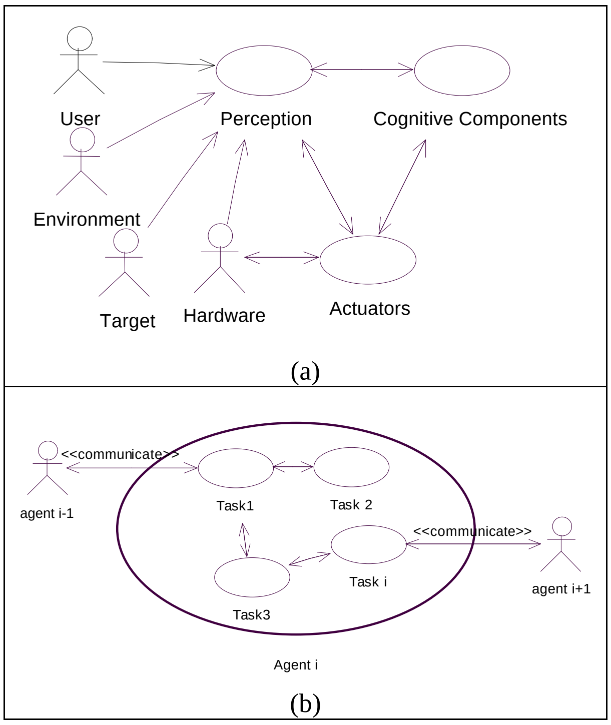
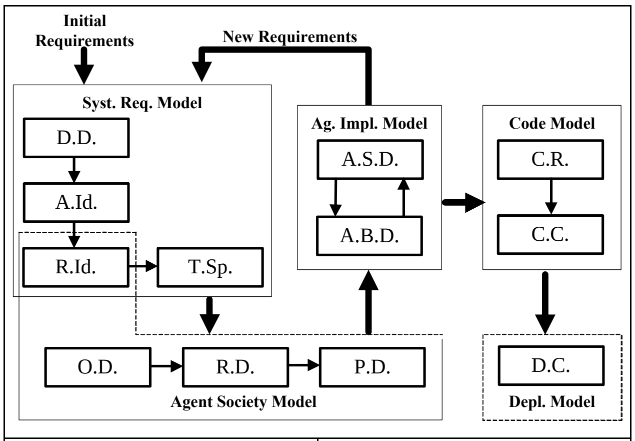


Figure 1: (a) The architecture of a single robot from the cognitive point of view. (b) The internal structure of the agent.

2. Agent Society Model. A model of the social interactions and dependencies among the agents involved in the solution. Developing this model involves three steps in addition to part of the previous model: (a) Role Identification (R.Id.): See the System Requirements Model. (b) Ontology Description (O.D.): Use of class diagrams and OCL constraints to describe the knowledge ascribed to individual agents and the pragmatics of their interactions. (c) Role Description (R.D.). Class diagrams are used to show the roles played by agents, the tasks involved, communication capabilities and inter-agent dependencies. (d) Protocol Description (P.D.). Use of sequence diagrams to specify the grammar of each pragmatic communication protocol in terms of speech-act performatives.

3. Agent Implementation Model. A classical model of the solution architecture in terms of classes and methods; the most important difference with common Object-oriented approach is that we have two different levels of abstraction, the social (multi-agent) level and the single-agent level. This model is composed of the following steps: (a) Agent Structure Definition (A.S.D.): Conventional class diagrams describe the structure of solution agent classes. (b) Agent Behavior Description (A.B.D.): Activity diagrams or state-charts describe the behavior of individual agents.



Key:	
D.D. – Domain Description	O.D. – Ontology Description
A.ID. – Agents Identification	R.D. – Roles Description
R.Id.– Roles Identification	P.D. – Protocols Description
T.Sp. – Task Specification	C.R. – Code Reuse
A.S.D.–Ag. Structure Defin.	C.C. – Code Completion
A.B.D.–Ag. Behavior Descr.	D.C. – Deployment Config.

Figure 2: The models and phases of the PASSI methodology.

4. Code Model. A model of the solution at the code level requiring the following steps to produce: (a) Generation of code from the model using one of the functionalities of the PASSI add-in. It is possible to generate not only the skeletons but also largely reusable parts of the methods implementation based on a library of code and associated design descriptions. (b) Manual completion of the source code.

5. Deployment Model. A model of the distribution of the parts of the system across hardware processing units, and their migration between processing units. It involves one step: Deployment Configuration (D.C.): deployment diagrams describe the allocation of agents to the available processing units and any constraints on migration and mobility.Testing: the testing activity has been divided into two different steps: the single-agent test is devoted to verifying the behavior of each agent regarding the original requirements for the system solved by the specific agent. During the society test an integration verification is carried on together with the validation of the overall results of this iteration.

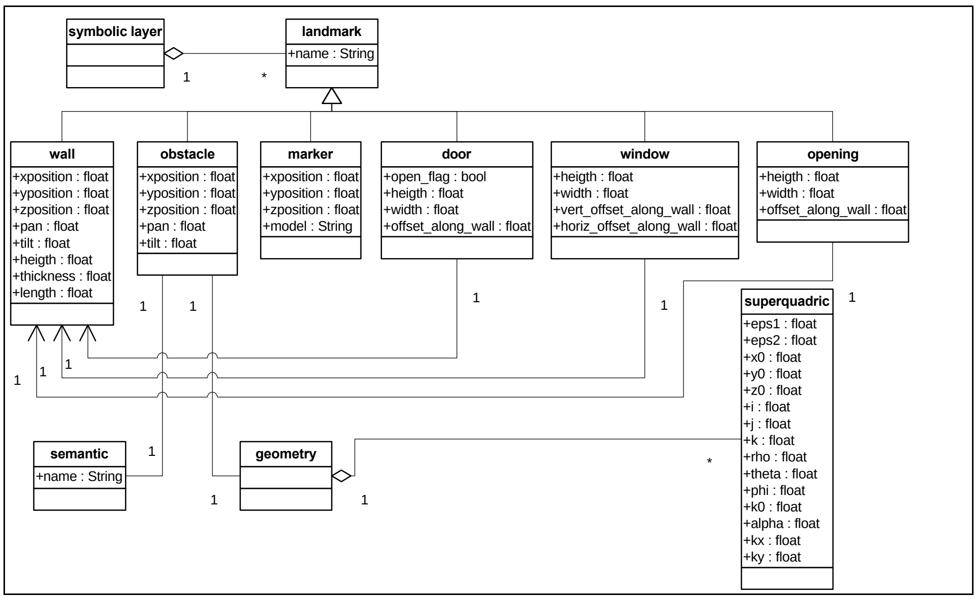


__Figure 3:__ The ODM Static Symbolic Layer. This ODM sub-model actually represents (part of) the DOD model of our domain: the knowledge structure we identified is then used to model the content of the communications in the COD phase and finally to define the agent communication ontology.

The Agent Test is performed on the single agent before of the deployment phase while the society test is carried on the complete system after its deployment.
A more detailed description of the methodology is beyond the scope of this paper. In the following section we will deal with its steps that describe the ontology of the domain and the robots' knowledge exchange.

4. Ontology Representation and Sharing

Multi-robot collaboration, in our approach, has a direct consequence in the knowledge design and sharing. We introduced the Ontology Description (OD) phase in the design methodology in order to model both the identification/representation of the domain ontology, and the communications that the robots use to exchange it. OD phase is composed by two sub-phases: the Domain Ontology Description (DOD) and the Communication Ontology Description (COD). In the DOD we represent the ontology as an UML class diagram that basically describes the overall domain ontology. In the COD (another UML class diagram), we describe the knowledge of each agent (thus addressing the implementation of its data structures) and its communications, from the ontological point of view, referring to the elements of the previous drawn DOD diagrams. The description of the domain ontology is obviously one of the most difficult phases of the entire process. We have developed a specific sub-process for it, divided in two subsequent phases: Ontology Identification Phase (OIP) and Ontology Description Phase (ODP). The OIP is a multi-perspective study driven by a functional analysis of the domain, and allows the identification of the different types of knowledge and of the related ontologies needed for the implementation of particular classes of tasks and functionalities. Result of the OIP is a general meta-ontological model (Ontology Identification Model, OIM) that encompasses a set of conceptually divided ontological abstract sub-models. Each of them is "generated" by a particular view that is assumed in front of the operating environment in order to capture those aspects that are meaningful for that particular class of tasks [14]. OIM uses UML as a representative formalism, grouping ontologies into more general categories according to the dichotomy between quantity and quality, and showing how consistency between these groups is determined by causal models.
In the next phase (ODP), the results of OIP are used as a framework to drive the actual definition of the Domain Ontology. Through re-interpretation and finer-grained specification of OIM abstract ontological sub-models, we now produce a global and implementable representational model (Ontology Description Model, ODM). Since conceptually different ontologies relate, in implementation optics, to different inner representational formalisms, in ODM we use knowledge layerisation to allow them to co-exist into an overall domain model. Furthermore, in order to allow symbolic manipulation of knowledge, and to avoid the problems that a fully adaptive representation model could arise [15], we augment quantitative information with

qualitative-semantic one (obtained through the semantic acquisition channel) and vice-versa, and we decouple static and dynamic knowledge. ODM layers are defined at a detail level that is able to capture all the information that is necessary for their implementation, and are expressed using UML as a representation formalism. As an example, the Static Symbolic ODM Layer is shown in figure 3. Formally, this layer is composed by landmarks representing structural static elements of typical indoor office environments named upon their semantic category. That is, walls and objects (furnishings) that are statically present in a specific position. Landmarks we have identified for office environments are: (i) walls; (ii) doors; (iii) windows; (iv) openings; (v) obstacles; (vi) markers. Each landmark is tagged with an unique identifier (ID) in order to distinguish it from others in the same category. Attributes needed to formally identify a wall are: its position and orientation (referring to a global coordinate system), its thickness and its height. Inside each wall, one or more windows, doors or openings can be present. Each of them has an ID and some parameters needed to identify its position relatively to the containing wall, its width and its height. An obstacle is any generic motion obstacle. Each obstacle has a geometry, modeled using a set of deformable superquadrics, and is related to its semantic through a name defining its category (a table, a chair). Finally, a marker is a physical entity (i.e. a label) that is easily distinguishable by the robot and can act as a trigger for some kind of behavior (i.e. in museum guide robot, reading a sign would cause a multimedia presentation to start). It is formally identified by its position and model description (a link to the file containing its graphical representation). Among the other layers we identified there are: the topological, geometrical and grid-based ones. We developed an XML based realization of the ODM through its translation in XML DTDs/Schemas, thus treating a new XML-based mark-up language (Environmental Knowledge Markup Language, EKML) that we use for knowledge representation. At present, we are testing the introduction of XML DOM-Parsers into Fipa agents to share and update knowledge stored in EKML data structures.

5. The Multi-Level Vision Architecture

In the following we will describe the realized vision architecture referring to an experiment performed using a real robot equipment in an unstructured environment. The robot was provided with obstacle avoidance capabilities in order to reach a static target. The implemented behaviour is quite simple because our study was mainly focused on testing architecture implementation rather than developing high quality solutions to accomplish the robot's tasks. We were particularly interested to stress multi platform communication features of the FIPA environment, and to cope with its lack of real-time control capabilities. Our robot was a K-Team Koala equipped with IR sensors, and controlled by a PC through a radio link. Vision was provided by a calibrated camera looking at the action field, and reporting localization information to the rest of the system. In order to test distribution of agents across multiple platforms, the camera was connected to a separate PC running part of the vision system code. Obstacle avoidance was simply implemented by processing IR sensors readings in order to detect obstacle proximity. Then the robot follows obstacle's contour until it has free path to reach the goal. Path planning consists of a series of "turn" and "go straight" movements that are computed starting from vision data. In what follows, a typical experiment as long as the implementation of the multilevel vision system will be reported in detail.

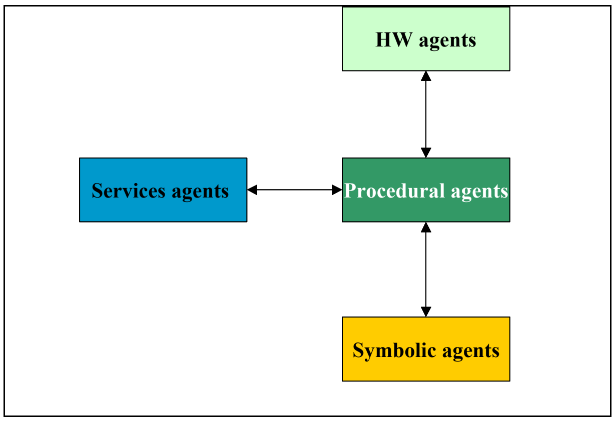


Figure 4: *The proposed multilevel architecture is based on various agents grouped in classes which have different level of knowledge: low level (HW agents), sub-symbolic level (Procedural agents and Services agents), high level (Symbolic agent).*

5.1 Distribution of agents across multiple platforms

The proposed multilevel architecture (see fig. 4) is based on various agents grouped in classes which have different level of knowledge: low sensorial level (HW agents), sub-symbolic level (Procedural agents and Services agents), high level (Symbolic agent). All the agents can be located on different platforms and the system provides them of the communications capabilities. In fig. 7 it is depicted an example of the agent activations during a generic planning task:

- The Planner Agent is part of the symbolic agent level and it plays the fundamental role of activation and coordination of the several agents involved. The most important knowledge of this agent is the map of the scene in which new data are added to the a-priori data (see fig. 7.d). The Planner Agent uses two Procedural Agents: Tracking Agent and 3D Reconstruction Agent.

- A Procedural Agent can receive collaboration from several Services Agents in order to process its knowledge. For example the 3D Reconstruction Agent (see fig 7.c) owns images acquired by visual sensors on which it requested to perform filtering, edge extraction, camera calibration and so on.
- The level of Services Agents is a extensible collection of simple low-processing agents (see fig. 7.b) useful to perform various calculations requested by one or more Procedural Agent.
- The source of visual data is the level of Hardware Agents: the Devices Manager Agent is the interface between video (or image) sources and Procedural Agents that include this type of data in their knowledge. Every video source has its specific camera agent to communicate to Device Manager Agent (see fig. 7.a).

5.2 The Sensorial Level and the Single Camera Agent

We use a fixed CCD camera, connected to a computer, viewing the scene (see fig. 5). The single camera agent can run on a different machine from the one that runs the rest of the system, communicating with it over the local net. In this way we have the possibility of performing the vision task in real time without adding high computational costs to the whole system.

5.3 Service Agents contributing to sub-symbolic knowledge

This section describes the process of localization of the Koala robot during its task, in order to give useful feedbacks to the planning agent [2,3,13]. The position of the robot in the image is calculated by simple low-level image processing operations performed by the corresponding Services Agents. The current frame is subtracted to the previous (gray level images), obtaining the pixels related to moving objects in the viewed scene. If more objects are moving, the Koala shape is selected using color and textural features. Naturally, some standard filtering operations are performed to reduce noise. Moreover, a corner detector is applied in the area of the image representing the Koala shape in order to obtain feature points to track.

5.4 The Procedural Agents and the sub-symbolic knowledge

The estimation of the position of the robot on the floor is based on this tracked points. The valuable capabilities of the 3d Reconstruction Agent and Tracking Agent in the whole system are:

- to individuate and segment the Koala robot also in contrasted and irregular backgrounds;
- to perform an estimation of the position of the robot by camera images;
- to interpret the sequence of movements of the robot giving information about the direction followed by it.

The implemented computer vision task can be decomposed in three main steps:

- localization of the robot on the image by low-level image processing of the single frame;
- estimation of the 2D location of the robot on the floor;
- reconstruction of the 3D position of the robot.

The position of the robot referred to a reference system is estimated using the homography between the image plane and the floor [11]. A generic 3D point X generates the point w on image:

$$l\,w = PX = K[R \quad t]X$$

if the 3D points are on a plane (i.e. Z=0), the transformation is simplified to a 3x3 matrix H:

$$l\,w = HX^P = [r^1 \quad r^2 \quad t]X^P$$

where H is the homography matrix, decomposable on the calibration 3x3 matrix K, and a 3x3 matrix that has the first two columns of the rotation matrix R and the translation vector t as third column. X and w are indicated using homogeneous coordinates. During a preliminary calibration process, the matrix H is estimated using detected points belonging to the floor; a grid placed in front of the camera is used to obtain the calibration matrix K and to fix the rotation and translation referred to the reference system. The tracked points on image are translated in 2D coordinates using estimated homography. The exact 3D position is recovered using the known real dimensions of the koala robot and the data coming from the calibration framework [8]. The estimated 2D coordinates of the robot and the direction of the detected movements are communicated to the system with messages. The path of Koala is recorded by the Planner Agent and it is the source of 3D data for a powerful dynamic visualization using a browser equipped with the plug-in for standard VRML language (see fig. 6).

5.5 The Planner Agents and the symbolic knowledge

Planning relies on several agents acting at different level of abstraction. The collaboration level agent uses information produced by procedural agents to derive the whole team strategy in achieving the goal. The result of this step is a different high level plan for each robot. Starting from this input, a single robot makes its routing plan using the Topological Layer of the ontology described in chapter IV.

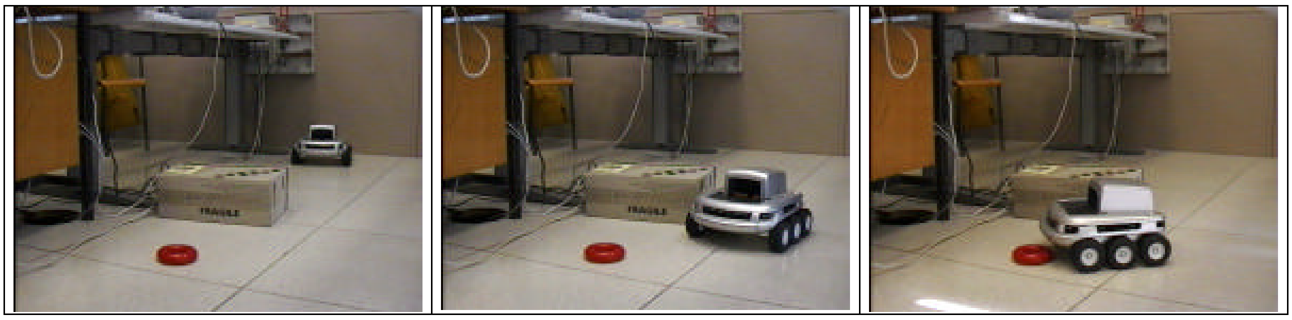


Figure 5: *some frames of the experimental sequence: the Koala robot avoids the obstacle and reaches the target.*

An A* algorithm is used to browse the graph connecting adjacent parts of the environment (rooms, corridors, etc.) through their openings. Low level navigation inside a single room and obstacle avoidance are performed using an approach based on potential fields [19].

6. Conclusion

A novel methodology for the design of multi-agent robot architectures including also vision agents is presented that extends the classical behaviour-based approach. It shall be showed that it can be profitably used both in the case of a single robot design, and in a multi-robot scenario.In order to strengthen the cooperation capabilities of a multi-robot system our methodology comprehends an extensive specification of ontology. Starting from a referring framework where qualitative and quantitative descriptions are related through their causal relationships, we deduct the proper ontology description model for the environment where the robots operate. We considered vision the main source for the environmental knowledge and therefore we produced a flexible, modular and distributable vision architecture, where each agent can take advantage of services provided by agents present in other computational nodes, producing a network of cooperating entities that reduces the need for duplication of most common services. The methodology presented has been implemented using a FIPA compliant platform, and the experimental results have been very encouraging. We are currently extending the methodology towards automatic code generation for a great part of the agents' implementation.

7. References

[1] FIPA Abstract Architecture Spec. (Refinements). FIPA specification documents (08-10-01). http://www.fipa.org

[2] Arkin R., Behavior Based robotics, The MIT Press, Cambridge, Massachusetts, London, England, 1998.

[3] Chella A., Gaglio S., Pirrone R., Conceptual representations of actions for autonomous robots, Robotics and Autonomous Systems, 34, (2001), 251-263.

[4] Jennings N.R., On agent-based software engineering, Artificial Intelligence 117 (2000), 277-296.

[5] Cossentino, M., Potts, C. A CASE tool supported methodology for the design of multi-agent systems in proc. of SERP'02 (Las Vegas, Nevada, Jul 2002)

[6] Chella, A., Cossentino, M., and Lo Faso, U. Designing agent-based systems with UML in Proc. of ISRA'2000 (Monterrey, Mexico, Nov. 2000).

[7] Chella, A., Cossentino, M., Infantino, I., and Pirrone, R. An agent based design process for cognitive architectures in robotics in proc. of WOA'01 (Modena, Italy, pt. 2001).

[8] I. Infantino, R. Cipolla, A. Chella, "Reconstruction of architectural scenes from uncalibrated photos and maps", IEICE - Transaction on Information and System, Vol.E84-D No.12 pp.1620-1625.

[9] Chella, A., Cossentino, M., Tomasino, G. An environment description language for multi-robot simulations in proc. of ISR 2001 (Seoul, Korea, 2001)

[10] Chella, A., Guarino, D., Infantino, I., Pirrone, R., A Vision System for Symbolic Interpretation of Dynamic Scenes Using ARSOM, Applied Artificial Intelligence, Vol. 15 No. 8, Issue Sep 2001,pp.723-734.

[11] Faugeras, O.: Three-Dimensional Computer Vision. MIT Press, Cambridge, MA, 1993.

[12] Horn B.P.K., Robot Vision, MIT Press, Cambridge, 1986.

[13] Russel S., Norvig P., Artificial Intelligence: A Modern Approach, Prentice Hall Int. Ed., 1995.

[14] Chella, A., Cossentino, M., Pirrone, R., Ruisi, A., Modeling ontologies for robotic environments, Proc of 14th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE 2002), July 15-19 2002, Ischia, Italy,.

[15] Fox, D., Burgard, W., Thrun, S., Probabilistic methods for mobile robot mapping, in Proc. Of the IJCAI-99 Workshop on Adaptive Spatial Representations of Dynamic Environments, 1999.

[16] A. Saffiotti, N.B. Zumel, and E.H. Ruspini. Multi-Robot Team Coordination using Desirabilities. Proc. of the 6th Intl. Conf. on Intelligent Autonomous Systems (IAS), pp. 107-114. Venice, Italy, 2000.

[17] A. Saffiotti and E.H. Ruspini. Global Team Coordination by Local Computation. Proc. of the European Control Conference (ECC). Porto, Portugal, 2001

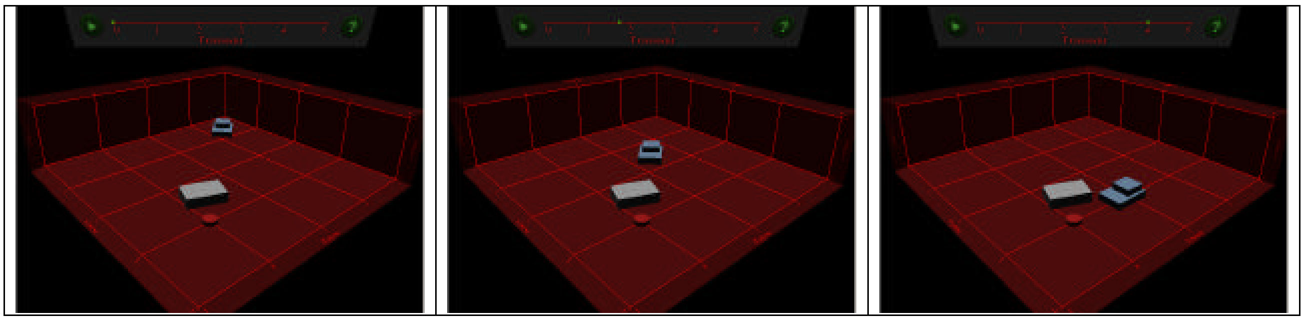


Figure 6: *some frames of the reconstructed scene using a standard VRML model.*

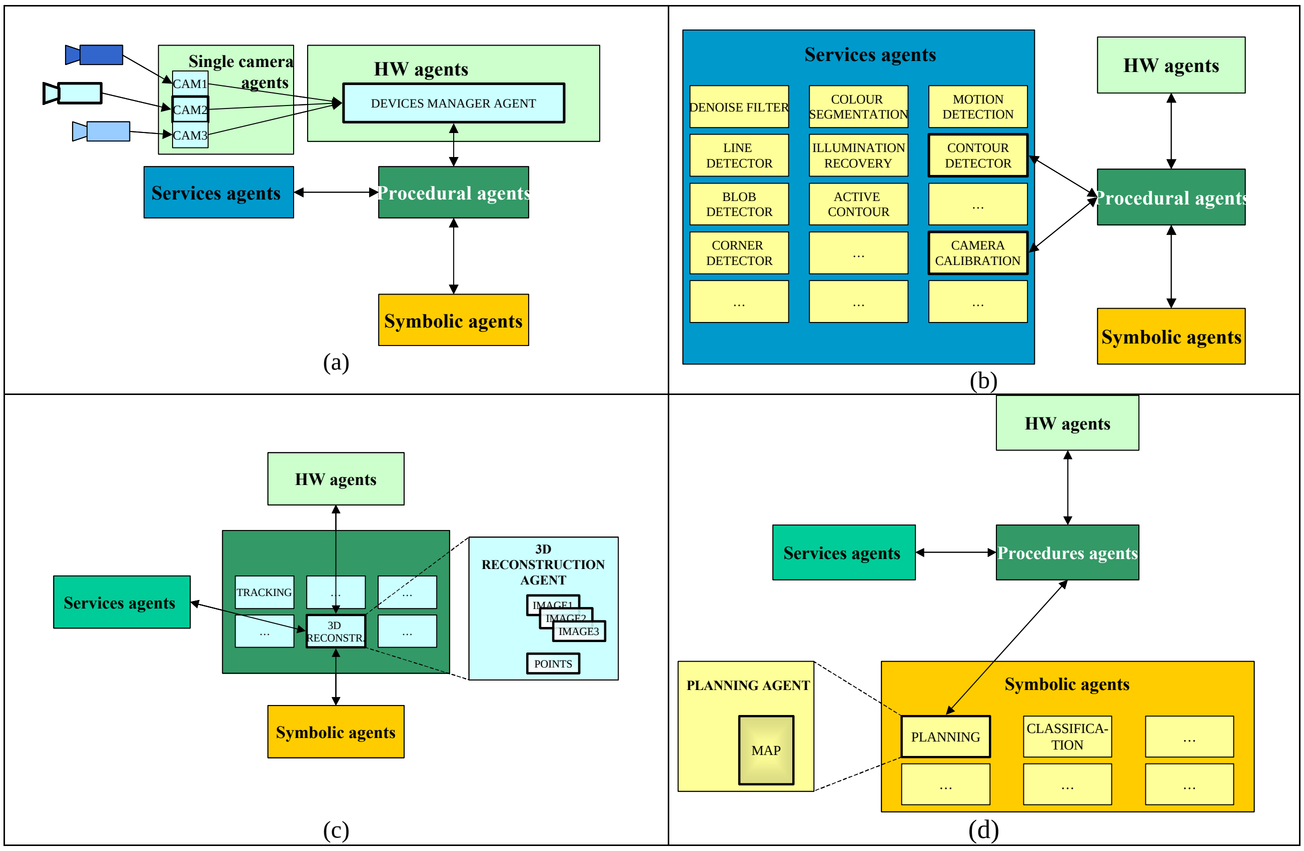


Figure 7: *the proposed architecture exploited: (a) The source of visual data is the level of Hardware Agents: the Devices Manager Agent is the interface between video sources and Procedural Agents; (b) The level of Services Agents is a extensible collection of simple low-processing agents useful to perform various calculations; (c) The 3D Reconstruction Agent owns images acquired by visual sensors on which it requested to perform filtering, edge extraction, camera calibration and so on using Process Agents; (d) The Planner Agent is part of the symbolic agent level and it plays the fundamental role of activation and coordination of the several agents involved.*

[18] Balch, T. and Arkin, R.C. Behavior-based formation control for multi-robot teams. IEEE Transactions on Robotics and Automation 14(6):926–939, 1998.

[19] Latombe, J.C., Robot Motion Planning, Kluwer Academic Publisher, Boston, 1996.

[20] Coradeschi, S., Saffiotti, A. Anchoring Symbols to Sensor Data: preliminary report. Proc. of the 17th AAAI Conf, 129-135. Austin, Texas, July 2000.