An Agent Based Multilevel Architecture for robotics vision systems

Ignazio INFANTINO, Massimo COSSENTINO, Antonio CHELLA

***Abstract* - An approach to the design and implementation of a robotics vision system based on agent inserted in a generic multi-level architectures for mobile robotics is presented, that is based on the Unified Modelling Language. The main goal of the work is to provide a framework to perform a rigorous agent-based design process for cognitive architectures both in the case of a single robot, and in a multi-robot scenario. Details of the methodology, system implementation using FIPA-OS environment, along with real experiments are reported.**

Index Terms: Vision agents, distributed architecture, mobile robotics, cognitive architectures, agent-based software engineering.

I. 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. We think that these approaches don't allow to manage very large problems like the case in which a single robot has to solve a very complex task, or when a fleet of robots cooperates to achieve a common goal. Nowadays, agent-based architectures are increasingly used to model more and more complex systems. This induces the designers to the introduction of software engineering principles in developing such systems. Starting from the previous considerations, our work aims to propose a novel methodology for the design of multi-agent robotic architectures using the Unified Modeling Language. The methodology has been applied to the cognitive architecture previously developed by some of the authors, that could be viewed as an extension of the behaviour-based approach. Particularly, the proposed methodology uses behaviour-based philosophy as a part of a wider process which begins with the requirements analysis for the whole system, identifies agents, and defines behaviours [2]. The agents defined in such a way are deployed on the required hardware platforms, thus allowing both single robot and multi-robot scenarios. 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 reports experimental results, while in section 5 some conclusions are drawn.

I. Infantino and M. Cossentino are with the CERE-CNR, c/o CUC, Viale delle Scienze, 90128, Palermo, Italy (e-mail: [infantino,cossentino]@ cere.pa.cnr.it).

A. Chella is with DINFO, University of Palermo, Viale delle Scienze, 90128, Palermo, Italy. (e-mail: chella@unipa.it).

II. Description of the Architecture

From the cognitive point of view, in our approach we refer to the architecture of fig. 1. In this structure it's 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 focus 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 in a deliberative fashion more effectively. 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 1 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. 2.

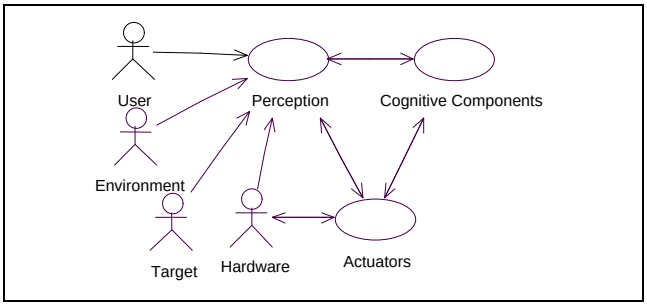

Fig. 1 The architecture of a single robot from the cognitive point of view.

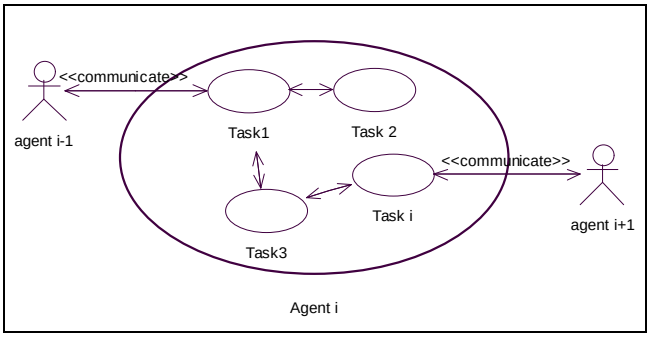

Fig. 2 The internal structure of the agent.

III. THE DESIGN METHODOLOGY

If. As will be discussed in the following sections, our approach to the vision subsystem (that is only one of the subsystems that can be identified in a robot) generates a relevant number of agents. It is flexible, scalable and versatile but the number of agents requires a strong commitment to the management of this complexity.

For this reason, we consider necessary the use of a design methodology coming from the agent-based software engineering. Our

Our methodology, called PASSI (Process for Agent Societies Specification and Implementation) 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 [3, 6, 7]). Its precursor, AODPU has been applied in the synthesis of embedded robotics software and is the basis of teaching materials in agent-based software engineering [4,6].

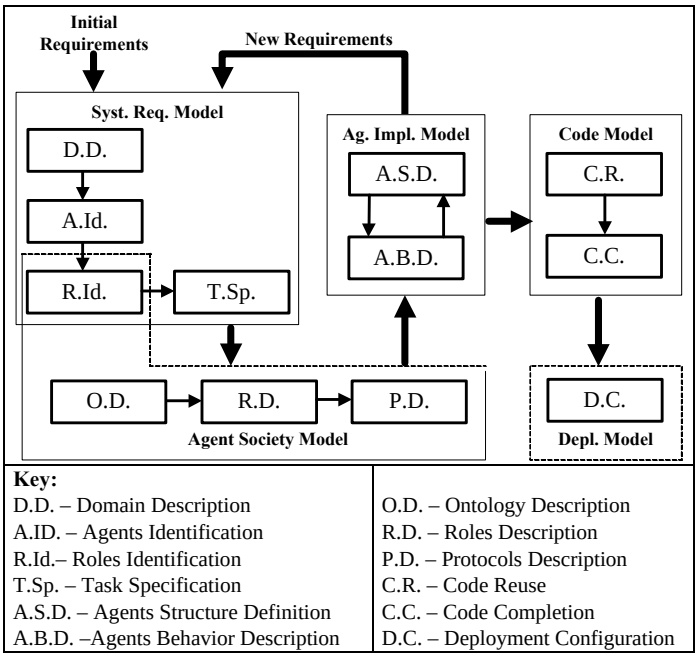

Fig. 3 The models and phases of the PASSI methodology.

It is composed of five models (System Requirements, Agent Society, Agent Implementation, Code Model and Deployment Model) which include several distinct phases (Fig. 3). 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.

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.

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. 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.

IV. Experimentation

Experimental phase has been performed using a real robot equipment in an unstructured environment. 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-OS 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.

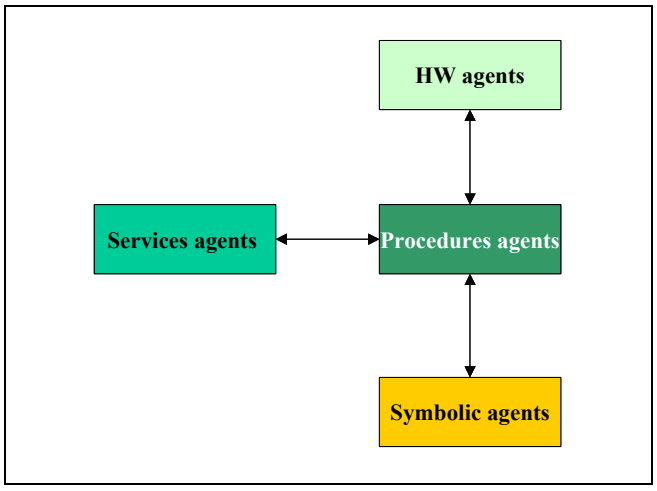

Fig. 4. The proposed multilevel architecture is based on various agents grouped in classes which have different level of knowledge: low level (HW agents), sub-symbolic level (Procedure agents and Services agents), high level (Symbolic agent).

A. 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 (Procedure 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:

1. 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 Procedures Agents: Tracking Agent and 3D Reconstruction Agent.

2. A Procedures 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.

3. 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 Procedure Agent.

4. The source of visual data is the level of Hardware Agents: the Devices Manager Agent is the interface between video (or image) sources and Procedures 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).

B. 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 to 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.

C. The Sub-symbolic Knowledge by the Service Agents.

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 on 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 object, 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.

D. The Procedure Agent and its symbolic knowledge

The estimation of the position of the robot on the floor it 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 of 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].

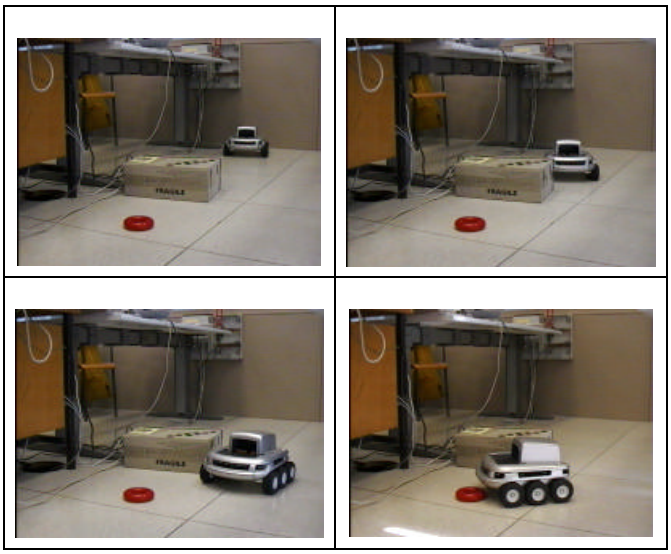

Fig. 5. Some frames of the experimental sequence: Koala robot avoids the obstacle (the box) and reaches the target (red toy).

A generic 3D point X generates the point w on image:

$$lw = PX = K\begin{bmatrix} R & t \end{bmatrix}X \quad (1)$$

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

$$lw = HX^{P} = \begin{bmatrix} r^{1} & r^{2} & t \end{bmatrix}X^{P} \quad (2)$$

where H is the homography matrix, decomposable on the calibration 3x3 matrix K, and 3x3 matrix that has the first two columns of the rotation matrix R and the translation vector t. 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 fixes 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 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).

V. 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. The methodology has been implemented using a FIPA compliant platform, and the experimental results are been very encouraging. We are currently extending the methodology towards automatic code generation for a great part of the agents' implementation.

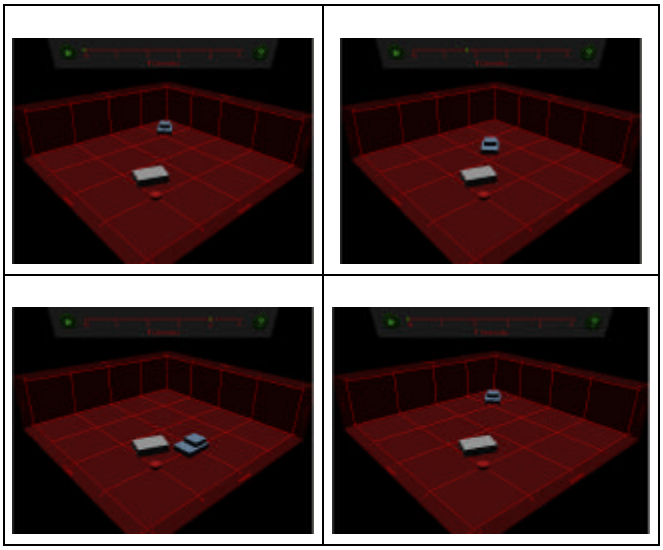

Fig. 6. Some frames of the reconstructed scene using a standard VRML model..

REFERENCES

[1] FIPA Abstract Architecture Spec. (Refinements). FIPA specification documents (08-10-01). http://www.fipa.org /specs/fipa00094/PC00094.html.

[2] Arkin R., Behavior Based robotics, The MIT Press, Cambridge, Massachussets, 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] Chella, A., Cossentino, M., and Lo Faso, U. Applying UML use case diagrams to agents representation. Proc. of AI*IA 2000 Conference. (Milan, Italy, Sept. 2000).

[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, Sept. 2001).

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

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

[10] Chella, A., Guarino, D., Infantino, I., Pirrone, R., A Vision System for Symbolic Interpretation of Dynamic Scenes Using ARSOM, Applied Artificial Intelligence Special Issue on "Machine Learning in Computer Vision", Volume 15 Number 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, MA, 1986.

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

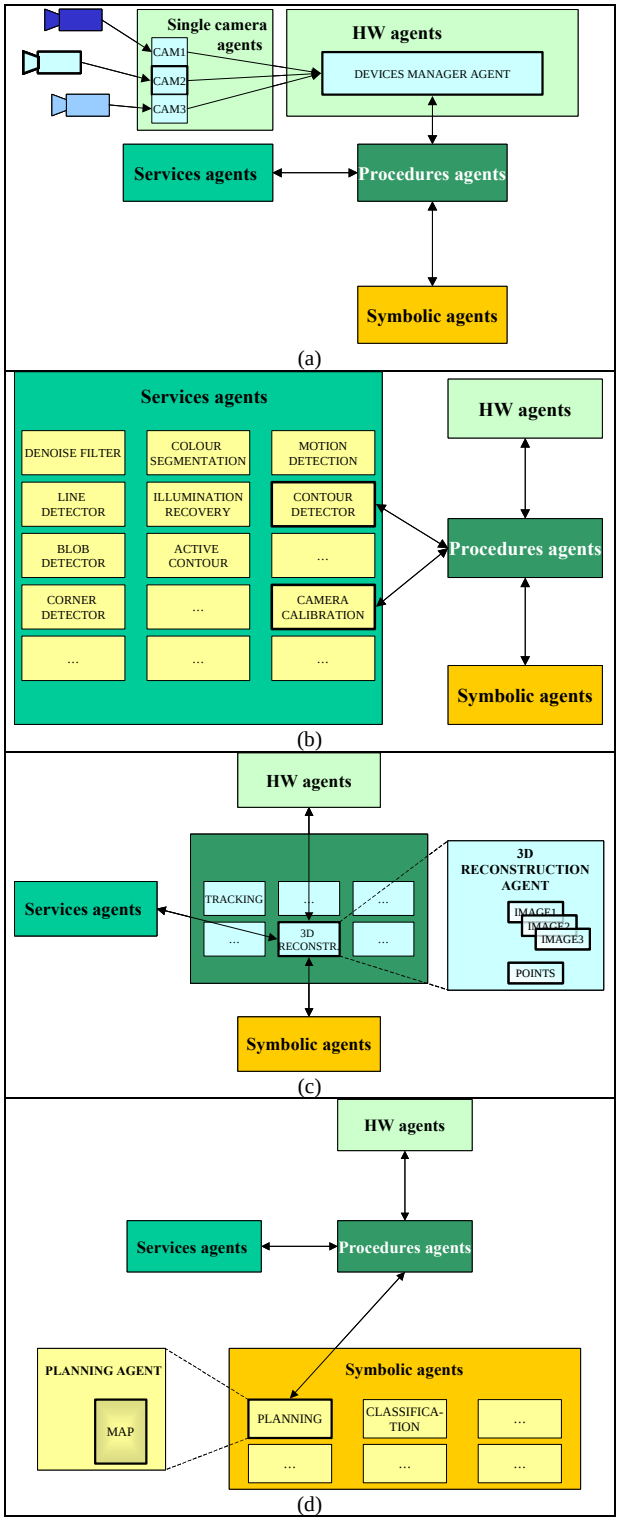

Fig. 7 The proposed architecture exploited: (a) The source of visual data is the level of Hardware Agents: the Devices Manager Agent is the interface between video sources and Procedures Agents; (b) The level of Services Agents is a extensible collection of simple low-processing agents useful to perform 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.