Deep State Representation: an Unified Internal Representation for the Robotics Cognitive Architecture CORTEX

Luis V. Calderita

Cáceres, 2016
University of Extremadura
Doctoral Dissertation

Dedicado a Lidia, a mis padres y a Didi.
Gracias por vuestro apoyo incondicional.

No creas que estoy huyendo
si me ves retroceder ¡espera!
que estoy cogiendo carrera.
Desafiar la perspectiva del fracaso
a la que estamos condenados.

Album *Pedrá*. **Robe Iniesta.**

Abstract:

One of the main goals of current Robotics is to build robots that can accomplish useful tasks in non-trivial environments. During the last few years there have been many achievements in several robotics subfields, such as emotions recognition, navigation and mapping, task planning and grasping, to mention just a few. Building on these skills, roboticists aim at orchestrating them into complex software architectures that are scalable, reusable, maintainable and open to the inclusion of new skills and cognitive abilities. At the centre of these architectures lies the problem of how to represent the environment and the robot itself. Roboticists and also philosophers, psychologists and computer scientists, have devoted a huge amount of effort, and some fierce discussions also, to this matter, and although the debate will remain open for a long time to come, there is a considerable consensus nowadays that real robots, performing real, complex tasks need a grounded, well structured and adaptable world representation.

Our efforts in this direction and during the last few years have produced the robotic cognitive architecture CORTEX which is built on top of RoboComp [Manso et al., 2010], RoboLab's robotics framework. CORTEX proposes a agent-based architecture, where the agents are in charge of classical perceptual, actuation and planning abilities. The agents should be autonomous and obedient, and they are in charge of a well defined functionally -whether it be reactive, deliberative of hybrid-. They communicate through a shared data structure called Deep State Representation, a multi-labelled graph that maintains a symbolic and geometric representation of the robot, its environment and the current plan. DSR represents the robot belief about itself and the world.

The functioning of the architecture can be easily explained it we picture if as a large dynamical system. Starting in a quasi-stationary state, the perceptual agents try to keep the internal representation synchronized with the world, updating parts of it as things change. But when a new mission is requested, a plan is generated and injected into the unified representation. This alteration creates a disequilibrium to which the whole system reacts trying to restore the initial balance.

This Thesis focuses on the developing of this Deep State Representation. The goal that is pursued seeks the design of a multi-labelled dynamic graph object that can hold several levels of abstraction, from basic geometry and sensorial state to high-level symbols and predicates describing the state of the robot and the environment during a short time lapse.

The final view of the new robotics cognitive architecure CORTEX can be seen as cylindrical surface splitted in several vertical slices, which are the agents. In the centre of the cylinder, the new data structure holds the current internal belief of the robot, affecting and being affected by all the agents, and always struggling for an equilibrium that the reality fades away.

Resumen:

Uno de los principales objetivos de la robótica actual es que los robots puedan desempeñar tareas útiles en entornos no triviales. Durante los últimos años ha habido muchos logros en diferentes áreas de la robótica, como el reconocimiento de las emociones, la navegación y la localización, la planificación de tareas y la manipulación, por mencionar sólo algunas. Sobre la base de estas habilidades, los *roboticists* -investigadores en róbotica- tratan de orquestarlas en complejas arquitecturas software que sean escalables, reusables, fáciles de mantener y abiertas a la inclusión de nuevas capacidades y a la inclusión de habilidades cognitivas. En el centro de estas arquitecturas se encuentra el problema de cómo representar el entorno y al propio robot. Los investigadores en Robótica, Informática e Inteligencia Artificial, al igual que los filósofos y los psicólogos, han dedicado una gran cantidad de esfuerzo -y también algunas feroces discusiones- a este respecto y, aunque el debate sigue abierto, en nuestros días hay un consenso considerable en que los robot reales diseñados para desempeñar tareas complejas, solos o en colaboración con los humanos, necesitan una representación del entorno fundamentada, estructurada y adaptable.

Durante los últimos años, nuestros esfuerzos en este sentido han provocado el nacimiento de la arquitectura robótica cognitiva CORTEX, la cual ha sido construida usando nuestro *framework* de desarrollo RoboComp [Manso et al., 2010]. CORTEX propone una arquitectura basada en agentes donde éstos se encargan de las habilidades clásicas, como la percepción, la actuación y la planificación. Los agentes deben ser autónomos y obedientes, y están a cargo de una funcionalidad bien definida -ya sea deliberativa, reactiva o híbrida. Entre ellos se comunican a través de una estructura de datos dinámica compartida denominada *Deep State Representation*, que es un grafo dirigido multi-etiquetado que mantiene una representación simbólica y geométrica del robot, de su entorno y del plan actual. *Deep State Representation* representa la creencia del robot sobre sí mismo y sobre el mundo.

El funcionamiento de la arquitectura puede ser fácilmente explicado si nos lo imaginamos como un gran sistema dinámico. A partir de un estado cuasi-estacionario, los agentes encargados de la percepción tratan de mantener una representación interna sincronizada con el mundo, actualizando partes de ella para reflejar los cambios que acontecen. Pero cuando al robot se le solicita una nueva misión, un nuevo plan se genera como respuesta a la petición y se inyecta en la representación unificada, creando una perturbación. Esta alteración entre la realidad y la representación crea un desequilibrio y todo el sistema reacciona tratando de restablecer el equilibrio inicial.

Esta Tesis se centra en el desarrollo de *Deep State Representation*. El objetivo que se persigue es encontrar el diseño de un grafo dirigido dinámico multi-etiquetado que pueda contener varios niveles de abstracción abarcando, desde la geometría básica y el estado sensorial, a los símbolos de alto nivel y predicados, y que juntos describan el estado del robot y del entorno durante un corto espacio de tiempo.

La recién nacida arquitectura robótica cognitiva CORTEX puede ser dibujada como una superficie cilíndrica, dividida en varias rodajas verticales que representan a los agentes. En el centro del cilindro, la nueva estructura de datos contiene la creencia interna actual del robot, afectando y siendo afectada por todos los agentes, y siempre luchando por mantener un equilibrio que la realidad se empeña en desvanecer.

Índice general

Parte I

Summary

Summary of the Thesis

Introduction

The economic benefits of robotics in industry are already clear and it is expected that their inclusion in everyday life will have a tremendous impact. The EU's H2020 initiative states that, as human assistants, tomorrow's robots will have the capacity to resolve many of the future economic and social challenges faced by the European society, such as aging and well-being. However, to access these new markets and to be competitive, robots have to be dependable, smarter and able to work in closer collaboration with humans. In these scenarios, human-robot interaction (HRI) is now more commonly envisioned as a relationship among companions than a mere master-slave relationship. Nevertheless, it can be considered that the complete design of a real co-robot is beyond the scope of what can be technically achieved today. This Thesis explores initial steps towards this ambitious goal, and it proposes a software architecture that allows the robot to consider its environmental context, and the influence of external factors in its action, including the role and activity of the human interaction partner. We believe that to achieve these capabilities the robot must internalize a coherent representation of the external world and of itself, including actions, perceptions, belief states, the interacting person's intentions, and even her emotions.

Robot's abilities have improved steadily over the years, evolving as a conglomerate of modules that let the robot negotiate increasingly complex indoor and outdoor environments. As a result, current robot architectures integrate multiple sophisticated algorithms for real-time perceptual, planning and action processing, including 3D object and space recognition, simultaneous localization and mapping, navigation and task planning, manipulation and grasping, action monitoring, human detection and tracking, recognition of human activities, human-robot dialogue or action monitoring.

These skills need to cooperate with others to perform a complex task, e.g., if the robot has to approach a person, it needs at least a person detector and a navigation system to drive the robot close to her. From the point of view of component-based frameworks [Manso et al., 2010, Quigley et al., 2009], these tasks are usually addressed by dividing the problem into modules and exchanging only relevant information between them on demand. This is a common and successful architectural design choice since it promotes loosely coupled scalable systems but as tasks get more complicated and the number and complexity of the modules increases, the network of dependencies grows too fast and the whole software system becomes difficult to manage and to evolve. Each functionality in the architecture, when assigned to a software module, gets trapped inside a barrier that limits the access to a wider context provided by other modules.

Currently, the most extended design criteria to achieve complex behaviours are the hybrid

control architectures that were elegantly synthesized by Erann Gat as the *three-layered architectures* [Gat, 1998]. Although widely used, these architectures present some limitations. The most important one, from our point of view, is the need of a principled way to share information among all agents, making each of them aware to some extent of the activity of the others. For example, if a navigation module is driving the robot to a target place and a person appears somewhere close to the planned path, how does the navigation agent differentiate between an obstacle and the person, so different avoiding social behaviours can be elicited? Or how does a conversational module knows that the person the robot is talking to is not paying attention anymore, and thus a change in the discourse is advisable?

The good engineering practice of decoupling the problem in parts of *infinite impedance*, takes away a crucial element, *context*. Complex behaviours require great amounts of shared context that, eventually, have to be accessible from the modules that are solving a part of the big problem. How to do this without abandoning the decoupling design principle is one of the goals of this Thesis. It is sometimes argued that context is somehow coded in the interactions between agents [Brooks, 1991] or in the dynamics of coupled differential equations [Eliasmith, 1999]. Although those are very interesting proposals that are being currently investigated, in this Thesis we make the hypothesis that a distributed agent-based architecture can overcome the context limitation situation by using an explicit shared representation designed to provide a rich symbolic-geometric representation and to be efficiently transmitted among modules.

Classical cognitive architectures focus on general-purpose reasoning and problem solving, and on modelling and measuring human performance in those tasks. Most of them deal with higher-level cognitive processes in non-robotics domains, although there is an increasing number of exceptions [Laird et al., 2012, Puigbo et al., 2013, Hanford, 2011]. The robotics cognitive architectures are more recent creatures[1], at least from the point of view of real robots working in complex non-engineered scenarios [Haazebroek et al., 2011, Lemaignan et al., 2010, Chella et al., 2008, Sloman et al., 2006a]. Both can benefit from each other. Robotic architectures would complement cognitive ones by providing the connection to the real world through continuous, parallel and asynchronous processes, acquisition of new objects, abstract concepts, their relationship, and using their mechanism to perform the actions, learning what actions can be applied and of course the evolution in the internal state of the robot [Scheutz, 2013]. The contributions of this Thesis, along with the work of other colleagues with whom I have had the privilege to work, have given birth to CORTEX, a robotics cognitive architecture that is being used by several research laboratories on different robots and real world scenarios. This architecture and specially the internal representational structure that all agents share and help to build and that is called Deep State Representation (DSR) are the main achievements of this work.

Background

The concept of deep representations was first described by Beetz et al. [Beetz et al., 2010] as:

> representations that combine various levels of abstraction, ranging, for example, from the continuous limb motions required to perform an activity to atomic high-level actions, subactivities, and activities.

[1] According to [Levesque and Lakemeyer, 2008] the term was first introduced by R. Reiter in 1993.

Symbolic and metric representations have been proposed in many different forms and uses. Symbolic knowledge representations have been at the core of AI since its beginnings [Russell and Norvig, 2009, Poole and Mackworth, 2010] and cover all forms of relational formalizations such as production rules, frames, schemes, cases, first order logic or situational calculus. At a high level of abstraction, the Robot Learning Language (RoLL) [Kirsch, 2009] could be used for learning models about human behaviour and reactions, joint plan performance or recognizing human activity. Also, human models have been employed by the Human-Aware Task Planner (HATP) [Alami et al., 2006]. On the other hand, metric (situational) knowledge representation is often done using kinematic trees and scene-graphs [2].

Deep representations advocates the integrated representation of robot's knowledge at various levels of abstraction in a unique, articulated structure such as a graph. To the best of our knowledge, the first works that use these representations focused only on geometric data: ROS' transform library, *tf: the transform library* [Foote, 2013], BRICS Robot Scene Graph [Blumenthal et al., 2013] and RoboCog's InnerModel [Bustos et al., 2013], all appeared as a response to the need for a structured, centralized representation of robot and world kinematics. Even tough those constructions are important advances towards better robotic architectures, a richer and deeper representation was needed to hold the complete set of beliefs of the robot.

Thus, our goal is to develop a shared structure to hold the robot's belief as a combination of symbolic and geometric information. This structure would represent knowledge about the robot itself and the world around it. From an engineering point of view it should be flexible, scalable, and easy to integrate in our robotics framework RoboComp [Manso et al., 2010], allowing easy interaction among dozens of existing components.

Formally, the proposed Deep State Representation (DSR) is a directed multi-labelled graph where nodes represent symbolic or geometric entities and edges represent symbolic and geometric relationships. The robotics cognitive architecture CORTEX is defined structurally as a configuration of software agents connected through this DSR. In the next sections a more detailed description of the proposal will be presented.

Agency

The concept of agent in Computer Science and Artificial Intelligence is rather broad and their varied meanings cover most of what a program can do [Franklin et al., 1997]. In this Thesis the following definition of *agent* is employed: *A computational entity in charge of a well defined functionality (whether it be reactive, deliberative of hybrid), that interacts with other agents inside a well-defined framework, to enact a larger system.*

When several of these agents are somehow interconnected to enact a higher-level function, they conform Agent-based Architectures. In robotics and AI, they have been used for a long time [Maes, 1991, Rao and Georgeff, 1995] being Minsky's Society of Mind [Minsky, 1988], probably the most famous one.

The choice of an agent-based architecture responds to flexibility, scalability and elegance.

[2] See, for example:
URDF: http://wiki.ros.org/urdf
COLLADA: https://www.khronos.org/collada/
http://www.existentialprogramming.com/

Agents are functional units that can be easily combined in a given structure. They can be defined compositionally, as made up of other simple agents, and there is always a rather simple connection to the underlying software components. This is the **first reason** why CORTEX is an agent-based architecture.

In CORTEX agents define the classic functionalities of cognitive robotics architectures, such as navigation, manipulation, person perception, object perception, dialoguing, reasoning, planning, symbolic learning or executing. Agents have to be autonomous and obedient, both at the same time. Autonomous to provide opportunistic behaviour so non-planned events in the environment can be detected, and obedient so when new goals arrive to the system, all agents start working to achieve the current sub-goal. This flexibility will depend upon the internal organization of the agents and on their interconnection patterns. Agent-based architectures provide a rich starting groundwork to achieve these behaviours and this is the **second reason** why agent-based architectures have been chosen as the underlying computational framework in this Thesis.

Communication among agents defines the structural part of the architectures. In cognitive architectures, instead of a search for a correct model of human intelligence, what is explored is the design space of embodied intelligence [Sloman et al., 2006a]. Inside this broad space we explore here the commonly accepted assumption that there are two main flows of information. First, a deliberative one in which non-deliberative agents must provide a symbolic description of the robot and the world to the deliberative agents, so they can reason about facts and plan a course of actions to achieve the current goal. These actions are sent back to the non-deliberative agents as local goals. Second, a reactive one in which non deliberative agents react to detectable events in the world. The reactions can be internal computations or direct interaction among agents, without the intervening of deliberative elements.

An important requirement for the architecture is to facilitate the transition between deliberatively controlled behaviors and autonomous behaviors, so there can be an overall improvement of task achieving performance. Either by hand coding or by automatic learning, the way the architecture interconnects agents must facilitate the incremental creation of autonomous, efficient and reliable skills. This Thesis will show that agent-based designs, when augmented with proper information interchange mechanisms, can fulfill this demanding requirement. The next sections describe our solution to this problem.

Deep State Representation

The idea of a shared representation among agents has its roots in several classical papers [Selfridge, 1958] [Newell, 1962] [Erman et al., 1980] that developed the concept of the blackboard architecture. Later, Hayes-Roth [Hayes-Roth, 1985] extended this idea into a complete control architecture. As A. Newell himself put it,

> "Metaphorically we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it, and to judge when he has something worthwhile to add to it. This conception is just that of Selfridge's Pandemonium [Selfridge, 1958]: a set of daemons, each independently looking at the total situation and shrieking in proportion to what they see that fits their natures..."

In the original blackboard systems, agents where conceived more as problem solvers, heterogeneous experts that contribute to the overall problem in a hybrid planned-opportunistic

way. They communicate through a shared structure where goals, sub goals and problems state were incrementally updated. In CORTEX, agents solve not only deliberative tasks but also perceptual, motor and behavioral tasks, so their communication needs are somewhat different.

Nevertheless, we gather some ideas from these architectures [McManus and Bynum, 1996, Corkill, 1991] and also others from graph theory and distributed databases. We will analyse here some specific reasons why DSR has been built using a graph.

The first reason to use a graph in DSR is because all internal information defining the state of the robot and its beliefs about the environment need to be coded in a very *generic structure*. That structure is a model of how sensor data can be interpreted and organized. As general data structures, graphs can hold any relational knowledge composed of discrete elements and relations among them. In this broad category fall almost all symbolic knowledge representation methods including frames, schemes, production rules and cases, and also the geometric knowledge that the robot has to maintain about itself and the environment. This geometric knowledge includes instances of the types of objects recognizable in the world like i.e. chairs, tables, cups or generic obstacles of undefined form. Also human bodies and its parts like arms, heads, legs, etc. All these parts are kinematically related through 3D transformations forming a scene-tree.

A second reason is that the graph can be made to evolve under some generative rules. Assuming that the type of nodes and edges are predefined, the graph can evolve by inclusions or deletions of parts, causing structural changes. Also it can evolve by changing the value of the attributes stored in nodes and edges. The structural changes can be regulated by a generative grammar that defines how the initial model can change. A typical example would be that of the robot entering a new room and, after exploring it, adding a new node to the graph. In that case, the grammar would impede the new node to be connected to something else but the corresponding door. So graphs give us the capacity needed to store objects and their relations, and combined with a grammar to control its evolution, give us a coherent growing model. Fig. 1 shows how the graph evolves when a person enters the scene. In the left side only the robot and the rooms are represented. In the right side, a person enters the room and the graph incorporates her as a sub graph correctly related to the existing structure and with symbolic attributes denoting what is known about her. A more detailed description on DSR is presented in next sections.

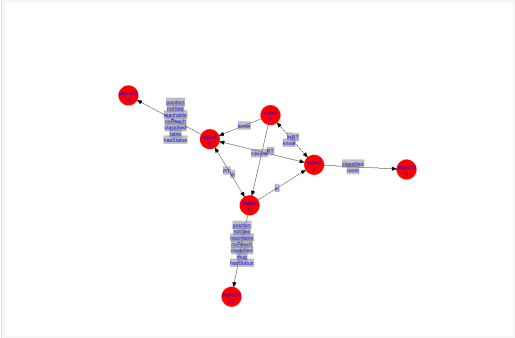
A third reason to use a graph structure is the possibility of translating it into a PDDL instance [Manso, 2013]. Depending on what is stored in the graph and the PDDL version used, there are certain restrictions but on completion state of the art planning algorithms can be freely used. Those algorithms are improved continuously by the Symbolic Planning community and provide access to a set of resources that otherwise would require a great additional effort.

A forth reason to support the choice of graphs is the facility to visualize its contents using a wide variety of tools and techniques. This is a crucial feature to debug the code of the agents, specially when interacting among them.

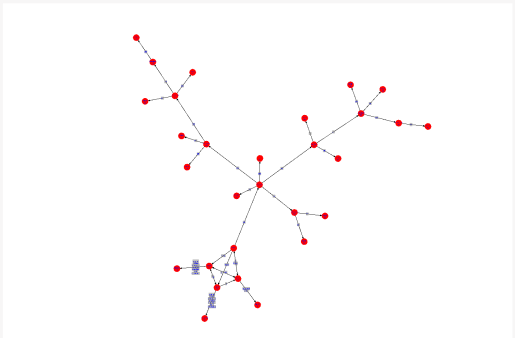
DSR Background

DSR is based and extends the *Active Grammar-based Modeling* (AGM) and the *Active Graph Grammar Language* (AGGL) proposed in [Manso, 2013]. AGM is a formalism to describe the grammar rules that can be used to modify the robot's model of the world using a domain-specific language.

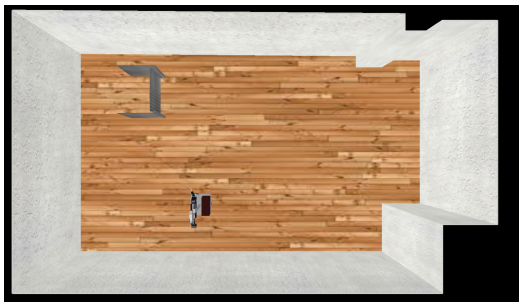
The representation of the world model at a high abstraction level is done through a semantic



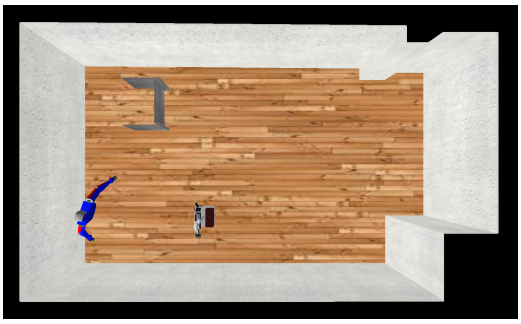
(a) Initial world model in DSR with the robot and the room.



(b) A person enters the room, and when detected by the *Person* agent, she is inserted in DSR.



(c) Graphic representation of the geometric view



(d) Graphic representation of the geometric view when a person is inserted in DSR.

Figura 1: Illustration of structural changes in DSR. The node representing the Robot has been collapsed to enhance the visibility.

graph. The nodes are symbols that represents concepts and the relationships between nodes are labelled in the edges, as predicates. Robot's missions are expressed in the same way as models, in graphs, but instead of expressing the model as a whole, they are expressed as a pattern that should be in the model when the target is met. This imposes a limitation: that all elements relating to the target must be represented explicitly in the model. The result of the planning of the mission is a set of rules that should be applied by the robot to change the graph (Fig. 2), and to reach the desired end state.

A major work of this Thesis has been to improve the above structure by adding geometric information. The new structure can hold an internal unified representation that includes both symbolic and geometric data, thus expressing more useful models of the world.

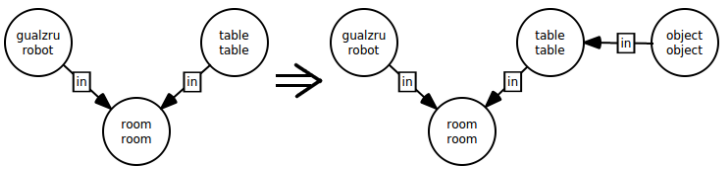


Figura 2: Example of an AGGL rule: in the robot's world model, new objects can be included in the table if the robot is in the same room as the table.[Manso, 2013]

DSR formalization

DSR is a multi-label directed graph that stores information at both geometric and symbolic level. The nodes store concepts that can be symbolic, geometric or a combination of them. Nodes can contain an indeterminate number of attributes in the form of a list of string key-value pairs. These pairs are related to aspects of the physical and geometrical representation, such as a three-dimensional mesh, the mass or the centre of gravity, a texture or the name of a person, the functional properties of an object, or the sentence said by the robot.

Edges represent relationships among nodes and are assigned labels that can store an indeterminate number of attributes as a list of string key-value pairs.

A formal definition of DSR can be given as a multi-label directed graph $G = (V, E)$ where V represents the set of nodes $\{v_1, ...v_k\}$. Each node v can store an indeterminate number of attributes as a list of string key-value pairs. $v_l = \{< key_1 : value_1 >, ... < key_n : value_n >\}$ E the set of edges $\{e_1....e_r\}$ where e_i is defined as $< u, v, l >$ where $u, v \in V$ and l is a list of string denoting the labels to that edge. Each label l can store an indeterminate number of attributes as a list of string key-value pairs. $l_l = \{< key_1 : value_1 >, ... < key_n : value_n >\}$

Geometric relations between two nodes $u, v \in V$ are represented as fixed label RT and their attributes containing the homogeneous transformation matrix expressed as $RT_{uv} = \{rx, ry, rz, tx, ty, tz\}$. The opposite direction represents the inverse matrix $RT_{vu} = RT_{uv}^{-1}$.

A path in G between two nodes $u, v \in V$ represents a symbolic path $C(u, v)$, when is through RT labels define the kinematic chain $C(u, v)$. From there, an equivalent transformation RT' can be computed by multiplying the transformations matrices. Note that the reverse path uses the inverse matrix, so it will obtain the inverse transformation.

This geometrical relations are showed in Fig. 3.

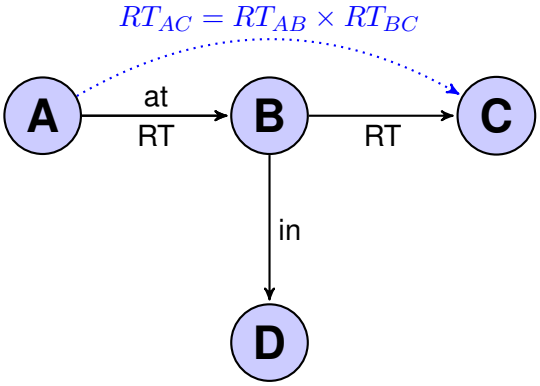


Figura 3: Unified representation as a multi-labeled directed graph. Edges are labeled “at” and “in” denoting logic predicates between nodes. Also, edges between A,B and B,C have a geometric type of label, “RT” that codes a rigid transformation between them. Geometric transformations can be chained or inverted to compute changes in coordinate systems.

The graph data structure is designed as a library to be compatible with RoboComp [Manso et al., 2010]. Its main class contains a vector of nodes and a vector of edges, in addition to the typical operations for handling nodes and edges.

The most significant functions are related to the access and update of DSR by agents. There are three operations that produce a change in DSR: insertion, deletion and modification. This

changes are structural. Alternatively, the updating of nodes' attributes and the updating of edges' attributes are non-structural changes.

Structural changes are firstly included in the local copy of the agent, that proposes this new representation of reality to be included in DSR.

Non-structural changes, that may affect nodes or edges, are made in the local copy and then updated in DSR by the agent, without requiring further supervision.

CORTEX internal organization around DSR

The functioning of the CORTEX architecture as a set of agents interacting through DSR can be easily explained if we picture it as a large dynamical system. Starting in a quasi-stationary state, the perceptual agents try to keep the internal representation synchronized with the world. But when a new mission is requested, a plan is generated and injected into the graph. This alteration creates a disequilibrium to which the whole system reacts trying to restore the initial balance. In this process the system extends its internal representation, capturing more details of the external world. The new knowledge is included as the next perturbation. We could see this process as an endogenous reaction in DSR triggered by an exogenous change in reality.

In Fig. 4, the slices that surround DSR represent the agents, that have been developed as networks of software components. They encode complete robotics functionalities -e.g. navigation, conversation, object detection, manipulation, planning and so on- and share and receive information through DSR. With elements defined and located in a static architecture, dynamism begins when the internal representation is pumped to all agents in each beat of DSR.

Each agent maintains a local copy of DSR which can be used and modified at will. The agent decides when to publish the changes introduced to the local copy, so the rest of the system becomes aware of it. Other agents may perceive the new changes as external events, similar to the events produced by directly connected sensors. Both types of events can be combined to trigger more context-aware changes in their behaviour. A typical examples is social navigation, where the Navigation agent is able to distinguish between avoiding an inanimate obstacle and a person walking by, even though it lacks the ability to detect people.

Example: simple scenario

We will use a simple example to illustrate the process of updating DSR and propagating these changes. In this example an initial graph has been created with an initial world where there are only two elements, a robot, represented with the node R, and the world represented with W (Fig. 5). In this scenario the robot aims to detect the presence of a person and then to approach her up to a suitable distance to start a conversation.

$G = (V, E)$ where $V = \{W, R\}$, $E = \{< W, R, [RT] >\}$,

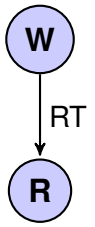


Figura 5: DSR's initial state

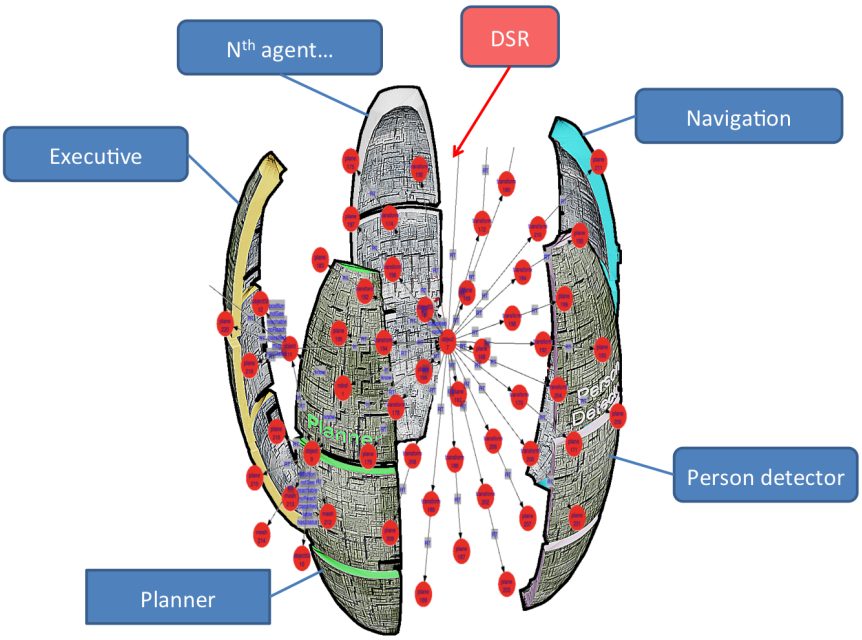


Figura 4: Overview of DSR and its location within the cognitive architecture CORTEX.

To achieve these objectives, we will require the following agents: (i) *Executive*, responsible for coordinating the actions of the plan and is aware of changes in DSR. Using the domain knowledge, it checks if a structural change is valid. If the validation is accepted the new DSR is published, (ii) *Planner*, responsible for the high-level planning of the current mission, integrating planning and re-planning, monitoring and learning abilities. (iii) *Person detector* is responsible for detecting and tracking people and (iv) *Navigation* is in charge of the robot motion and localization.

The robot and the world are represented as single nodes. Nodes and edges that define their physical structure have not been included in this example. The robot's position is defined in the edge, specifically in the label "RT" that represents the transformation between the world -the origin of the coordinate system- and the robot.

The domain knowledge is modelled as a set of grammatical rules. The first one changes the graph from its initial state (Fig. 5) to a state in which a person has been detected. Thus, when a person appears in the scene -exogenous factor-, an endogenous reaction is fired. This reaction will consist on inserting the model of a person in DSR to synchronize it with the real world. The rule is described in Fig. 6. As depicted, when a person is detected the agent will

(a) Left hand side.

(b) Rigth hand side.

Figura 6: Rule: *detect person*.

propose a change to the model. This change includes inserting the person node P and creating a geometric link between W and P, and a symbolic link between R and P with the label *detected*

(Fig. 7). This proposal should be verified by the *Executive* agent which will accept the change if it could have been generated by the grammar. If accepted, the next action will be requested to the *Planner* agent.

$$G = (V, E), \text{ where } V = \{W, R, P\} ,$$
$$E = \{< W, R, [RT] >, < W, P, [RT] >, < R, P, [detected] >\}$$

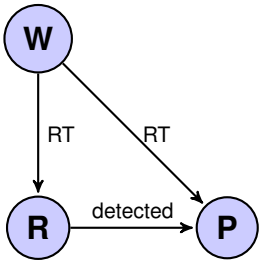


Figura 7: DSR's state when a person is detected

Once in the new state, as shown in Fig. 7, another rule is required to move the robot towards its goal: approaching the detected person. This rule is defined as described in Fig. 8. The action

(a) Left hand side.

(b) Rigth hand side.

Figura 8: Rule: *approach person*.

approachPerson needs two agents running at the same time: first, *Person detector* keeps the person position updated in the internal representation. Second, the agent *Navigation* leads the robot towards the person and also updates its position on the graph (Fig. 9). As these last changes are non-structural, updated geometric values are not checked by the *Executive* and they are immediately propagated to the rest of the architecture.

$$G = (V, E), \text{ where } V = \{W, R, P\} ,$$
$$E = \{< W, R, [RT] >, < W, P, [RT] >,$$
$$< R, P, [detected, approachPerson] > \}$$

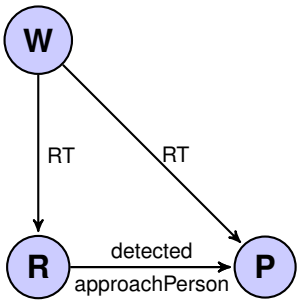


Figura 9: DSR's state when the robot approaches towards the person

From this state (Fig. 3.12) another rule is required to achieve the last goal: stopping the robot at the right distance of the person to interact with her. This rule is defined as described in Fig. 10.

The benefits of the unified representation are clearly shown in this last step. A single agent, the *Navigation* agent, proposes a change to DSR when the distance between the robot and the person is correct. The agent has access to the nodes but also to the rotation and translation of the person with respect to the robot. Therefore, when the distance is adequate it proposes a change to the inner representation that replaces the unnecessary labels *detected* and *approachPerson* by *atSocialDistance*. Fig. 11 shows the final state of DSR.

(a) Left hand side. (b) Rigth hand side.

Figura 10: Rule: *at social distance*.

$$G = (V, E), \text{ where } V = \{W, R, P\}\ ,$$
$$E = \{< W, R, [RT] >, < W, P, [RT] >,$$
$$< R, P, [atSocialDistance]\}$$

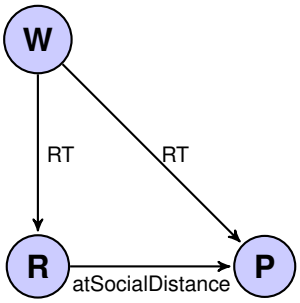


Figura 11: DSR's state when the robot is at the right distance to interact.

Therefore, this unified representation allows the agents to coordinate themselves through the actions contained in the plan. These actions are encoded in rules which provide both, the necessary conditions that have to be previously met, and the transformations that the internal model will suffer on completion of the action.

Rules can be reused in different domains. Therefore, different types of scenarios and new skills can be easily addressed through the use of different agents that use DSR to share the context of the action and combine their actions to execute different tasks.

Finally, having DSR as a central representation shared by the agents saves developing time, allows a quicker integration of cross-agent abilities and facilitates the collaborative work.

Gualzru, the robotic salesman

Gualzru is an example of a robot that uses an early version of the CORTEX architecture and is presented here as a real validation experiment of this proposal.

The robot Gualzru (Fig. 12) has been created for the ADAPTA project [3]. Its mission is to wait by an interactive advertising panel observing the people passing by. At some point, it selects a promising candidate and approaches her to initiate a conversation. After a small verbal interaction the robot is supposed to convince the person to walk back to the advertising panel, leaving the rest of the selling task to an interactive software embedded in it.

Thus, Gualzru's goals can be summarized as: *getting a* yes *or* no *answer from the potential customer after a brief conversation*. If the answer is yes, the robot will guide the client to the panel and if the answer is no, the robot will politely part from her and will look for a new candidate.

Gualzru has been repeatedly evaluated within the project in order to improve and correct their deficiencies [Romero-Garcés et al., 2015]. One of the most limiting problems in the robot's interaction with humans was the difficulty to understand and be understood when talking to the potential client. The reasons are to be found in the dimension and bad acoustic response of big empty or crowded spaces, such as malls or airports. Also, the expectation created by the robot

[3] Innterconecta Programme 2011 project ITC-20111030 ADAPTA

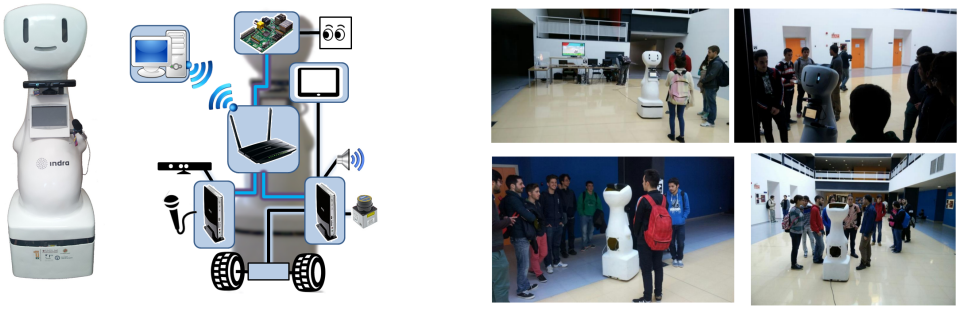


Figura 12: Gualzru and its Hardware structure. Images taken during one of the trials.

attracted many people that gathered around it, most of the time talking to each other (Fig. 6.6). These environmental conditions, taken together, caused many difficulties for the Conversational system to successfully conclude its tasks.

The solution to these issues was the inclusion of a touch screen panel in the robot's torso. This screen displays the phrases uttered by the robot to facilitate their comprehension in noisy environments. It also shows a set of answers to be chosen by the potential customer. Thus, the human-robot interaction improves since the human can communicate either by speaking or touching the panel.

If no DSR were employed the inclusion of this new interface would require whether to: (a) increase the functionalities of the Conversational system to include an interface with the touch screen panel; or (b) create a new software module to deal with the panel and connect it to the Conversational system. Either way, the Conversational system would have to be modified to deal with multiple sensory input and outputs. Also, if the tactile panel were required for additional functionalities (e.g. displaying images of offered products), they had to be whether, (a) included in the Conversational system even if they are not related to the conversation; or (b) included in other components, each of them has to be synchronized with the Conversational system to manage the usage of the panel. Both options are inefficient and they do not scale well.

The CORTEX architecture and its core, DSR, represent a more efficient, scalable and robust solution. The initial implementation of CORTEX and DSR in Gualzru has been experimentally validated and the results collected in the article: *The cognitive architecture of a robotic salesman* (in press). One of the main objectives of the work presented in this paper was to improve Gualzru's conversation attitudes using DSR. To test it, thirty-three people interacted with the robot and were asked to fill a questionnaire after the experience. The questionnaire was designed as a Likert scale but using six levels, from 0 to 5, to remove the neutral option. Zero value means a complete disagreement and five a completely agreement. Table 1 shows the two questions related to the process of understanding the robot and the human-robot interaction, along with the results obtained before including the touch screen and DSR -two rightmost columns- and after including them -two leftmost columns.

As mentioned above, another source of data was introduced to reinforce conversational skills: a Touch Screen that is managed by the *Touch screen* agent. When the sentence is uttered by Gualzru, it is simultaneously written to a node of DSR. Then, the *Touch screen* agent displays the phrase on the screen. The client is allowed to answer questions verbally or by touching on the screen. The answer to the query is written and propagated through DSR, resulting in the triggering of the next action of the plan. The interesting outcome of this approach is that the whole system has been strengthened because if the robot understands the human better, the

Question	$\bar{x}$	σ	$\bar{x}_{prev}$	σ_{prev}
2.1 Have you understood the robot?	4.27	1.23	3.57	1.28
2.2 Has the robot understood you?	3.72	1.28	2.70	1.30

Cuadro 1: Detail of the questionnaire results for 33 tests. The rightmost columns show the result obtained in previous tests and leftmost columns the results after DSR [Romero-Garcés et al., 2015].

whole system will improve its response, showing a new skill and without barely altering other parts of the architecture. The DSR is representing the robot's perceived reality but now the same state -human response- can reach inside by means of two different sources, and yet the rest of the agents in the architecture become aware of it and can act accordingly. Moreover, additional sensory inputs/outputs -e.g. responses provided using body language and gestures- can be easily added to the system by implementing new agents that connect to these nodes, without having to change already implemented agents.

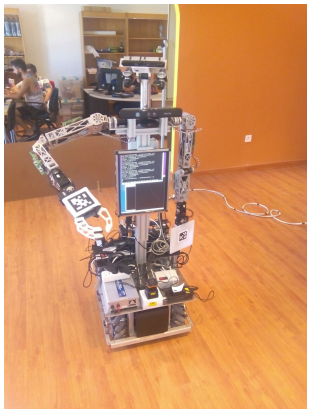
Shelly, an autonomous mobile manipulator

Shelly is an anthropometric robot that is currently being used in RoboLab, at the University of Extremadura. It is a social robot designed to help in daily life tasks. It is composed of an omnidirectional base, two 7-DOF arms with two-fingered grippers and a RGB-D camera attached to a pan-tilt-yaw structure. It has another RGB-D camera on the upper part of the torso which is used to detect human bodies and a lidar for navigation.

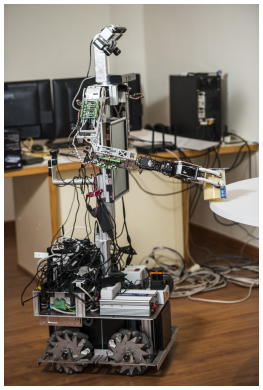
The purpose of this experiment is to describe a complex but very common task in our lives: the robot must pick up an object from a table -a mug- and bring it to a person. The experiment validates the internal working of the architecture since the task requires interactions from planning, perception, manipulation, grasping, navigation and monitoring within the context of an autonomous mobile manipulator. The following agents have been coded and deployed for the experiment:

- *Executive* coordinates the actions of the plan and checks the validity of structural changes in DSR, using the domain knowledge. If the validation is accepted the new DSR is published.
- *Planner* is responsible for the high-level planning of the current mission, integrating planning and re-planning, monitoring and learning abilities.
- *Person detector* detects and tracks people.
- *Navigation* is in charge of the robot motion and localization.
- *Object detector* perceives the orientation and position of objects.
- *Proprioception* keeps updated the internal representation of the robot with the sensory information of its pose.
- *Grasping* coordinates and executes the gripping action.

The experiment has been constrained to the robot's current agent functionalities. For example, we use AprilTags [Olson, 2011] marks to reliably detect objects. Fig. 13c, shows the room with a table that is labelled with a mark on its center. On the table there is a cylindrical object -a mug- also labelled with a mark that is to be grasped and lifted by the robot (Fig. 13b).

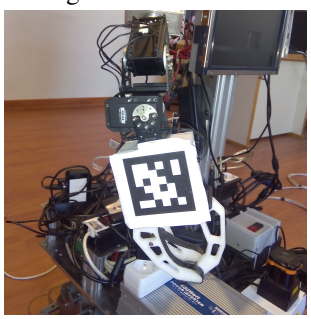


(a) Shelly general view.



(b) Shelly grasping the "mug".

(c) Table and *mug*. The AprilTag marks identify and locate the objects.

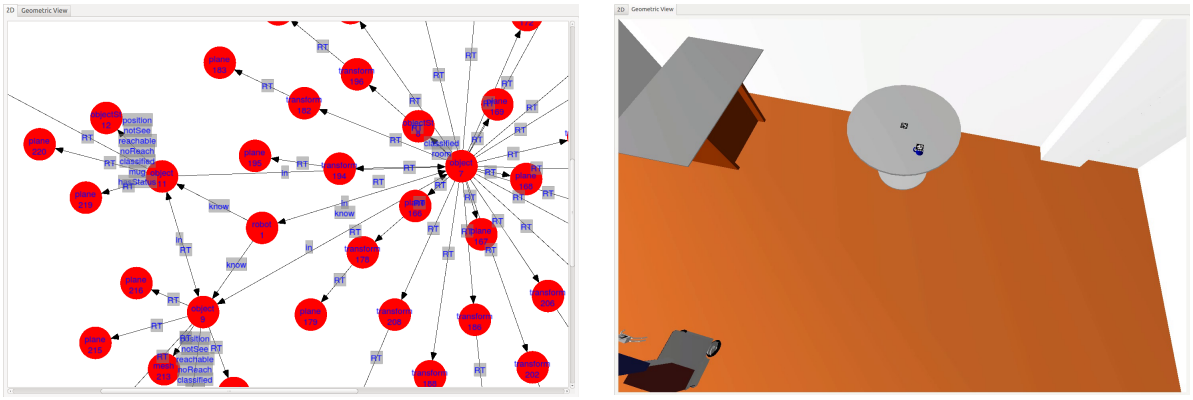


(d) Wrist and finger, detail view. The AprilTag mark helps to locate the robot's wrist.

Figura 13: The mobile manipulator Shelly at *RoboLab*

The task which has to be accomplished by the robot starts with the robot receiving a verbal command from the person, then going to the proximity of the table, and finally picking up the mug and bringing it to her. The DSR, at the start of this experiment, includes a number of elements as a priori knowledge of the environment (Fig 6.9). Geometric nodes and edges include walls, the table, the mug and the structure of Shelly. There is also a fixed obstacle in the environment, a table without mark. This object cannot be perceived by the *Object detector* agent, but the *Navigation* agent can, and uses it as an obstacle to be avoided when moving around. As Fig. 14b shows, the robot knows that there is one table an one mug in the room, but the position of the mug in DSR is different from its real position (Fig. 13c), since it has not yet been perceived by the *Object detector* agent.

The experiment starts when the robot receives a voice command from the human. Once the mission is active the *Planner* agent receives a target pattern and the current DSR's state. After computing a plan, the *Planner* agent provides a sequence of actions or rules that will transform the current state to the final desired state described in the pattern. The *Executive* agent sends the first rule to all agents. The rule is shown in Fig. 15.



(a) DSR two-dimensional view.

(b) DSR Geometric view.

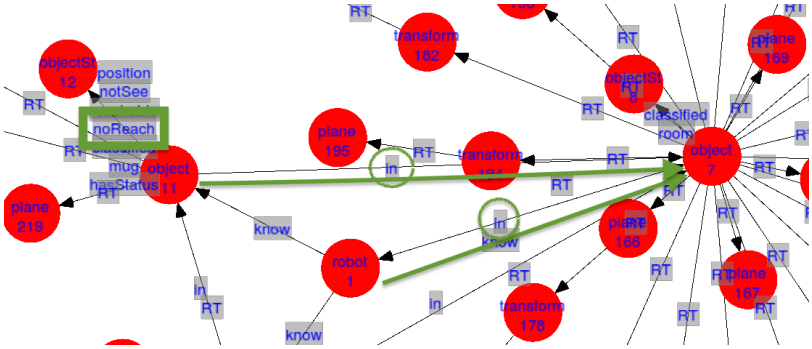
Figura 14: Detail view of DSR. Two-dimensional and geometric view.

This rule describes DSR's state conditions (Fig. 15a) and the actions to perform when the conditions are met (Fig. 15b). The operation is simple. The *Grasping* agent checks DSR and uses the geometric information to compute the distance between the position of the robot -updated by the *Navigation* agent through non-structural changes- and the position of the mug -updated by the *Object detector* agent also through non-structural changes-. When the distance is correct, the *Grasping* agent will modify DSR state changing the label from *noReach* to *reach*.

It can be seen that the left side of the rule (Fig. 15a) is a pattern that is already in the graph (Fig. 15c), which will make the agents work to satisfy the right side of the rule (Fig. 15b). At the same time the *Object detector* agent and the *Person detector* agent could be updating DSR including new objects in the environment.

(a) Left hand side.

(b) Rigth hand side.



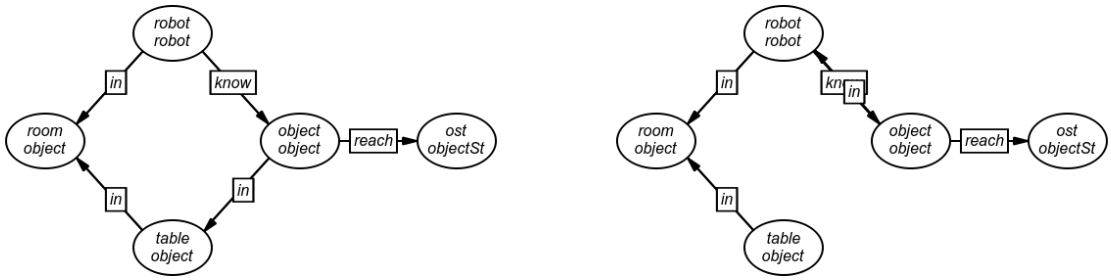
(c) We only draw the rule's left hand side in DSR for better viewing.

Figura 15: Rule: *SetObjectReach*. The *Navigation* agent approaches the robot to the mug and the *Grasping* agent changes the label *not reach* to *reach* when the robot is at a suitable distance.

Once the robot gets close to the mug the next rule is fired (Fig. 16). The robot is ready for grasping as the *Grasping* agent has already computed the 3D grid coding the arm reachable space. Each 3D position of this space is encoded as a node in the working copy of the agent

(black boxes in Fig. 17a), containing the euclidean positions and the corresponding values of the arm DOFs that allow the robot to reach them. The agent then will check different paths to the goal (purple box in Fig. 17a), and select the closest one. Thus, DSR allows to perform short-scale predictions and evaluate the consequences of actions, at the situational or sensorimotor level.

This is an example of a set of *internal emulations* executed in a local, working copy, without affecting the rest of agents nor DSR. In fact, some elements in this working copy -i.e. the nodes representing reachable positions- are not present and will not be updated in DSR once the agent decides the course of action. Internal emulation is thus a powerful resource that allows the agent to use its local copy of DSR to anticipate the outcome of its actions. In doing so, the agent simulates the outcome of actions with synthetic perceptions and executes tentative actions before commanding the robot to actually do them.



(a) Left hand side.

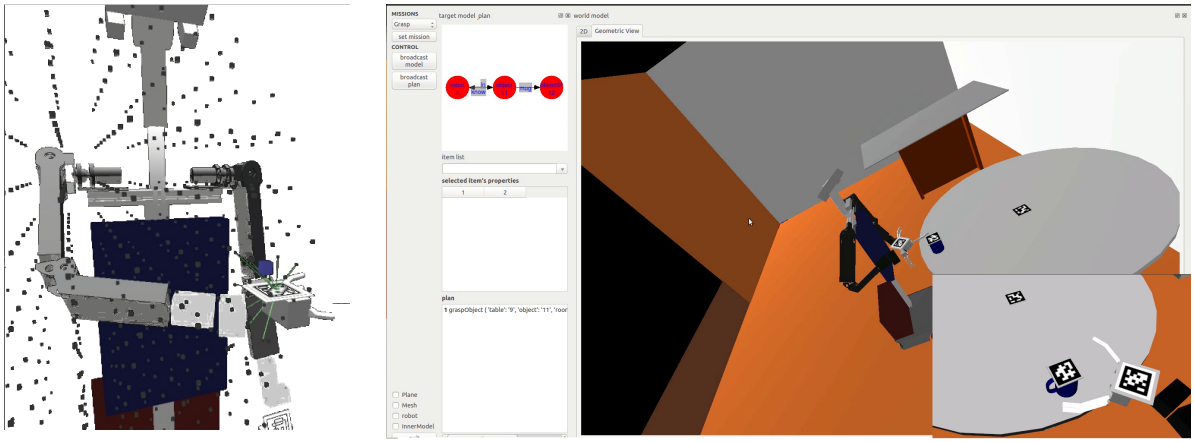
(b) Rigth hand side.

Figura 16: Rule: *graspObject*. The *Grasping* agent in coordination with the *Object detector* agent are the main actors for this action.

One goal that is achieved in this step of the plan is to bring the hand inside the frustum of the camera in what can be considered a *saccadic* movement. Once the hand and the target are both visible, visual servoing techniques are employed to cancel out the misalignment effects of mechanical backslash and calibration uncertainty. The *Grasping* agent will check the real positions -provided by AprilTag marks- of the hand and the mug, and will use them to update the positions in the working copy of DSR. This is a classical example of visual servoing used here to cancel calibration and backslash errors.

Once the hand reaches the correct pose it grasps the mug and, afterwards, the agent updates DSR with the structural changes defined in the rule showed in Fig. 6.11. The operation consists on delete the edge labelled with *in* between the mug object and the table object and create a new edge labelled with *in* between the mug object and the robot.

Once the mug has been grasped and DSR has been updated, the robot handles the mug to the human. The *Person detector* agent is in charge of detecting a person in the surroundings of the robot and of updating her location in DSR. The *Navigation* agent will use the person position as a new goal for the robot motion. Once the robot is close to the person, it offers her the mug, Fig. 18. A voice command from the person will finish the use case forcing the robot to release the mug.



(a) Local emulation.

(b) DSR's state after the execution in the reality.

Figura 17: First movement towards a position close to the mug. Lower right corner. Synthetic detail of the view from the head's RGBD camera.

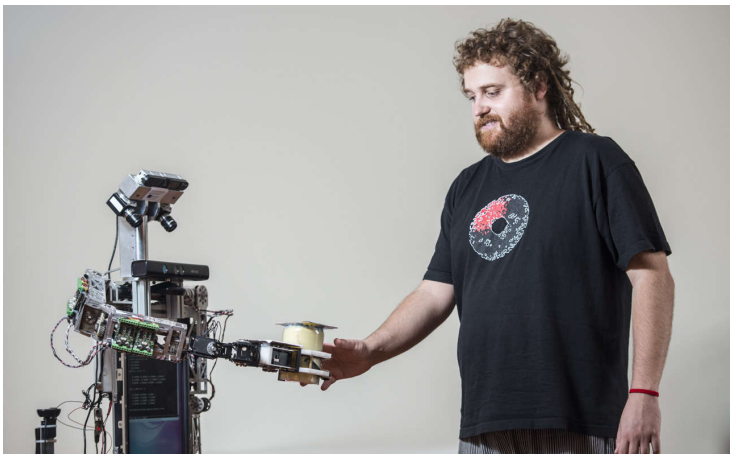


Figura 18: The author receives the "mug".

Conclusions

This summary has presented CORTEX as an agent-based cognitive architecture that uses a central, shared representation named DSR as its core representation and communication support. DSR has been defined, formalized and described in detail, explaining its multi-graph structure, its origin and the related scientific papers that have been published about it. This hybrid representation has been shown to work efficiently as a vehicle for reasoning, as well as for the representations of beliefs.

DSR is the major achievement of this Thesis, and it is its condition as an unified representation encompassing geometric and symbolic knowledge that can evolve in the robot's time according to a set of domain rules, what makes it an useful tool for the future development of cognitive robotics. The most relevant features of DSR can be summarized in three key points:

- *Structural*: every element, geometrical or symbolic, is inserted in a particular place and must be related to other elements. This approach wildly differs from the more frequent distributed system lacking a central representation and in which agents exchange a great

amount of messages or topics to keep a partial representation of the robot beliefs.

- *Generative*: the size of DSR changes as space, objects, people, etc. are added or removed, but this dynamics is constrained by a set of rules and patterns, all sharing a common dictionary of names.
- *Adaptive*: DSR is continuously changed by the agents, updating links and nodes, as when the mug is grasped and it becomes the child of the Hand. Changes in on side of the graph are propagated and made visible to all agents inthe architecture, providing the necessary information to perform context-aware computations.

The CORTEX architecture, with DSR in its core, represents a step towards a more comprehensive, coherent, functional and elegant software architecture for an intelligent robot. It also facilitates the reuse of complex code, saving time and money. For instance, many of the plan's actions, or even entire missions are repeated in different settings with minimal changes to the architecture and to most of the code base developed so far.

The CORTEX architecture also opens many possibilities and questions for further for research. For example, regarding the validation of the changes using the grammar, how will be geometric changes validated? How to verify that they are "legal" in a given context? In the current implementation each agent is responsible to validate its proposals, but a more advances architecture will include an agent specialized in geometric computations and planning, as is suggested in [Alili et al., 2010] o [Kaelbling and Lozano-Pérez, 2013].

The DSR mixes the geometric and symbolic levels and thus it could be the starting point to generate an episodic memory. This idea raises several questions: would DSR be rich enough to conform such a memory or will it need additional elements? What is to be stored and what forgotten? How would we find in the episodic memory, the memories that match the current state of the representation? Is the search for patterns in the graph the best approach? We believe that all these questions can be analysed from the perspective of DSR and the CORTEX architecture

Another interesting discussion is related to the granularity of DSR. If raw data from sensors were directly inserted into the graph, a more homogeneous system would result but at the cost of an enormous requirement of band width among agents. In order to achieve this objective new research has been already started in our group exploring ways of using proxies to substitute real copies and maintaining the same functionality. Strictly speaking, the data will not be stored but a similar effect is achieved using the plasticity and dynamic properties of our component-oriented framework.

To sum up, if DSR is correctly employed to encode each situation that the robot faces, it is possible to build around it new agents or use several of them to evolve the cognitive architecture towards the new challenges these situations represent.

Parte II

Tesis

Capítulo 1

Introducción

Los beneficios económicos de la robótica en la industria ya han quedado demostrados en multitud de aplicaciones y ámbitos de trabajo. La robótica social, sin embargo, está apenas empezando y se espera que tenga un impacto enorme en nuestra vida diaria. La iniciativa de investigación e innovación *Horizon 2020* de la Unión Europea establece que, como asistentes humanos, los robots del futuro tendrán la capacidad de resolver muchos de los desafíos económicos y sociales a los que se enfrentará la sociedad europea, como el envejecimiento y el mantenimiento del bienestar social. Sin embargo, para acceder a estos nuevos mercados y ser competitivos, los robots tienen que transmitir confianza, ser inteligentes y capaces de trabajar en estrecha colaboración con los seres humanos. En ese escenario, la interacción humano-robot (HRI) es, muchas veces, más parecida a un relación entre compañeros, que a una relación entre un operario y una herramienta. Ciertamente, se puede pensar que el diseño completo de un verdadero compañero robótico está más allá del alcance de lo que puede lograrse con la tecnología actual.

Esta Tesis explora los pasos iniciales hacia ese ambicioso objetivo y propone una arquitectura software que dirija la forma en que el robot tiene en cuenta su contexto ambiental y circunstancial, así como los factores externos que influyen en sus acciones, sin olvidar la inclusión del ser humano en el bucle, su actividad, comportamiento y papel desempeñado en la tarea, como única vía para su colaboración mutua.

Es cierto que las habilidades de los robots han mejorado constantemente a lo largo de los años, evolucionando como un conglomerado de módulos que le permiten lidiar con entornos interiores y exteriores cada vez más complejos. Como resultado, las arquitecturas robóticas actuales integran multitud de algoritmos sofisticados para la percepción en tiempo real, la planificación y el procesamiento de la acción, incluyendo el reconocimiento de objetos 3D y del espacio, la localización y el modelado simultáneo del entorno (*SLAM*), la navegación y la planificación de rutas, la manipulación y el *grasping*, la monitorización de la acción, la detección y captura del movimiento humano, el reconocimiento de las actividades humanas o el diálogo humano-robot, entre otras muchas.

Pero también es cierto que estas habilidades necesitan cooperar entre sí para realizar una tarea compleja. Por ejemplo, si el robot tiene que acercarse a una persona necesitamos al menos un detector de personas y un sistema de navegación para desplazar el robot hasta la posición de la persona. Desde el punto de vista de los *frameworks* robóticos basados en componentes [Manso et al., 2010, Quigley et al., 2009], es común abordar estas tareas dividiendo el problema en módulos e intercambiando sólo la información relevante entre ellos cuando se precisa. Este

diseño arquitectónico es común y obtiene buenos resultados debido a que promueve sistemas escalables débilmente acoplados. Sin embargo, cuando la tarea se vuelve más complicada y el número y complejidad de los módulos aumenta, la red de dependencias crece demasiado rápido y el sistema entero se torna difícil de manejar y de hacer evolucionar. Cuando cada funcionalidad de la arquitectura es asignada a un módulo software, ésta acaba siendo atrapada dentro de una frontera que limita el acceso al contexto más amplio que proporcionan los otros módulos.

En la actualidad, los criterios de diseño más extendidos para lograr comportamientos complejos son las arquitecturas de control híbrido que fueron elegantemente sintetizadas por Erann Gat como *three-layered architectures* [Gat, 1998]. Sin embargo, estas arquitecturas presentan algunas limitaciones. La más importante, desde nuestro punto de vista, es la necesidad de regular y estructurar la forma de compartir la información entre todos los agentes, haciendo a cada uno de ellos consciente, en cierta medida, de la actividad de los otros. Por ejemplo, si un módulo de navegación está desplazando el robot hacia un destino y una persona se cruza en algún sitio cerca o interrumpiendo la trayectoria planificada, ¿cómo diferenciaría el agente a cargo de la navegación entre un obstáculo común y la persona?, ¿podría exhibir un comportamiento social -*social behaviour*-, evitando a la persona de forma diferente a como lo haría con el obstáculo? o ¿cómo puede un agente encargado de la conversación saber si la persona a la que está hablando, no le está prestando atención, y en ese caso cambiar su discurso?

La buena praxis de la ingeniería que persigue conseguir un bajo acoplamiento dividiendo el problema en partes de *impedancia infinita* olvida, en nuestra opinión, un elemento crucial, el *contexto*. Los comportamientos complejos requieren grandes cantidades de contexto compartido que en algún momento tendrá que ser accesible para los módulos que están resolviendo solo una parte del problema general. Cómo hacer esto sin abandonar el principio de diseño del desacoplamiento, es uno de los objetivos de esta Tesis. A veces se argumenta que el contexto está codificado de alguna manera en las interacciones entre los agentes [Brooks, 1991] o en la dinámica de ecuaciones diferenciales acopladas [Eliasmith, 1999]. Aunque esas son propuestas muy interesantes que se están investigando actualmente, en esta Tesis proponemos la hipótesis de que una arquitectura distribuida basada en agentes puede superar la limitación del contexto usando una representación compartida explícita, diseñada para proporcionar una representación rica, tanto simbólica como geométrica del entorno que pueda ser trasmitida eficientemente entre los agentes.

Encontramos en la literatura que tradicionalmente las arquitecturas cognitivas se han centrado en el razonamiento de propósito general y la resolución de problemas, así como en el modelado y la medida del rendimiento humano en esas tareas. La mayoría de ellas se centran en los procesos cognitivos de alto nivel y en ámbitos ajenos a la Robótica, aunque hay un creciente número de excepciones [Laird et al., 2012, Hanford, 2011]. Las arquitecturas para Robótica, por otro lado, son estructuras más recientes[1], al menos desde el punto de vista de la implementación en robots reales y en escenarios complejos [Choi et al., 2009, Haazebroek et al., 2011, Lemaignan et al., 2010, Chella et al., 2008, Sloman et al., 2006b].

Ambas pueden beneficiarse mutuamente. Las arquitecturas para Robótica complementarían a las cognitivas, proporcionando la conexión con el mundo real a través de procesos continuos, paralelos y asíncronos que permitiesen la adquisición de nuevos objetos o conceptos abstractos, sus relaciones, y usando sus mecanismos para completar las acciones, aprendiendo, incluso, cuáles, cómo y cuándo se pueden aplicar y, por supuesto, permitiendo la evolución del estado

[1] De acuerdo a [Levesque and Lakemeyer, 2008] el término fue introducido por primera vez por R. Reiter en 1993.

interno del robot [Scheutz, 2013].

Las contribuciones de esta Tesis, junto con los estudios previos realizados por el grupo de investigación en el que se desarrolla este trabajo, han provocado el reciente nacimiento de CORTEX, una arquitectura cognitiva para Robótica que está siendo utilizada por varios laboratorios de investigación en diferentes robots y escenarios del mundo real. Esta arquitectura y en especial la estructura de representación interna que es compartida y construida por todos los agentes, y que hemos denominado *Deep State Representation* (DSR) son los principales logros de esta Tesis.

1.1. Motivaciones

La motivación de la propuesta de esta Tesis surge tras los años de trabajo en diferentes proyectos de Robótica. En el año 2009 al terminar mi proyecto fin de carrera, entendí que la programación orientada a componentes era una buena idea para desacoplar el problema y tratar de construir sistemas que, interconectados, pudieran abarcar nuevos retos. Al mismo tiempo, fomentaban el trabajo colaborativo y así diferentes ingenieros e investigadores podían desarrollar y encapsular sus conocimientos en componentes, definir una interfaz y poder usarlo junto al de otros.

Sin embargo, a lo largo de los años, las limitaciones de la programación orientada a componentes se han ido manifestando, fundamentalmente desde el grupo de investigación siempre se ha tenido el objetivo de construir una solución general que pudiese ser aplicada a diferentes robots, distintos escenarios y casos de uso a los que nos enfrentaban los proyectos de investigación. Así, se han abarcado diferentes ámbitos que van desde el mundo de la rehabilitación al del marketing y venta, pasando por la Robótica educativa entre otros.

El *framework* basado en componentes -RoboComp- desarrollado por el grupo respondió bien ante esas situaciones pero se empezó a comprobar que, a medida que se trasladaba el desafío del laboratorio al mundo real, se hacía más complejo reutilizar el control del alto nivel entre las diferentes tareas. Cada vez se hacía más difícil interconectar componentes. La proliferación de componentes intermedios que combinaban información de otros y que terminaban su ciclo de vida en ese proyecto era inevitable. En la fase de diseño era casi imposible especificar todas las conexiones de antemano, ya que las necesidades evolucionaban con el desarrollo del proyecto.

El problema de las conexiones predefinidas, la definición de las interfaces y la especificación de qué era lo que debía ser compartido entre los componentes, llevaba siempre a la ausencia de contexto compartido entre los componentes. Cuando se desarrollaba un componente o grupo de ellos para hacerse cargo de una tarea, sistemáticamente otro necesitaba alguna información adicional que no estaba disponible localmente. Para una mayor versatilidad se desarrollaron herramientas que permitían especificar y (re)generar el componente sin alterar los algoritmos ya implementados, con el objetivo de poder adaptarlos rápidamente a nuevos requerimientos. Pero siempre llegaba un momento en la red de componentes crecía más allá de un punto en el que se tornaba complicado seguir, entender y depurar las interacciones dinámicas que se generaban al ejecutarse todos juntos. En ese punto, la necesidad de una *representación común* comienza a emerger. Dicha representación ha de tener en cuenta la variada naturaleza de la información compartida entre los agentes, la necesidad de sincronización, y también la rapidez de respuesta requerida en sistemas que controlan el comportamiento de robots, que en muchos

casos trabajan en entornos cotidianos.

La motivación principal de esta Tesis ha sido encontrar y construir esa representación.

1.2. Contribuciones

La principal contribución de esta Tesis es *Deep State Representation* (DSR), un grafo dirigido multi-etiquetado que mantiene una representación simbólica y geométrica del robot, de su entorno y del plan actual. El desarrollo de esta estructura ha permitido el reciente nacimiento de CORTEX, una arquitectura cognitiva para Robótica que está siendo utilizada por varios laboratorios de investigación en diferentes robots y escenarios del mundo real.

DSR es una representación interna de la realidad y del robot, detallada y útil, ya que unifica la naturaleza simbólica y geométrica del entorno en una misma estructura en forma de grafo. La estructura permite representar la creencia interna del robot y su mundo.

Al mismo tiempo, la estructura DSR es usada por el robot para razonar y decidir el plan necesario para llevar a cabo la tarea en cuestión. Es compartida por todos los agentes de CORTEX, permitiendo combinar sus habilidades y compartir el contexto de la acción.

1.3. Estructura del documento

Esta tesis está organizada en las siguientes partes:

En el capítulo 2 se describe el estado de la cuestión. En el se encuentran los pilares y razones que han llevado a la propuesta de CORTEX y DSR para tratar de alcanzar una verdadera arquitectura para Robótica Cognitiva.

En el capítulo 3 se detalla el porqué de la propuesta de DSR, así como los trabajos relacionados, sus funcionalidades y mecanismos para permitir a los agentes afrontar los desafíos de la Robótica actual.

En el capítulo 4 se detalla uno de los agentes clave en la interacción humano-robot, un detector de personas. Se describe su funcionamiento y la representación del humano en DSR. Los algoritmos internos de este agente permitieron generar una importante contribución científica del autor y ha sido crucial para el desarrollo geométrico de DSR.

En el capítulo 5 se realiza una pequeña descripción del resto de agentes que conforman la arquitectura CORTEX. El capítulo destaca que algunos de ellos integran el conocimiento de especialistas en la materia. También hace hincapié en las posibilidades de extensión de CORTEX gracias a DSR, ya sea mediante la incorporación de nuevos agentes o mediante la combinación de los existentes.

En la primera parte del capítulo 6 se describe la evolución del robot Ursus. Su desarrollo condujo a la conclusión de la necesidad de una representación compartida.

En la segunda parte se describen dos robots que usan DSR y CORTEX para desempeñar su tarea: el robot vendedor Gualzru, que incorpora esta arquitectura en su última versión y un robot manipulador móvil, al que hemos llamado Shelly. Como experimento, se propone el caso de uso complejo que representa la tarea *tráeme la taza*. Esta tarea integra un elevado número de fuentes de información y actuadores y, al mismo tiempo, precisa de las habilidades de planificación, coordinación e integración de diferentes y diversas habilidades. Se detalla como la propuesta de esta Tesis ayuda y simplifica su realización.

En el capítulo 7 se exponen las conclusiones alcanzadas tras el desarrollo de la propuesta.

En el capítulo 8 se perfilan las líneas de investigación futuras que ha generado el estudio y la realización de la Tesis.

En el capítulo 9 se listan las publicaciones del autor y su relación con la consecución de los objetivos de la Tesis.

Capítulo 2

Estado de la cuestión

La arquitectura CORTEX persigue un diseño funcional, adaptable, reusable y escalable. Para ello recoge y conjuga diferentes ideas y elementos de la Inteligencia Artificial y de la Robótica. En su concepción se pretende construir una arquitectura de propósito general que aúne las funcionalidades distribuidas, asíncronas y de ejecución en tiempo real, propias y necesarias de la Robótica actual, con una estructura compartida de representación de la creencia del robot. En este capítulo se revisarán conceptos e ideas de la Inteligencia Artificial, la Ciencia Cognitiva, la Robótica y la Informática, que han permitido diseñar y construir CORTEX y *Deep State Representation*.

2.1. Modelos de pizarra, *blackboard*

El concepto de *blackboard* tiene su origen en varios artículos clásicos [Selfridge, 1958, Newell, 1962]. De acuerdo con [Nii, 1986] la primera referencia al término *blackboard* en la literatura de la Inteligencia Artificial aparece en Newell [Newell, 1962]:

> "Metaphorically we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it, and to judge when he has something worthwhile to add to it. This conception is just that of Selfridge's Pandemonium [Selfridge, 1958]: a set of daemons, each independently looking at the total situation and shrieking in proportion to what they see that fits their natures..."

Newell investigaba los problemas de organización de programas de esa época, como por ejemplo, los programas que jugaban al ajedrez o los demostradores de teoremas como respuesta al flujo de control secuencial y jerárquico predominante. La propuesta de la pizarra pretendía dotar al sistema de mayor flexibilidad en el control permitiendo a las subrutinas acceder a una estructura de datos común.

Por tanto, es esencialmente un modelo de resolución de problemas, esto es, un esquema para organizar las etapas del razonamiento y el conocimiento del dominio. Su objetivo final es construir la solución a un problema. El meollo con el que lidia la resolución de problemas es con la selección de qué parte del conocimiento se debe usar, cuándo se debe usar y cómo se debe usar [Nii, 1986].

El modelo de pizarra *blackboard model* es un caso especial de la resolución de problemas oportunista -aquél que aplica la estrategia de razonamiento más apropiada en cada momento-.

Además, como una aplicación estratégica del conocimiento, el modelo prescribe la organización del conocimiento del dominio, así como, todas las entradas y soluciones parciales e intermedias necesarias para resolver el problema. El espacio de la solución está representado por todas las posibles soluciones parciales o completas del problema.

Históricamente, el modelo de pizarra como herramienta para la resolución de problemas surge de la abstracción de las características de HEARSAY-II [Erman et al., 1980], que es un sistema de reconocimiento del habla desarrollado entre 1971 y 1976. HEARSAY-II entendía una pregunta realizada por un humano sobre la ciencia de los computadores. La comprendía en el sentido de que era capaz de contestar a a la pregunta a partir de la información contenida en la base de datos. En ese trabajo el modelo de pizarra es descrito como un conjunto que consta de tres componentes principales:

- *Knowledge Sources KSs*. El conocimiento necesario para resolver el problema es fragmentado en fuentes de conocimientos separadas e independientes.
- *Blackboard structure BS*. Los datos del estado de la solución del problema son almacenados en un basa de datos global. Las fuentes de conocimiento producen cambios en la pizarra dirigidos incrementalmente hacia la solución del problema. La comunicación e interacción entre las fuentes de conocimiento se hace a través de la pizarra.
- *Control*. De forma oportunista, selecciona las fuentes de conocimiento que mejor responden a los cambios en la pizarra.

Este modelo conceptual es explicado magníficamente por [Nii, 1986] usando dos metáforas. La primera es la de un grupo de personas que cooperan para construir un puzle. La segunda, más adaptada a las implementaciones y usos reales, describe cómo se resolvería el reconocimiento e identificación de un koala en un árbol usando este modelo de resolución de problemas. En la primera, la pizarra es un puzle enorme y cada persona dispone de un conjunto de piezas. Estas están sentadas en frente de la pizarra y observan el estado del puzle. Cada persona compara sus piezas con el estado actual del puzle y coloca su pieza en la posición correspondiente. Cada intervención independiente provocará que el estado del puzle -de la pizarra- cambie. Esto desencadenará que otra persona coloque otra pieza de su conjunto en ella y así sucesivamente, hasta completar el puzle. Es interesante observar cómo el problema se podía haber realizado en completo silencio, sin comunicación directa entre las personas, simplemente atendiendo a los cambios de la pizarra. Cada persona de forma oportunista, evalúa la pizarra y sus piezas y aporta su conocimiento colocando una más. Esto conduce de forma incremental al conjunto entero hacia la resolución del problema.

Se plantean algunas cuestiones relativas a la selección de las fuentes de conocimiento, que en este caso son las personas. Por ejemplo, ¿cuántas piezas se le permite poner a la persona?, ¿cuál es el criterio para seleccionarla? etc. Esto permite la introducción del tercer elemento, el control, como administrador de las fuentes de conocimiento. Además, el algoritmo de control es clave debido a que en la época, la única forma de trasladar a la realidad esta entidad conceptual era mediante la programación en serie.

La metáfora del koala está mas dirigida a la organización real del conocimiento dentro de la pizarra. En pocas palabras el problema consiste en localizar e identificar un koala en un árbol. Para ello se dirige la vista hacia una zona del árbol. Un fuente de conocimiento ha elegido ese lugar, por su alta probabilidad de que allí se encuentre un koala, apoyándose en otras fuentes de conocimiento relacionadas con el color, bordes visuales, formas o partes del cuerpo.

El contenido de la pizarra, que al principio, podía ser entendido como una mancha borrosa, se va aclarando y va tomando forma gracias al conocimiento de los especialistas. Desde el nivel inferior, basado en simples líneas, regiones o formas, pasando por uno intermedio donde se produce el reconocimiento de las partes, patas, torso, brazos..., hasta uno superior, donde emerge el koala basado en la conjunción de sus partes. Cada especialista, puede utilizar el conocimiento de otro para suplir la falta de información. Por ejemplo, el detector de brazos necesita que en la pizarra, el detector de segmentos haya aportado su conocimiento. En ese instante el control lo activará. Esto provocará que lo que era una forma en la pizarra se convierta en un brazo. El proceso continuará hasta que en la pizarra se represente la solución del problema.

2.1.1. Ejemplos históricos

Desde una perspectiva ingenieril el concepto ha sido trasladado a diferentes implementaciones. Las dos siguientes muestran claramente el patrón de los elementos definidos y su organización, lo que sirvió para definir la estructura general de las arquitecturas de pizarras.

2.1.1.1. HEARSAY-II

El objetivo de HEARSAY-II [Erman et al., 1980] era comprender y reconocer expresiones del habla. Su tarea era responder a preguntas sobre una colección de resúmenes en el área de la Inteligencia Artificial, a la vez que recuperaba el documento. La Figura 2.1 muestra un esquema de la arquitectura donde se identifican sus componentes principales.

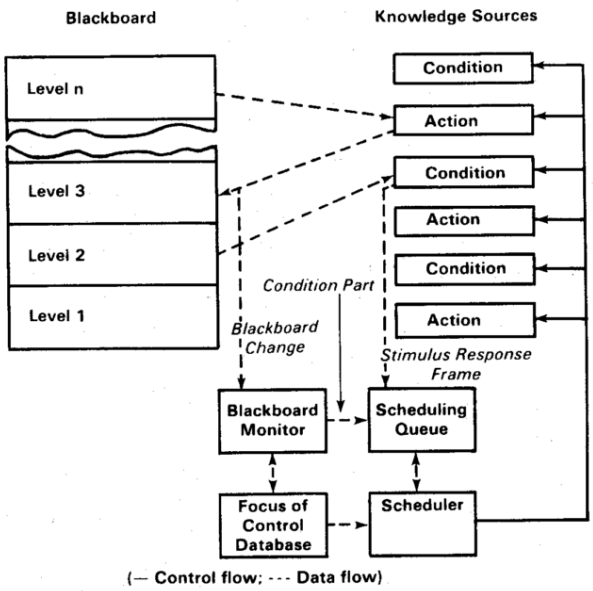


Figura 2.1: Esquema de la arquitectura HEARSAY-II [Nii, 1986].

Blackboard. La estructura de la pizarra esta dividida en varios niveles de análisis que corresponden con los distintos niveles del proceso de decodificación de la frase recibida. Los niveles son jerárquicos y cada nivel depende de la información del anterior. La entrada es una señal de audio y su nivel más bajo es el parámetrico y el nivel más alto el conceptual -la trascripción del audio-.

Knowledge source. Cada fuente de conocimiento tiene dos partes, una parte con la condición y otra con la acción. La parte de la condición está compuesta por un conjunto de hipótesis que satisfacen la condición -*stimulus frame*-, y una descripción de los cambios que la acción debe producir en la pizarra -*response frame*-.

Durante la ejecución la parte de la condición es buscada en la pizarra con el objetivo de encontrar hipótesis que son de interés para la acción. Tras la activación, la parte de la acción procesa todas las hipótesis recibidas. Las tareas de las fuentes de conocimiento abarcan desde la clasificación, por ejemplo, clasificar segmentos acústicos en clases fonéticas, al reconocimiento de palabras.

Control. El control reside en un monitor atento a los cambios en la pizarra -*blackboard monitor*-, y de un organizador que selecciona la actividad a ejecutar -*scheduler*-. Las actividades están compuestas por la parte de la condición y las fuentes de conocimiento a invocar. La zona de control está representada en la Figura. 2.1 mediante los cuatro bloques inferiores.

El componente de control ejecuta iterativamente los siguientes pasos, (i) el *scheduler* selecciona del *scheduling queue* una actividad para ser ejecutada, (ii) Si se selecciona y se ejecuta la condición satisfactoriamente, un conjunto de *stimulus-response frames*, junto a un puntero a la dirección de la fuente de conocimiento invocada, serán almacenados en la *scheduling queue*, (iii) Si la acción es seleccionada y ejecutada, la pizarra será modificada. El *blackboard monitor* enviaría al *scheduling queue* los punteros a las condiciones que podrían continuar ese cambio.

Obviamente esta es la arquitectura de la que se extrae el modelo arquitectónico radiografiado en el apartado anterior, por tanto la similitud es completa. A continuación se mostrará otro ejemplo clásico de este modelo. Aunque los componentes principales se identifican perfectamente, se observan adaptaciones debidas a la adecuación del modelo a la tarea, así como a las decisiones particulares de diseño y conexión de las partes.

2.1.1.2. HASP

El proyecto comenzó en 1972 y terminó en 1975 pero fue retomado en 1976 bajo el nombre de SIAP [Nii et al., 1982]. El propósito de HASP fue desarrollar y mantener un sistema de vigilancia oceánica. En una pantalla representaba la situación de las actividades de distintas plataformas, barcos y submarinos. Para ello utilizaba un flujo múltiple y continuo de señales acústicas. La entrada de datos era proporcionada por *arrays* de hidrófonos cada uno encargado de monitorizar un segmento del área de barrido. Además el sistema era capaz de integrar informes de inteligencia. Los informes eran otra entrada de datos y lo hacían a través de los niveles superiores. Los informes contenían información sobre los movimientos de los barcos y submarinos -enemigos o no-, así como de la actividad de barcos comerciales [Nii, 1986].

Blackboard. La estructura de la pizarra está dividida en una abstracción progresiva jerárquica en niveles, consistente en segmentos, en el nivel más bajo, y líneas, conjuntos de armónicos, fuentes acústicas, plataformas y flotas en el nivel más alto. La señal acústica llega al nivel del segmento y el informe de inteligencia a los niveles superiores, bien al nivel de flota o al de plataforma, según el contenido del informe.

Knowledge source. Cada fuente de conocimiento está formada por una parte que describe la precondición y por otra que describe la acción. La precondición consiste en una lista de pares de *tokens*: nombre del evento, modificador. La acción esta formada por un conjunto de reglas. Está organización de la fuente del conocimiento puede ser vista como un simple disparador que activa un conjunto de reglas.

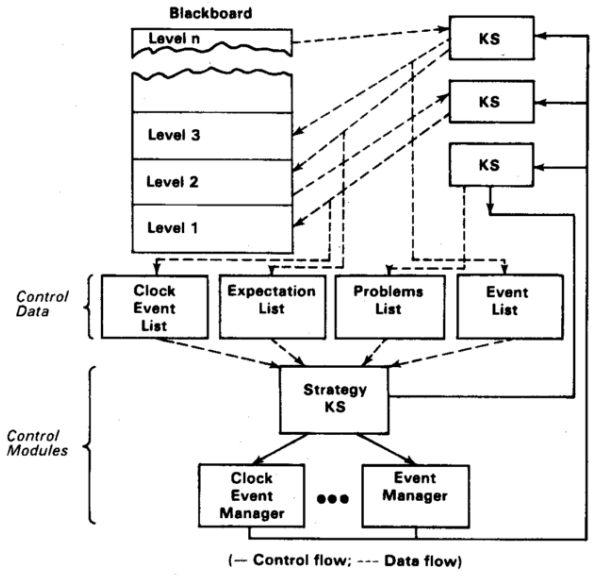


Figura 2.2: Organización del sistema HASP/SIAP [Nii, 1986].

Control. Los módulos de control están escritos de la misma forma que el dominio, como un conjunto de reglas. La iteración básica del componente de control consiste en: (i) Decide en que categoría de evento centrarse -*clock, expectations, problems or blackboard*-. (ii) De esa categoría elige un evento a procesar, que contiene el nombre y el tipo de modificación asociado. (iii) Las fuentes de conocimientos asociadas a ese tipo de modificación se ejecutan. (iv) La ejecución de las fuentes de conocimiento producen cambios en la pizarra y estos cambios se almacenan.

En resumen, en HASP había cuatro categorías de eventos. Cada categoría de eventos contenía un conjunto predeterminado de tipos de eventos (*control data* en Figura 2.2). Para cada tipo de evento, se predeterminaban las fuentes de conocimiento que podría procesar una instancia del tipo de evento (*control modules* en Figura 2.2).

Esta división manifiesta al nivel de arquitectura una decisión de diseño que adapta el modelo a las necesidades del proyecto. Además, el sistema era *open ended* ya que los nuevos tipos de eventos y las nuevas fuentes de conocimiento podrían añadirse sin perturbar a las existentes, gracias al alto grado de modularidad que presenta.

2.2. Arquitecturas para Robótica

Tradicionalmente las arquitecturas software en la Robótica eran de tipo jerárquico, debido a que estaban altamente influenciadas por la Inteligencia Artificial. Prevalecía que el sistema de control de un robot autónomo móvil debía ser descompuesto en tres elementos funcionales: el sistema de sensorización, un sistema de planificación y un sistema de ejecución. Este paradigma es conocido con el nombre de *sense-plan-act* (SPA). La función del sistema de sensorización es trasladar los datos en crudo provenientes del sensor a un modelo del mundo. En aquella época (la década de los 80), sonar, láser o visión eran la entrada de datos habitual. La función del planificador era tener en cuenta ese modelo del mundo y un objetivo y generar un plan que le permi-

tiese alcanzarlo. La función de la capa de ejecución era recibir el plan y secuenciar las acciones prescritas. El paradigma no funcionaba demasiado bien, en parte por la dificultad de modelar el mundo, ya que dependía demasiado de la tecnología de unos sensores que no eran adecuados o no estaban lo suficientemente desarrollados [Orebäck and Christensen, 2003](problemas que aún persisten), y en parte porque la planificación era un proceso que consumía mucho tiempo. El mundo podía haber cambiado durante ese periodo e invalidaba el resultado del plan.

A mitad de década empieza a quedar claro que el paradigma SPA no es suficiente. El modelado del mundo y la planificación presentan serios problemas. La ejecución de un plan en bucle abierto era claramente insuficiente frente a la incertidumbre e imprevisibilidad del entorno [Gat, 1998]. Incluso varios autores sugieren diferentes mecanismos de ejecución [Payton, 1986, Agre and Chapman, 1987].

En 1986 Rodney Brooks revoluciona el área presentando una arquitectura basada en comportamientos reactivos puros. Esta arquitectura, la famosa *Subsumption architecture*, se caracterizaba por tener un limitado o ningún conocimiento del mundo [Brooks, 1986]. Aunque pretende alejarse del paradigma SPA, E. Gat [Gat, 1998] nos recuerda que en el artículo original, la arquitectura de *Subsumption* es presentada como un intento de hacer SPA más eficiente, mediante la aplicación de restricciones dependientes de la tarea a los niveles de subsunción.

No obstante, el punto donde la arquitectura propuesta por Brooks se aleja definitivamente del paradigma tradicional es al descartar la necesidad del plan, e incluso de cualquier tipo de representación simbólica [Brooks et al., 1989]. La arquitectura *Subsumption* alcanzó un éxito sorprendente en el área de la navegación de Robots. Un robot llamado *Herbert* llegó a ser programado para encontrar y recoger latas de soda en una oficina [Connell, 1989].

Sin embargo el esquema puramente reactivo no funciona adecuadamente a la hora de realizar tareas complejas, ya que presenta el problema de la imposibilidad de cambiar la tarea para la que fueron programados, sin que ello implique reescribir el programa de control. Finalmente un enfoque híbrido acabó instaurándose como la elección más común entre los investigadores en Robótica.

De acuerdo a Gat [Gat, 1998] al menos tres grupos de investigadores -incluyendo el suyo- convergen en una solución similar para las arquitecturas de control a principios de los noventa. La primera es la Tesis del propio Gat [Gat, 1991] y las otras dos soluciones son: [Bonasso, 1991] y [Connell, 1992]. Las tres soluciones se componen, de un mecanismo de control reactivo realimentado, de un planificador deliberativo, y de un mecanismo de secuenciación que conecta ambos.

La clave de esta decisión de diseño común tiene su origen en el papel que juega el estado interno. En los procesos de alto nivel se generan estados intermedios que reflejan el estado del mundo. Ahí es donde reside el origen de las dificultades en el paradigma SPA, mostrándose claramente cuando el estado interno pierde sincronía con la realidad que intenta representar.

La solución reactiva de Brooks a este problema es minimizar, tanto como sea posible, el uso del estado interno. Si no hay estado interno, no es posible perder sincronía con la realidad. Desgraciadamente, extraer y abstraer información del mundo exterior parece necesario. Un obstáculo ocluido puede tenerse en consideración fácilmente si antes había sido percibido y almacenado en una representación interna.

Gat concluye que las arquitecturas en tres niveles organizan los algoritmos según su la cantidad de estado mantenido: i) sin estado en el nivel reactivo, ii) con estados que reflejan el presente y el pasado, en el nivel de ejecución, y iii) con estados que representan predicciones sobre el futuro en el nivel deliberativo.

2.2.1. Arquitecturas de tres niveles

La anatomía de este enfoque híbrido modelado en tres niveles dispone el nivel reactivo en su nivel más bajo. Su misión es adquirir los datos en crudo de los sensores y hacer cálculos sobre ellos. Esto debe hacerse en frecuencias cercanas al tiempo real para satisfacer las necesidades críticas de seguridad y reaccionar adecuadamente. En el nivel más alto se manejan la planificación, el razonamiento y la interacción con el humano. En el nivel intermedio, también llamado el nivel de secuenciación o supervisión, se pretende unir el hueco entre lo deliberativo y lo reactivo, coordinándolos.

El nivel reactivo está basado en comportamientos. Un comportamiento se define aquí como un segmento de código que produce un comportamiento físico, tal como la captura de datos del sensor o el movimiento de un actuador. El nivel está compuesto por comportamientos separados, donde cada comportamiento tiene una tarea específica, acotada y simple. Obviamente, el comportamiento está estrechamente ligado con el sensor y/o el actuador y suelen carecer de estado interno. Existen módulos que fusionan la información de varios sensores para extraer datos relevantes y se los proporcionan a los niveles superiores. Esto puede implicar que varios comportamientos estén activos al mismo tiempo. De la misma manera, varias órdenes de respuesta pueden ser enviadas a diferentes actuadores para responder adecuadamente.

La función del nivel intermedio consiste en seleccionar el comportamiento y los parámetros adecuados para ser ejecutado por el nivel reactivo. La dificultad reside en cómo seleccionar el comportamiento preciso. Este nivel de supervisión debe ofrecer un resultado condicionado a la situación actual y en un corto espacio de tiempo, ya que el período de tiempo debe estar en consonancia con la frecuencia con la que el entorno cambia.

En el nivel deliberativo tienen lugar los procesos que consumen mucho tiempo de proceso, como la planificación. La característica arquitectónica clave del nivel deliberativo es que pueden ejecutarse varios comportamientos, entre el instante en que un algoritmo deliberativo es invocado, y el instante en que produce un resultado. Los procesos en este nivel se comunican con el resto del sistema de dos formas, o bien produce planes para que sean ejecutados por el secuenciador -nivel intermedio-, o bien responde a consultas específicas realizadas por él.

En consecuencia, la taxonomía de las arquitecturas de tres niveles surge de la observación empírica de los algoritmos usados para el control de robots móviles. Los algoritmos son clasificados atendiendo a un factor clave: el tiempo de proceso.

En [Orebäck and Christensen, 2003] se puede encontrar un completo estudio, evaluación y análisis de las arquitecturas para Robótica más conocidas de la época, y que seguían este patrón de diseño. Algunos ejemplos son:

- **AuRA** [Arkin et al., 1987]. Formada por un sistema jerárquico que consistía en un planificador de misiones, un razonador espacial y un secuenciador del plan vinculado con un controlador que respondía de forma reactiva. Un administrador se encarga de supervisar y controlar el comportamiento durante la ejecución. Cada comportamiento está asociado con un esquema perceptual que proporciona el estímulo requerido por el comportamiento.
- **3T**[1] [Peter Bonasso et al., 1997]. Los principios para ubicar los módulos en las diferentes niveles son: tiempo, ancho de banda, necesidades y frecuencia de modificación de la misión. El nivel reactivo consiste en habilidades que se emparejan con los comportamientos. En el nivel intermedio, el secuenciador opera usando RAP [Firby, 1987], un lenguaje

[1] Su estructura es utilizada en [Gat, 1998] como ejemplo descriptivo

especifico de planificación reactiva y ejecución. El nivel deliberativo está a cargo de la planificación.

- **Berra** [Lindström et al., 2000]. Los sensores y los módulos de fusión de los mismos se denominan *resources*. Los *Controllers* representan tanto a los actuadores como a los módulos que fusionan diferentes actuadores. La capa intermedia se denomina *Task Execution Layer*.

Tras el estudio de las arquitecturas anteriores, los autores Orebäck y Christensen propusieron un conjunto de requisitos deseables para una arquitectura para Robótica. Subrayaron como una de las características más importantes la consecución de un alto grado de diseño modular. En su opinión, las tecnologías basadas en componentes presentaban una buena alternativa a seguir para conseguirlo -y atendiendo al desarrollo posterior de la Robótica sin duda acertaron. El concepto de interfaz surge como oposición a los múltiples niveles de herencia y podría ser una opción para ayudar a resolver los nuevos desafíos, tal y como sucedía en la programación orientada a objetos, pero avanzando en su autonomía y funcionalidad.

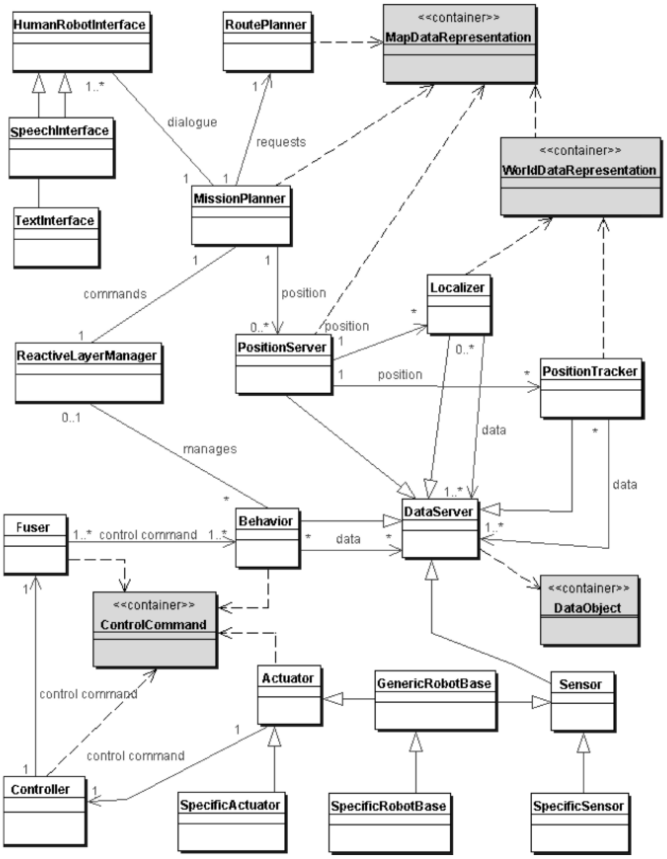


Figura 2.3: Propuesta de diseño genérico de una arquitectura Robótica [Orebäck and Christensen, 2003].

La Figura. 2.3 muestra un ejemplo de diseño genérico donde se puede apreciar que la parte del contexto que recibe cada módulo dependerá de las conexiones de entrada que tenga. El problema reside en que no podemos conocer de antemano que es lo que debemos compartir y cómo debemos hacerlo. Por supuesto esto se acentúa si lo que queremos diseñar es una arquitectura

para Robótica de propósito general, no limitada a una tarea específica y que aglutine diferentes habilidades y comportamientos. En esta Tesis defenderemos la idea de que la solución no consiste en elevar el número de conexiones cuando se necesite, sino en dotar a la arquitectura de la capacidad de que cada módulo pueda acceder a la cantidad de contexto que precise.

2.3. Arquitecturas cognitivas

Las arquitecturas cognitivas constituyen y han constituido una línea de investigación muy importante, tal vez debido a que persiguen un objetivo central de la inteligencia artificial y la ciencia cognitiva: la creación y la comprensión de agentes sintéticos que soporten las mismas capacidades que los seres humanos [Langley et al., 2009]. Existen líneas de investigación centradas en el modelado de los aspectos invariantes de la cognición humana, mientras que otros esfuerzos ven a las arquitecturas cognitivas como un camino eficaz para la construcción de agentes inteligentes.

Esta segunda dirección es la más cercana a la propuesta de esta Tesis y a una vertiente importante de la Robótica actual. Así, aquí se presenta un sistema que aspira a crear una inteligencia colectiva o a nivel de sistema, que se sustenta en el uso de diferentes agentes, combinados entre sí y con la arquitectura.

El objetivo de las arquitecturas cognitivas es tender a la amplitud de la cobertura a través de un conjunto diverso de tareas y dominios. Es decir, si lo comparamos con un sistema experto, entendido como un componente o método diseñado específica y completamente para resolver una tarea especializada, la arquitectura cognitiva trata de ofrecer un comportamiento inteligente a nivel general, que le permite combinar habilidades para resolver la tarea y enfrentarse a otras nuevas.

El término *cognitive architecture* fue introducido en la ciencia cognitiva por Allen Newell en 1990: *"the fixed (or slowly varying) structure that forms the framework for the immediate processes of cognitive performance and learning"*. Más tarde ha habido otras definiciones, por ejemplo, Anderson en 2007 lo definió como: *"a cognitive architecture is a specification of the brain at a level of abstraction that explains how it achieves the function of adaptive cognition"* [King et al., 2007].

La definición que nos parece más acertada es la aportada por Langley en 2009: *"A cognitive architecture specifies the underlying infrastructure for an intelligent system"* [Langley et al., 2009], debido a que sintetiza y aclara la idea de que la estructura articulará la inteligencia y permite preguntarse cómo debe ser o qué elementos debe incluir, a la vez que deja el horizonte abierto para poder incorporar a esa infraestructura subyacente cualquier elemento que la haga más inteligente. Dicha infraestructura debe incluir aquellos aspectos de un agente cognitivo que son constantes a lo largo del tiempo y en diferentes dominios. Normalmente [Langley et al., 2009]:

- La *memoria* a corto plazo y largo plazo que almacena las creencias del agente, los objetivos y el conocimiento.
- La *representación* de los elementos contenidos en esas memorias y su organización en estructuras mentales.
- Las *operaciones* y funcionalidades que operan y utilizan esas estructuras, así como los mecanismos de aprendizaje.

Hay una analogía directa con la arquitectura de un edificio que consta de características permanentes -o al menos su modificación es menos frecuente-, como sus cimientos, distribución de las habitaciones, paredes, techo, etc. mientras que sus muebles y electrodomésticos se redistribuyen en un mayor número de ocasiones. Esos muebles son los contenidos, conocimientos o creencias que pueden cambiar con el tiempo en la memoria del agente. No se consideran parte de la arquitectura pero sí deben existir los elementos que los soporten y los permitan.

Al igual que ocurre con los diseños diferentes de edificios construidos para la misma funcionalidad, las arquitecturas cognitivas difieren en su elección de representación y adquisición de conocimientos y creencias, así como en los mecanismos de utilización pero, en general, tratan de satisfacer los mismos objetivos globales.

A continuación se presentan algunos ejemplos de arquitecturas cognitivas que suelen aparecer en la literatura. Se ha seguido la revisión sobre las arquitecturas cognitivas descrita por Langley et. al. en [Langley et al., 2009] debido a que se discute la forma en que las arquitecturas representan, organizan, utilizan y adquieren el conocimiento. Este enfoque resulta especialmente interesante, desde un punto de vista conceptual, para la propuesta de *Deep State Representation* y de CORTEX ya que sugiere claves para su interpretación en el ámbito de la Robótica.

2.3.1. Ejemplos

Los siguientes ejemplos de arquitecturas cognitivas han sido seleccionados puesto que describen el espectro de decisiones tomadas en el diseño de las mismas, son recurrentes en la literatura y suelen aparecer como casos de estudio en la mayoría de revisiones.

2.3.1.1. ACT

ACT-R (Adaptive Control of Thought–Rational) [Anderson et al., 2004] es la última de una familia de arquitecturas cognitivas preocupadas principalmente por el modelado del comportamiento humano que ha experimentado un desarrollo continuo desde finales de 1970. ACT-R asume que el conocimiento humano puede ser dividido en declarativo y procedimental. El conocimiento declarativo es representado como pedazos o*chunks*. La arquitectura está organizada en un conjunto de módulos, cada uno de los cuales procesa un tipo diferente de información.

Estos incluyen módulos perceptivos y motores -*Perceptual-motor modules*, compuestos por un módulo sensorial para el procesamiento visual -*visual module*- y un módulo de motor para la acción -*motor module*-. La memoria se divide en dos, el modulo declarativo -*Declarative module*-, para el conocimiento declarativo, y el módulo procedimental -*Procedural memory*- , que es codificado como producciones, las cuales representan el conocimiento sobre cómo hacemos las cosas. Cada módulo tiene un *buffer* asociado que tiene una estructura declarativa relacional *chunk* y es la única forma de acceder al módulo. Tomados en conjunto, estos *buffers* comprenden la memoria a corto plazo de ACT-R. La Figura. 2.4 muestra como se organiza información en ACT-R.

El modulo *Procedural memory*, selecciona las reglas y coordina su procesamiento en los módulos. Las condiciones de cada producción están en los *chunks* y sus acciones modifican los *buffers* previa solicitud. Algunos cambios modifican las estructuras existentes, mientras que otros inician acciones en los módulos asociados, tales como la ejecución de un comando de motor o la recuperación de un *chunk* de la memoria declarativa. Cada *chunk* declarativo tiene un valor de activación asociado que refleja su uso en el pasado e influye en su recuperación de

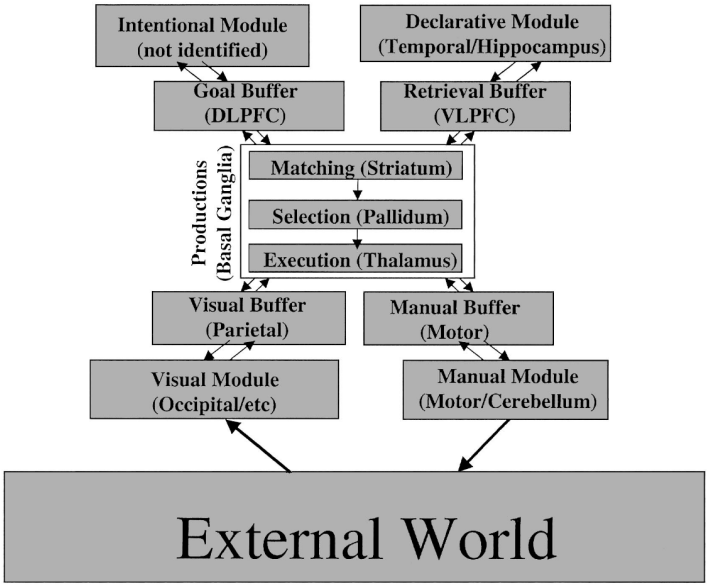


Figura 2.4: Organización de la información en ACT–R 5.0 [Anderson et al., 2004].

la memoria a largo plazo, mientras que cada producción tiene un costo esperado -en términos de tiempo necesario para alcanzar los objetivos- y su probabilidad de éxito.

En cada ciclo, ACT-R determina qué producciones coinciden con el contenido de la memoria a corto plazo. Este proceso de recuperación se ve influido por el valor de activación de los *chunks* que coincidan. ACT-R calcula la utilidad para cada producción asociada como la diferencia entre su beneficio esperado -la conveniencia de su objetivo por su probabilidad de éxito- y su coste esperado. El sistema selecciona la producción con la mayor utilidad -después de añadir ruido a esta puntación- y ejecuta sus acciones. La nueva situación conduce a nuevas producciones que se emparejan con el *chunk* correspondiente y son ejecutadas, por lo que el ciclo continúa.

El aprendizaje se produce en ACT-R tanto en lo estructural como en niveles estadísticos. Por ejemplo, el valor base de la activación para el *chunk* declarativo aumenta con el uso de producciones pero decae en caso contrario. Por otro lado el costo y la probabilidad de éxito para las producciones se actualiza basándose en su comportamiento observado. La arquitectura puede aprender nuevas reglas a partir de las soluciones de la muestra. Mediante un proceso de compilación de producciones que analiza las dependencias de múltiples activaciones de la reglas, reemplaza constantes con variables y las combina en nuevas condiciones y acciones.

La comunidad de ACT-R ha utilizado su arquitectura para modelar una variedad de fenómenos de la literatura de la psicología experimental, incluyendo aspectos de la memoria, atención, razonamiento, resolución de problemas y procesamiento del lenguaje.

2.3.1.2. Soar

Soar [Laird et al., 1987, Newell, 1994, Laird, 2008] es una arquitectura cognitiva que ha estado en constante desarrollo desde principios de los 80. El conocimiento procedimental a largo plazo se expresa en forma de reglas de producción que a su vez se organizan en términos de operadores asociados con el espacio del problema. Algunos operadores describen acciones sim-

ples, como primitivas que modifican el estado interno del agente o generan primitivas externas, mientras que otros describen actividades más abstractas.

Durante mucho tiempo Soar representó todo el conocimiento a largo plazo de esta forma pero, desde hace unos años, se ha añadido una memoria episódica y otra semántica independientes entre sí (Figura. 2.5). La memoria episódica [Nuxoll and Laird, 2007] mantiene un histórico de los estados previos, mientras que la semántica contiene hechos conocidos previamente. La memoria de trabajo a corto plazo -*working memory*- contiene un conjunto de elementos con atributos y valores.

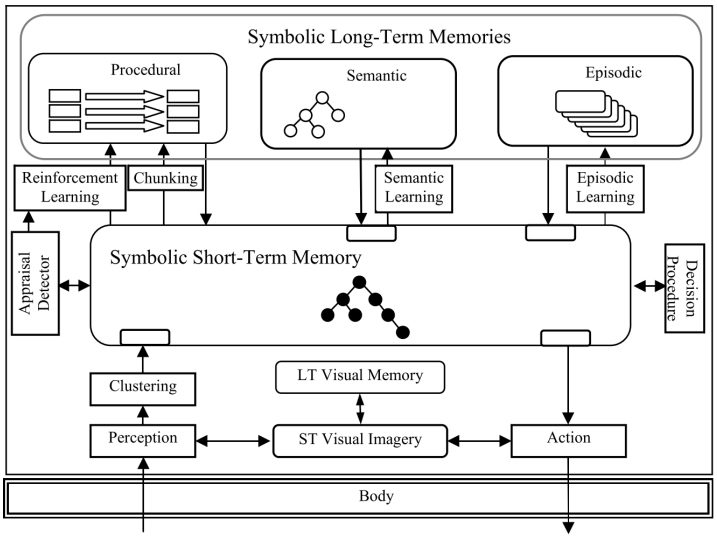


Figura 2.5: Estructura de Soar en su versión 9 [Laird, 2008].

Todas las tareas en Soar son formuladas como intentos para lograr el objetivo. Los operadores realizan los actos deliberativos básicos del sistema. Usan el conocimiento para determinar dinámicamente su selección y aplicación. El ciclo básico de proceso es proponer, seleccionar y aplicar operadores para resolver el problema avanzando, de decisión en decisión, hacia el objetivo.

El sistema actúa enfrentando las producciones contra los elementos de la memoria de trabajo generando subobjetivos automáticamente cuando no puede continuar. En Soar esto se conoce como punto muerto *impasse*. Para resolver el subobjetivo se puede acceder deliberadamente a la memoria episódica o semántica, y recuperar el conocimiento relevante para resolver el *impasse*. Cuando, tras procesar el subobjetivo, éste le ha permitido al agente superar ese *impasse*, la arquitectura agrega un nuevo *chunk* a la memoria a largo plazo. Un *chunk* es representado como una regla de producción que sintetiza el procesamiento del subobjetivo. Esto es conocido como *chunking* y es un mecanismo de refuerzo del aprendizaje en Soar. Así cuando el agente ha aprendido un *chunk*, este puede ser utilizado en una situación similar evitando el *impasse*.

Muchos investigadores han utilizado Soar para desarrollar una variedad de agentes sofisticados que han demostrado funcionalidades impresionantes. Tal vez una de la más conocida sea TAC-Air-Soar [Tambe et al., 1995], que modelaba pilotos de combate para ejercicios de entrenamiento en combates aéreos. También, basado en Soar, se han creado una serie de agentes inteligentes que controlaban personajes sintéticos en videojuegos interactivos [Magerko et al., 2004]. En [Lewis, 1993], Soar ha sido usada para modelar el procesamiento del lenguaje humano y en otras facetas de la cognición, aunque su uso estaba más orientado a

demostrar la funcionalidad a alto nivel que a una evaluación o medida cuantitativa de la calidad del procesamiento.

2.3.1.3. ICARUS

ICARUS es una arquitectura más reciente propuesta en 2004 [Langley and Messina, 2004, Langley and Choi, 2006]. Representa el conocimiento a largo plazo en memorias jerárquicas separadas, una para las habilidades y otra para los conceptos. En la memoria a corto plazo se almacenan creencias, objetivos e intenciones como instancias de la habilidad o el concepto.

ICARUS almacena el conocimiento de dos formas diferentes. Por un lado, los conceptos describen tipos de situaciones ambientales -*environmental situation*- en términos de otros conceptos y percepciones, mientras que por otro, las habilidades especifican como lograr los objetivos descomponiéndolos en subobjetivos ordenados. Ambos, conceptos y habilidades, involucran una relación entre objetos e imponen una organización jerárquica en una memoria a largo plazo. La primera basada en percepciones y la segunda en las acciones ejecutables. Además, las habilidades hacen referencia a *conceptos-objetivos* a alcanzar, así como a cuáles son sus condiciones de inicialización y de continuación.

ICARUS opera en un bucle de reconocimiento y acción. En cada paso, la arquitectura deposita descripciones de los objetos visibles en un *buffer* perceptivo. El sistema compara conceptos primitivos con esa percepción y añade instancias de esos conceptos coincidentes a la memoria a corto plazo, como creencias. Estas a su vez, disparan coincidencias en los conceptos del nivel superior, por lo que el proceso continúa hasta que ICARUS infiere todas las creencias deductivas implícitas. A continuación, a partir de un objetivo de nivel superior, se encuentra un camino descendiente a través de la jerarquía de habilidades, donde cada subtarea tiene que satisfacer las condiciones pero no satisfacer el objetivo. Cuando el recorrido termina en una tarea primitiva, la arquitectura aplica las habilidades del recorrido al entorno. Esto provoca nuevas percepciones, modificaciones en las creencias y ejecuciones reactivas de nuevos recorridos a través de la jerarquía de habilidades para satisfacer el objetivo del agente. La Figura. 2.6 muestra los procesos funcionales y sus conexiones con las memorias en la arquitectura ICARUS.

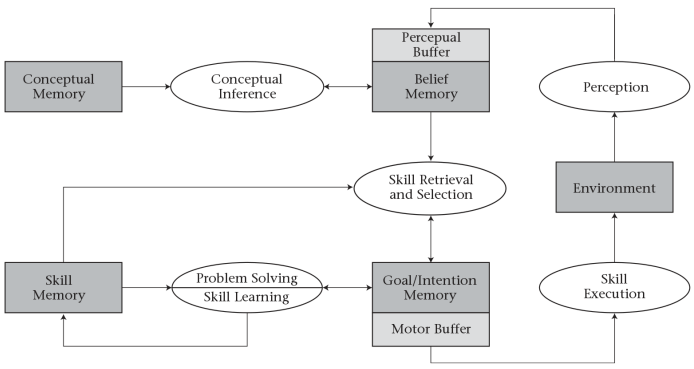


Figura 2.6: Procesos funcionales de Icarus y sus conexiones con las memorias[Langley, 2006].

Cuando no es posible encontrar un camino que resuelve el objetivo se recurre a un módulo que usa una variante del análisis medios-fines, *means-ends*, la cual retrocede en el recorrido jerárquico hasta encontrar una habilidad que permita lograr el objetivo actual. Si no es posible

encontrarla, el objetivo será desactivado de la definición del concepto. Siempre que la resolución de problemas consiga un objetivo, el aprendizaje de Icarus crea nuevas habilidades basadas en la traza que condujo al éxito. Además esa nueva habilidad será indexada para ese objetivo.

Los investigadores han utilizado Icarus para desarrollar agentes para un número de dominios que implican una combinación de inferencia, ejecución, resolución de problemas y aprendizaje. Estos han incluido tareas como las Torres de Hanoi, el juego *FreeCell Solitaire* y la planificación logística [Langley et al., 2009]. También han utilizado la arquitectura para controlar los personajes sintéticos en entornos virtuales simulados, incluyendo los que implican la conducción urbana y los videojuegos de acción en primera persona *First-Person Shooter* [Choi et al., 2007].

2.3.1.4. PRODIGY

PRODIGY [Carbonell et al., 1991] es una arquitectura cognitiva que tuvo un desarrollo intensivo desde la mitad de los ochenta hasta finales de los noventa. Incorpora dos tipos principales de conocimiento a largo plazo en estructuras separadas: *domain and control knowledge*. Los operadores o reglas de dominio describen los efectos de las acciones y se refieren tanto a las acciones físicas que afectarán al entorno como la inferencia de reglas puramente cognitivas. Las reglas de control especifican las condiciones en las cuales el sistema debe rechazar, seleccionar, o preferir un operador determinado.

Como en ICARUS, Prodigy cuenta con un módulo básico de resolución de problemas, que conduce la búsqueda de la solución a través del espacio del problema para lograr uno o más objetivos. Prodigy posee estructuras a corto plazo que incluyen descripciones de los estados actuales y utiliza una pila para almacenar los objetivos. La Figura. 2.7 muestra una visión general de la arquitectura Prodigy. Sitúa en el centro el módulo de resolución de problemas y en la parte superior los tipos de conocimiento.

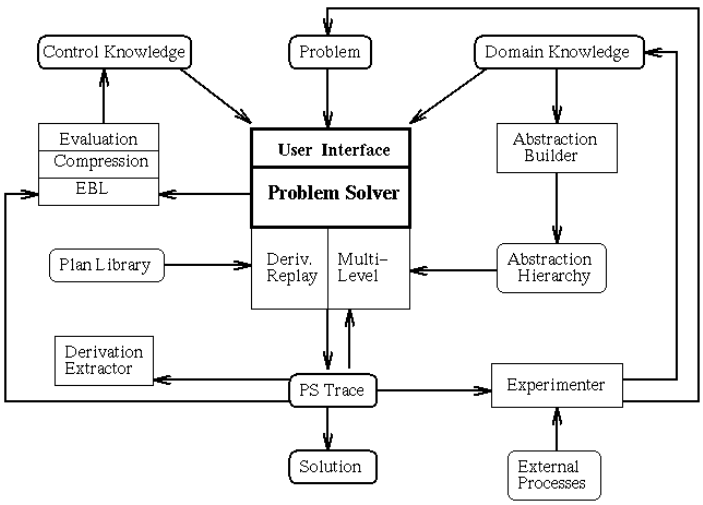


Figura 2.7: Representación de la arquitectura Prodigy[2].

En cada ciclo, Prodigy usa el módulo de resolución de problemas para seleccionar el operador que reduce la diferencia entre el objetivo y el estado actual, hasta encontrar una secuencia de operadores que alcanza el objetivo final. Un módulo de aprendizaje utiliza las

[2]http://cogarch.org/index.php/Prodigy/Architecture

trazas del módulo de resolución de problemas para modificar los criterios de selección de operadores. La arquitectura también recopila estadísticas sobre las reglas aprendidas, y almacena sólo aquellas que resuelven el problema más eficientemente. Incluye módulos independientes, pero con funcionalidades similares, que tratan de mejorar la calidad de la solución [Veloso and Carbonell, 1993, Wang, 1995, Perez and Carbonell, 1994].

Aunque la mayoría de las investigaciones relacionadas con esta arquitectura se han ocupado exclusivamente de la planificación y la resolución de problemas [Langley et al., 2009], cabe destacar que ha servido de base para un sistema que intercala la planificación y la ejecución en un robot móvil, y que aceptaba peticiones asíncronas de usuarios [Haigh and Veloso, 1998].

2.3.2. Funcionalidades y características

De las arquitecturas anteriores se puede extraer que una de las claves del diseño de la arquitectura cognitiva es cómo y dónde se almacena el conocimiento, al igual que cómo se permite a los distintos agentes acceder a él. Por ejemplo, el conocimiento del entorno procede de la percepción, el conocimiento de las implicaciones de la situación actual proviene de la planificación y el razonamiento, y el conocimiento sobre el pasado viene de los recuerdos y el aprendizaje.

Atendiendo a lo expuesto en [Langley et al., 2009], las capacidades o funcionalidades que una arquitectura cognitiva tiene o debería soportar podrían ser enumeradas de la siguiente forma:

- **Reconocimiento y categorización**. Un agente inteligente debe de tener algún contacto entre su entorno y su conocimiento.
- **Toma de decisiones y elección**. Para operar en el entorno el sistema inteligente requiere la habilidad de tomar decisiones y elegir entre distintas alternativas.
- **Percepción y evaluación de la situación**. La cognición no ocurre de forma aislada. Un agente inteligente existe en el contexto de algún entorno externo y debe sentirlo, percibirlo y evaluarlo.
- **Predicción y monitorización**. Las arquitecturas cognitivas deben existir en el tiempo, lo cual significa que pueden beneficiarse de la habilidad de predecir situaciones futuras y eventos.
- **Resolución de problemas y planificación**. Ya que un sistema inteligente debe lograr sus objetivos en situaciones nuevas, las arquitecturas cognitivas deben soportar la generación de planes y la resolución de problemas.
- **Razonamiento y mantenimiento de la creencia**. La resolución de problemas se relaciona estrechamente con el razonamiento. Otra actividad clave que permite al agente aumentar su conocimiento. El razonamiento extrae conclusiones mentales desde las creencias que mantiene el agente.
- **Ejecución y acción**. La cognición se produce con el fin último de apoyar y dirigir la actividad en el entorno. Para este fin, la arquitectura cognitiva debe ser capaz de representar y almacenar las habilidades motoras que permiten tales actividades.
- **Interacción y comunicación**. A veces, la manera más efectiva de obtener el conocimiento es gracias a otro agente. Por tanto, la comunicación es otra característica importante que debe poseer la arquitectura.

- **Memoria, reflexión y aprendizaje**. La arquitectura cognitiva debe beneficiarse de las capacidades horizontales que atraviesan las anteriores, operando sobre las estructuras mentales utilizadas y producidas en ellas.

Una vez descrito un posible conjunto de capacidades que describe la operatividad deseada para una arquitectura cognitiva, se puede observar que deben existir unas propiedades internas que den soporte al abanico de funciones expuesto anteriormente. Obviamente, una propiedad clave es la representación del conocimiento, pero también lo es la organización que impone ese conocimiento, o cómo son los mecanismos disponibles en el sistema para adquirirlo, utilizarlo y revisarlo mediante el aprendizaje.

El formalismo de representación en el que un agente codifica su conocimiento constituye un aspecto central de una arquitectura cognitiva [Langley et al., 2009]. Los enfoques tradicionales distinguen entre una representación declarativa y procedimental. En el enfoque declarativo el conocimiento puede ser manipulado por los mecanismos cognitivos sin importar su contenido. Por el contrario, el enfoque procedimental codifica el conocimiento para llevar a cabo una tarea.

La mayoría de las arquitecturas cognitivas se centran en la memoria semántica de conceptos genéricos y procedimientos. Tal vez porque se trata de un método natural para obtener el comportamiento generalizado que se le desea a un agente inteligente. Además, el uso de una memoria episódica parece muy adecuado para la recuperación de hechos y acontecimientos específicos.

Otra forma de representación es la red semántica [Borgida et al., 1991, Ali and Shapiro, 1993] que codifica tanto el conocimiento genérico como específico en un formato declarativo que consta de nodos -para los conceptos o entidades- y enlaces -para las relaciones entre ellos.

La manera de organizar el conocimiento en las arquitecturas cognitivas puede estar basada en la *relación* entre los elementos que la componen: así, puede que los elementos no hagan referencia explícita los unos a los otros, lo que no quiere decir que no puedan estar relacionados, o puede que haya una estructura jerárquica que los relaciona por definición.

También lo pueden hacer atendiendo a su *granularidad*. Si se almacenan formas de conocimiento expresadas, por ejemplo, en lógica de primer orden se denomina de grano fino *fine-grained*. Por el contrario, si se almacenan, por ejemplo, planes como macro operadores se conoce como de grano grueso *coarse-grained*.

Se puede observar otra forma de organización en el *número de memorias* distintas y cómo están relacionadas entre ellas. El agente requiere habilidades y conceptos de una memoria a largo plazo a la que accede menos frecuentemente. También requiere de otra memoria a corto plazo, más dinámica, donde almacenar sus creencias y objetivos actuales. En la mayoría de los casos se enfrentan una a otra como parte del ciclo de funcionamiento de la arquitectura.

El uso del conocimiento comprende un amplio espectro de situaciones que incluyen desde actividades de bajo nivel, como el reconocimiento y la acción, a procesos de alto nivel como el aprendizaje y la reflexión.

La adquisición del conocimiento suele comprender un proceso de aprendizaje basado en la experiencia. En general, en las arquitecturas existen métodos de aprendizaje que, o bien aprenden nuevas estructuras de conocimientos por completo, tales como reglas de producción o planes, o bien afinan las estructuras existentes mediante el uso de funciones numéricas.

Por tanto, se observa en el análisis que la triada inicial entre memoria, representación y funcionalidad queda patente en las distintas características y propiedades de las arquitecturas,

aunque se debe distinguir entre la teoría y la implementación. Por ejemplo, ACT-R, la cual ha sido implementada en varios lenguajes, en general no refleja exactamente la teoría, aspecto que desde nuestro punto de vista no es un problema, primero por las limitaciones evidentes de la tecnología actual y segundo porque son estos marcos teóricos los que inspiran el desarrollo tecnológico que los harán posibles. Las distintas implementaciones son aproximaciones parciales dentro de la ambiciosa teoría cognitiva propuesta en ACT-R.

Por otra parte, las arquitecturas cognitivas están en constante evolución y sus motivaciones suelen estar orientadas al alto nivel. Por desgracia esto provoca que los investigadores se centren menos en el proceso de extracción de la información del mundo exterior por el agente, y en su incorporación a la arquitectura. En este proceso de abstracción se suele usar vectores de características. Tampoco se suelen especificar cuales son las pautas de integración, o si la extracción de características es la misma para cualquier perceptor, o que ocurre cuando existe incertidumbre en la percepción. Tampoco se suele tener en cuenta la posición en el espacio de lo percibido, o si se puede extraer algún tipo de información de esa relación espacial. Esta fase se deja al margen y se trabaja con una abstracción expresada en el lenguaje común de la arquitectura, razones por las que se debe estudiar y decidir qué ideas pueden ser trasladadas al ámbito de la Robótica actual. En la siguiente sección se mostrará la Robótica cognitiva, cuyo objetivo es combinar la Robótica convencional con la Ciencia Cognitiva.

2.4. Robótica cognitiva

En los últimos años ha habido un aumento en el interés del uso de arquitecturas cognitivas para controlar plataformas Robóticas [Benjamin et al., 2004, Choi et al., 2009]. Los avances en las capacidades básicas de los robots contribuyen a esta tendencia tratando de satisfacer la demanda de los mecanismos de control de alto nivel. Como se ha visto, la tradición de las arquitecturas cognitivas hacia la búsqueda de una inteligencia general, las dota de un gran potencial para coordinar las habilidades de los robots en tareas específicas de manera que permitan la resolución de otras más complejas.

La Robótica cognitiva se podía definir como propone Levesque [Levesque and Lakemeyer, 2008][3]:

> "we take *cognitive robotics* to be the study of the knowledge representation and reasoning problems faced by an autonomous robot (or agent) in a dynamic and incompletely known world."

En el centro de ese esfuerzo está la comprensión de la relación entre el conocimiento, la percepción y la acción, y cómo debe ser desarrollado y adaptado al robot. Surgen algunas cuestiones difíciles de responder [Levesque and Lakemeyer, 2008], como: ¿cuánto debe conocer el robot de antemano y cuánto conocimiento debe adquirir mientras ejecuta la tarea? o ¿cuánto necesita saber el robot de su entorno y cuánto es necesario que sea conocido por el desarrollador? o ¿cuándo una acción debe ser razonada por el robot y cuándo esa acción puede ser ejecutada como una acción primitiva?

Por ejemplo, para un humano puede ser obvio que un objeto está encima de otro -una taza sobre una mesa- pero para un robot puede ser un reto importante determinar esa situación. Ello implica reconocer los objetos, calcular distancias y decidir relaciones. Por otro lado, sería

[3] De acuerdo con Levesque el termino fue acuñado por Reiter en 1993

también posible insuflarle ese conocimiento parcial del entorno y hacer que el robot asuma que los objetos estarán sobre la mesa.

Por tanto, el agente debe ser capaz de generar acciones en el mundo que sean apropiadas a su capacidad y sean acordes al conocimiento disponible en ese momento. La clave hacia una Robótica cognitiva es centrarse en ese mundo cambiante o dinámico que debe ser representado en un lenguaje lo suficientemente fluido. Debe disponer de métodos, como predicados o funciones simbólicas, que permitan modificar sus valores, acorde a los cambios percibidos del entorno, para actuar en consecuencia [Levesque and Lakemeyer, 2008].

En definitiva, el problema radica en cuál es la representación del conocimiento más adecuada y, en base a esa representación, cuál es la forma de razonamiento más apropiada para guiar y controlar el comportamiento del robot.

De acuerdo de nuevo con Levesque y en base a lo estudiado en la sección 2.3, donde se describió y analizó la representación del conocimiento en las arquitecturas cognitivas, se puede sintetizar lo siguiente:

- La representación debe incluir o servir para modelar la acción e incluirá un conjunto más o menos amplio de ellas.
- La representación debe trasladar los cambios producidos en el mundo exterior a través de los sensores y actuadores.
- La representación debe permitir al robot diferenciar qué es lo que sabe de lo que no.

Del lado del proceso de razonamiento hay dos aspectos que en los robot cognitivos juegan un papel muy importante. La posibilidad o *legality task* que determina si una acción o secuencia de acciones se puede realizar a partir de un estado inicial -gramática-, y la proyección o *projection task* que determina si la condición se mantendrá, como cierta o falsa, tras una secuencia de acciones realizadas a partir de un estado inicial -planificar.

2.4.1. Uso de arquitecturas cognitivas en robots

En los siguientes apartados se muestran diferentes robots que utilizan alguna de las arquitecturas cognitivas anteriores. La adaptación de la arquitectura al campo de la Robótica no es directa. Las arquitecturas pretenden abarcar una extensa área de la cognición humana para la que la Robótica actual no está preparada. El uso habitual consiste en que la arquitectura controle el alto nivel del robot y con ello su comportamiento. El robot suministra información del entorno y la arquitectura le devuelve la habilidad que debe usar.

2.4.1.1. ICARUS y el robot MAHRU

En [Choi et al., 2009] se usa la arquitectura cognitiva ICARUS para controlar al robot humanoide, MAHRU. ICARUS proporciona mecanismos para inferir el estado del mundo, tomar decisiones basadas en él y aplicar las acciones necesarias. Aunque ICARUS ha estado centrada en las capacidades del alto nivel, siempre ha existido el deseo y la necesidad de abarcar un dominio que envolviese algún tipo de maquinaría física. En [Choi et al., 2009] se muestra una aplicación de ICARUS en un entorno de robótica simulado. El entorno de simulación es una representación computacional realista del robot humanoide MAHRU. El ejemplo demostrativo

consiste en un escenario en el mundo de los bloques. El robot debe ordenar bloques de diferentes colores, apilarlos y construir torres del mismo color. En un primer paso hacia una integración completa, el simulador y la arquitectura se integran a través de una interfaz común.

Por tanto, ICARUS controla el alto nivel y dirige la versión simulada del robot MAHRU. MAHRU ejecuta las órdenes recibidas para lograr su objetivo. El conjunto de tareas parte de una condición inicial: encima de la mesa hay siete bloques diferentes, dos rojos, tres verdes y dos azules. La tarea más compleja consiste en construir torres del mismo color partiendo de esa condición inicial. Dicha tarea está representada en la Figura. 2.8 que describe la condición inicial y la secuencia de pasos que la completan.

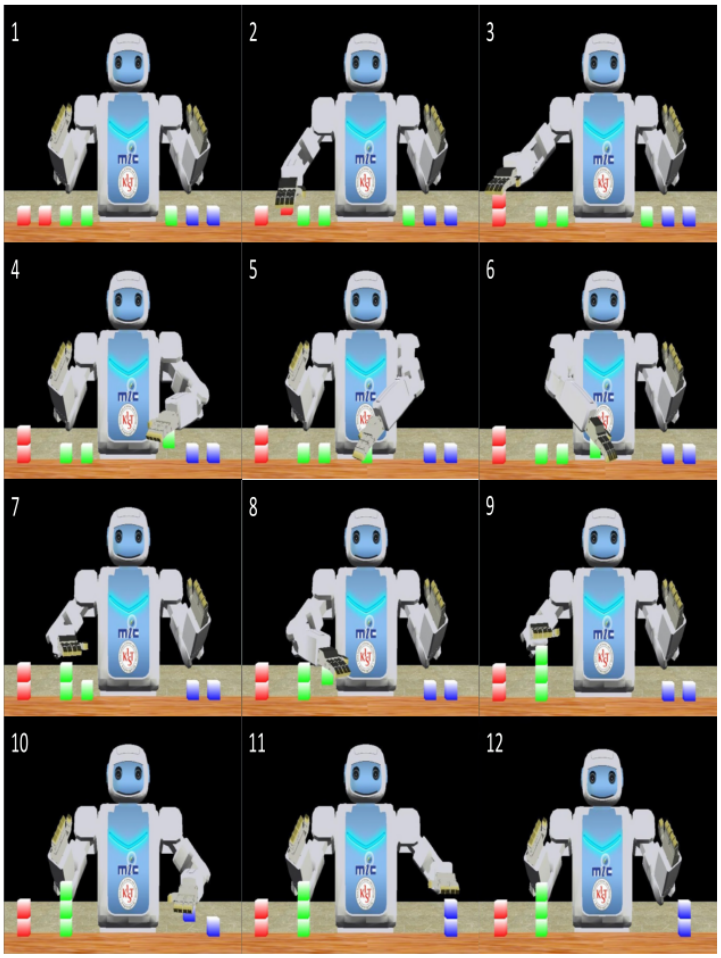


Figura 2.8: Secuencia de imágenes de MAHRU realizando la clasificación por color de los bloques[Choi et al., 2009].

Alterando la probabilidad de éxito de la habilidad se puede conseguir que ICARUS cometa errores inesperados en la simulación y acercarlo con ello un poco más a la realidad. En tales casos el sistema reacciona ante la nueva situación, intentando arreglar o ejecutando una habilidad diferente apropiada para ese nuevo estado del mundo. Por tanto, ICARUS demuestra que es capaz de recuperarse ante fallos como no coger un bloque correcto o tirarlo [Choi et al., 2009]. Sin embargo, poco se especifica acerca de cómo sería el proceso físico que permitiese al robot percibir el error. O cómo sería detectada la posición de los bloques o su forma. Y si esto influiría o no a la hora de realizar la acción de manipulación del bloque o incluso la tarea completa. De

la misma forma, se da por supuesto que los bloques son siempre bien percibidos, sin oclusiones ni problemas derivados.

Este comentario no es tanto una crítica a este trabajo en el que se pretende mostrar como en ICARUS el control de alto nivel gestiona con éxito la tarea, incluso ante situaciones de error, sino para advertir que aún se está lejos de una verdadera integración en una arquitectura para Robótica cognitiva completa. Aún así, el artículo es un paso importante hacia esa integración.

2.4.1.2. ACT-R/E y MDS

ACT-R/E [Trafton et al., 2009, Trafton et al., 2013] ha modificado ACT-R haciendo que perciba el mundo físico a través de sensores y efectores usados en robótica (Figura. 2.9). Para ello añade una serie de módulos que facilitan la cognición corpórea, *embodied-cognition*, de ahí su nombre *ACT-R/Embodied*. En particular, ACT-R/E añade un módulo espacial y modifica el visual y el motor para trabajar con el robot y usar sensores reales. El módulo visual, que normalmente recibe la información de un monitor de ordenador, ha sido modificado para aceptar como entrada una cámara de vídeo. El módulo motor original permite los movimientos de una mano virtual (teclear o usar el ratón). En ACT-R/E está permitido enviar órdenes de navegación y movilidad. El módulo también permite la autolocalización del robot en su entorno. Para completar el control motor al ámbito de operación, éste controla los movimientos de la cabeza del robot, incluidos los relacionados con los movimientos horizontales y verticales de los ojos.

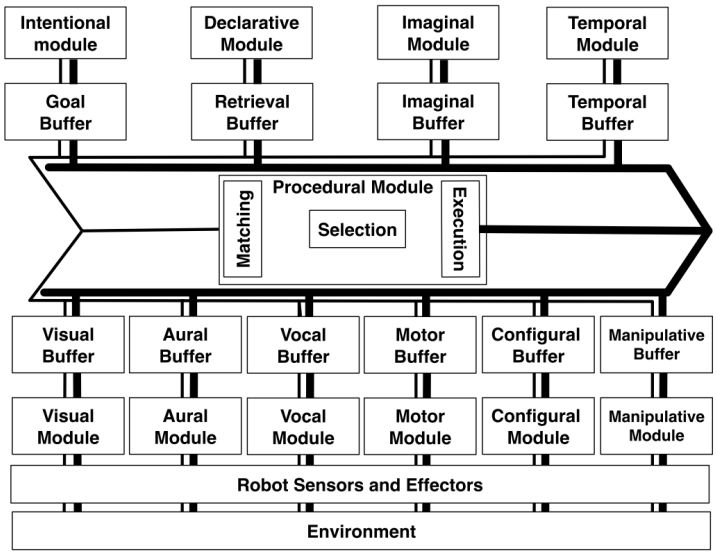


Figura 2.9: Diagrama arquitectónico de ACT-R/E [Trafton et al., 2013].

El módulo espacial utiliza la teoría conocida como *Specialized Egocentrically Coordinated Spaces* SECS [Harrison et al., 2003]. SECS proporciona dos módulos egocéntricos espaciales que son responsables de codificar y transformar la representación del espacio para satisfacer las necesidades de la navegación y la manipulación. Por ejemplo, el módulo espacial permite que el agente represente un mapa cognitivo del espacio a pequeña escala. El módulo de manipulación permite representar información relevante sobre los objetos, como su orientación y posición. También sirve para coordinar el movimiento de la cabeza y los ojos.

Todo ello responde a las necesidades de la *embodied-cognition* que añade la restricción impuesta por el cuerpo a la cognición. Es decir, que ésta ocurre dentro de un cuerpo físico, y éste debe ser tenido en cuenta en el proceso cognitivo. Según el cuerpo, sus características y el espacio donde se encuentre, la navegación, la manipulación o la percepción, serán de una u otra forma. Por tanto, existe una vinculación entre el lugar, el sujeto y el proceso cognitivo.

La plataforma Robótica usada es *Mobile-Dexterous-Social* MDS[4] El robot MDS tiene 18 *DoF* para el cuello y la cabeza incluyendo el movimiento de los ojos. El robot es capaz de dirigir la mirada a varios sitios del espacio tridimensional. La percepción visual incluye una fuente de vídeo en color y una de profundidad.

En [Trafton et al., 2009] el objetivo que se plantean los autores es presentar un *embodied model* para el seguimiento de la mirada -*gaze-following*-. El modelo aprende cómo seguir la mirada del otro mediante el uso de mecanismos cognitivos. Este modelo es un componente muy importante en el camino hacia la atención visual compartida, que es aquella que se produce cuando dos personas miran un mismo objeto. Varios investigadores [Baron-Cohen, 1995] han sugerido que está fuertemente relacionada con la habilidad humana de inferir los estados mentales del otro, lo que la convierte en una habilidad muy interesante para la interacción humano-robot. El modelo cognitivo propuesto en [Trafton et al., 2009] iguala los resultados del clásico [Corkum and Moore, 1998] y además es capaz de conseguirlo funcionando a bordo del robot MDS.

Lo interesante desde nuestro punto de vista es la conclusión de la necesidad de la representación espacial para lograrlo. En palabras de los autores: *"The model has a psychologically plausible representation of space that is critical to the success of the model"* [Trafton et al., 2009]. En este caso la tarea ha sido aprender a seguir la mirada del otro. Pero existen muchas tareas en la interacción con objetos, o en la interacción con humanos, o la misma atención visual compartida, que necesitan de la representación espacial.

2.5. Agentes

El concepto de agente en Ciencias de la Computación e Inteligencia Artificial es bastante amplio y sus variados significados cubren la mayor parte de lo que una aplicación informática puede hacer. Franklin [Franklin et al., 1997], después de una revisión exhaustiva de las muchas propuestas sintetiza un agente autónomo como:

> "a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."

La arquitectura determina los mecanismos que utiliza un agente para reaccionar a los estímulos, actuar o comunicarse [Corchado and López, 2002]. Las arquitecturas de agentes se definen como un conjunto de agentes interconectados para conseguir una funcionalidad deseada. Aplicando aquí los conceptos desarrollados en las secciones anteriores, también en las arquitecturas de agentes podemos diferenciar tres tipos: deliberativas, reactivas e híbridas, estas últimas definidas como una combinación de aspectos de las anteriores.

[4] http://robotic.media.mit.edu/portfolio/mobile-dexterous-social-robots/

2.5.1. Arquitecturas de agentes

Deliberativas

Son aquellas arquitecturas que utilizan modelos de representación simbólica del conocimiento. Suelen estar basadas en la teoría clásica de planificación. Estos agentes parten de un estado inicial y son capaces de generar planes para alcanzar sus objetivos [Maes, 1991].

Por tanto, un agente deliberativo -o con una arquitectura deliberativa- es aquel que contiene un modelo simbólico del mundo y lo usa para tomar decisiones utilizando mecanismos de razonamiento lógico, basados en la correspondencia de patrones y la manipulación simbólica, con el propósito de alcanzar los objetivos del agente. Estos agentes están dotados de modelos de planificación capaces de generar planes a partir de las creencias e intenciones [Jennings, 1993].

Un ejemplo relevante de este tipo es la arquitectura deliberativa BDI *Belief, Desire, Intention* [Rao and Georgeff, 1995] que está caracterizada por el hecho de que los agentes que la implementan están dotados de los estados mentales correspondientes a Creencias, Deseos e Intenciones. Fue desarrollado para proporcionar soluciones en entornos dinámicos con incertidumbre, ya que las aplicaciones en el mundo real se deben manejar en este entorno cambiante e incierto. El software convencional está *orientado a la tarea* en lugar de *al objetivo*, de forma que cada tarea -o rutina- se ejecuta sin ningún recuerdo de por qué ha comenzado su ejecución. Sin el pasado no se puede aprender, ni recuperarse ante fallos [Corchado and López, 2002]. Este es el cometido de las creencias -*Belief*-.

El objetivo está expresado en los deseos -*Desire*- y representa el estado final deseado. La semántica es una lógica de deseos que le permiten alcanzar el objetivo, es decir, a partir de las creencias existentes es necesario definir un mecanismo de planificación que nos permita identificar las intenciones.

El sistema necesita comprometerse con los planes y subobjetivos, pero también debe ser capaz de reconsiderarlos. Los planes vinculados a la consecución de un objetivo constituyen las intenciones -*Intention*- del agente. Las intenciones son un conjunto de caminos de ejecución que pueden ser interrumpidos de una forma apropiada al recibir información acerca de cambios en el entorno [Rao and Georgeff, 1995]. La arquitectura presentada por Rao y Georgeff es una abstracción de los conceptos teóricos. La Figura. 2.10 muestra una representación de este modelo.

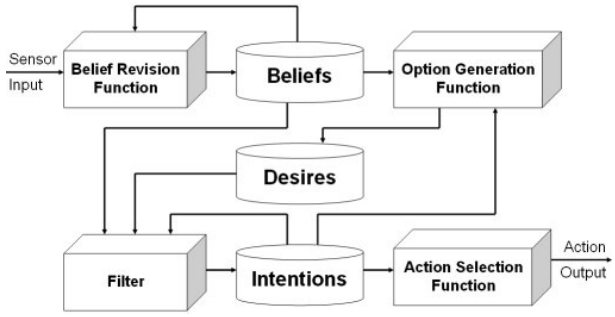


Figura 2.10: Representación de la arquitectura *BDI4jade*.[5]

El éxito de esta arquitectura se debe posiblemente a que combina elementos interesantes: un modelo filosófico del razonamiento humano fácil de comprender, un número considerable

[5] http://www.inf.ufrgs.br/prosoft/bdi4jade/?page_id=46

de implementaciones y una semántica lógica, abstracta y elegante [Corchado and López, 2002].

Reactivas

Los problemas que lleva asociado utilizar una representación simbólica del conocimiento han conducido al estudio de modelos más efectivos de representación del mismo [Peter Bonasso et al., 1997]. Un problema fundamental era el tiempo de respuesta. Las arquitecturas reactivas surgieron para evitar o minimizar cualquier tipo de representación y se caracterizan por no tener como elemento central de razonamiento un modelo simbólico.

Su máximo exponente es la propuesta de Rodney Brooks, conocida como arquitectura de subsunción o *subsumption* [Brooks, 1991]. Esta arquitectura se basa en las hipótesis de que la inteligencia es una propiedad emergente de ciertos sistemas complejos y de que ello permite generar comportamientos inteligentes sin necesidad de construir un modelo simbólico. Como mencionamos en 2.2 su aplicación ha sido crucial en el campo de la Robótica. El robot completo se considera un agente -agente físico-, que en conjunto exhibe este comportamiento reactivo. La arquitectura de un agente en Robótica vendría definida por la arquitectura de control del robot [Corchado and López, 2002]. Las arquitecturas de subsunción manejan jerarquías de tareas que definen un comportamiento. Suelen estar organizadas jerárquicamente en niveles, de menor a mayor nivel de abstracción. La Figura. 2.11 muestra una representación de esta arquitectura.

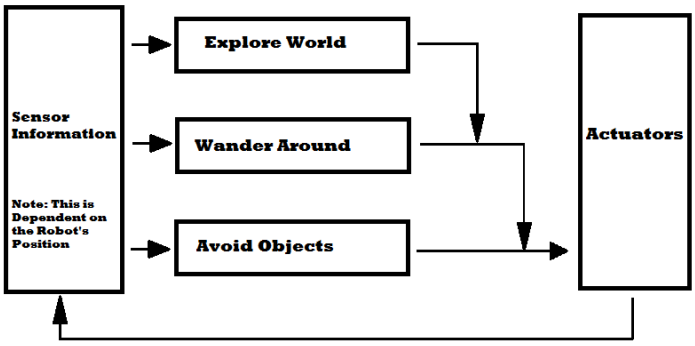


Figura 2.11: En la *subsumption architecture* los niveles superiores subsumen a los niveles inferiores de acuerdo a la información sensorial [Brooks, 1999].

El problema aparece al considerar problemas complejos donde el espacio de búsqueda de la solución es tan amplio que resulta, o al menos parece muy difícil, alcanzar la solución mediante la ejecución, sin un plan previo, de distintas acciones básicas [Corchado and López, 2002].

Híbridas

Como hemos visto, tanto las arquitecturas reactivas como las deliberativas presentan ciertas limitaciones. Por ello, se han propuesto sistemas híbridos que pretenden combinar aspectos de ambos paradigmas. Las características perseguidas del agente híbrido son las de un sistema que razona e interactúa con su entorno en tiempo real.

Una primera propuesta puede ser construir un agente compuesto de dos subsistemas: uno deliberativo, que utilice un modelo simbólico y que genere planes, y otro reactivo centrado en

reaccionar a los eventos del entorno y que no requiera un mecanismo de razonamiento complejo. Dicho agente dispondría de dos niveles: (i) El físico, que efectúa los trabajos relacionados con la percepción y la acción, tales como interpretar, filtrar y reaccionar al ambiente dinámico en el cual el agente está embebido; (ii) El cognitivo, que recibe información del nivel físico, construye un modelo evolutivo del mismo, e interpreta, razona y planifica a partir de él. Pero del dato en crudo al símbolo y del símbolo al actuador, la transición no suele resultar tan sencilla. Es en ese nivel intermedio donde, mayoritariamente, se han centrado los esfuerzos para construir una arquitectura híbrida que combinase lo mejor de ambos mundos.

2.6. ¿Por que CORTEX está basado en agentes?

En primer lugar y tras el estudio de las diferentes alternativas del concepto de agente, proponemos nuestra definición, similar a la sintetizada por [Franklin et al., 1997] que vimos al principio de la sección, pero con matices que nos concedan más libertad y nos permitan abarcar un espectro más amplio:

> "Un agente en CORTEX es una entidad computacional a cargo de una funcionalidad bien definida, ya sea reactiva, deliberativa o híbrida, que interactúa con otros agentes dentro de un contexto bien definido dando lugar a un sistema más complejo."

Los agentes en CORTEX son entidades conceptuales que se implementan con uno o varios componentes software de un *framework* de desarrollo, que en nuestro caso es RoboComp. Cuando varios de estos agentes están de alguna manera interconectados para lograr una función de nivel superior, se les conoce como arquitecturas basadas en agentes. Estos sistemas, también llamados multiagentes en Inteligencia Articial, permiten la gestión inteligente de un sistema complejo, coordinando los distintos subsistemas que lo componen e integrando los objetivos particulares de cada uno de ellos en un objetivo común.

La elección de una arquitectura basada en agentes para esta Tesis responde a la simplicidad computacional y la elegancia. Los agentes son unidades funcionales que pueden ser combinadas fácilmente para formar una estructura determinada. Pueden definirse de forma recursiva a partir de otros agentes simples, y siempre hay una conexión sencilla con los componentes software subyacentes. Esta es la **primera razón** por la que CORTEX es una arquitectura basada en agentes.

En CORTEX los agentes definen las funcionalidades clásicas de la Robótica, como la navegación, la manipulación, la percepción de la persona, la percepción de objetos, la conversación, el razonamiento, la planificación, el aprendizaje simbólico o la ejecución. Para satisfacer estas necesidades queremos que los agentes sean a la vez autónomos y obedientes. *Autónomos* para que proporcionen un comportamiento oportunista ante eventos no planeados y *obedientes* para que cuando los nuevos objetivos lleguen al sistema, todos los agentes trabajen orientados a la tarea: lograr el subobjetivo actual. Esta flexibilidad en la organización interna de los agentes, como bloques de construcción interoperables en CORTEX, es la **segunda razón** de la elección de una arquitectura basada en agentes.

La comunicación entre los agentes define la parte estructural de las arquitecturas. En las arquitecturas para Robótica cognitivas, más que la búsqueda de cómo es la cognición humana, lo que se explora es el espacio de diseño de la *embodied intelligence* [Sloman et al., 2006b]. Se

buscan arquitecturas que den lugar a la inteligencia, más que la arquitectura de nuestra inteligencia. En este sentido es posible y deseable jugar con todas las posibilidades de diseño para determinar qué arquitecturas serán las que den lugar a comportamientos inteligentes. Uno de los parámetros de diseño más importantes es el que define los flujos principales de información en la arquitectura. El flujo deliberativo es aquél en el que los agentes perceptivos deben proporcionar una descripción simbólica del robot y del mundo a los agentes deliberativos, para que éstos puedan razonar sobre hechos y planificar una secuencia de acciones para lograr el objetivo. Estas acciones son enviadas de vuelta a los agentes no deliberativos como objetivos locales. El flujo reactivo es aquél en el que los agentes no deliberativos reaccionan a cambios detectables en el mundo modificando la representación que mantienen de él. Esta reacción puede ser individual o mediante la interacción entre varios agentes, sin que intervengan elementos deliberativos. Ambos flujos deben coexistir e interaccionar en la arquitectura.

Un requisito importante para una arquitectura cognitiva para Robótica es facilitar esta coexistencia e interacción entre agentes deliberativos y reactivos. La distribución interna en agentes de la arquitectura facilita que estos flujos puedan cambiar, bien mediante la reprogramación manual, o bien mediante el aprendizaje automático. Así, la forma en que los agentes de la arquitectura se interconectan debe facilitar estos cambios y adaptaciones. Está es la **tercera razón** de porqué CORTEX es una arquitectura basada en agentes.

Esta Tesis trata de mostrar que los diseños basados en agentes, cuando se completan con los mecanismos de intercambio de información adecuados, son arquitecturas válidas para explorar este espacio de diseño de la inteligencia corpórea o *embodied*.

2.7. Conclusiones

Como se ha mostrado a largo del capítulo son varios los pilares sobre los que se asienta el diseño y desarrollo de CORTEX y de DSR. Los matices particulares y la adaptación de los mismos son la seña de identidad de la propuesta de esta Tesis.

La idea de una estructura compartida en el modelo de pizarra tiene el objetivo de resolver un problema dentro de su espacio de soluciones. Todos las fuentes de conocimiento trabajan independientemente contra un determinado nivel de información de la pizarra, para alcanzar la solución. Este proceso incremental hacia la solución consigue que emerja dentro la estructura de datos. En nuestro concepto de DSR, no solo emergerá la solución al problema, sino que también guiará a las fuentes de conocimientos -los agentes- y almacenará información ajena al espacio del problema -relación espacial, datos intermedios- pero de gran utilidad para el objetivo común de la misión. De la misma forma, la jerarquía de trabajo, la división del conocimiento en niveles sobre los cuales trabaja cada fuente de conocimiento para construir la solución, no es aplicable en nuestra propuesta.

De forma general, se podría decir que en los sistemas basados en pizarra originales, los agentes son concebidos principalmente como solucionadores de problemas, expertos heterogéneos que contribuyen al problema general de una manera híbrida entre la planificación y el oportunismo. En CORTEX, los agentes no solamente solucionan tareas deliberativas, sino también, perceptivas, motoras y de comportamiento. Por lo tanto la estructura de la pizarra será distinta y los agentes también la usarán de forma distinta.

En relación a las arquitecturas cognitivas se observa que la triada memoria, representación y funcionalidad queda patente en sus distintas características y propiedades. Si bien todas están

relacionadas entre sí, quedan muchas cuestiones abiertas que suscitan preguntas interesantes, por ejemplo ¿hasta qué punto la memoria a corto plazo que representa la creencia del sistema no es, en sí misma, una simple representación de su estado?, o, ¿por qué no usar la predicción en una memoria a corto plazo que permitiese validar un operador o una regla, pero sin la necesidad de que acabe formando parte del conocimiento a largo plazo, o tenga que ser ejecutada en la realidad?

Para nuestra propuesta, uno de los puntos más relevantes extraídos de las arquitecturas cognitivas es la necesidad de esa representación del conocimiento para poder alcanzar un verdadero estado cognitivo. Aunque sólo unas pocas habilidades, como el reconocimiento de la situación y el proceso de decisión, serían requisitos estrictos para alcanzar una arquitectura cognitiva suficiente, parece que el conjunto de las actividades humanas sugiere un conjunto más amplio de ellas.

Nuestra propuesta de representación trata de dar los primeros pasos hacia ese ambicioso objetivo con un diseño e implementación que aspira en el futuro a soportar las capacidades que se esperan de una arquitectura cognitiva, al menos desde el punto de vista de la Robótica, buscando una estructura que permita al robot representar y razonar sobre su estado y el entorno que le rodea.

Además, CORTEX, pretende ir un paso más allá y usar la representación no sólo para almacenar el conocimiento relativo al alto nivel, sino también el relativo a un conocimiento situacional o espacial del entorno. Para ello, la propia estructura DSR almacena la representación simbólica y la representación espacial. Parte del conocimiento tiene una representación material, algunos conceptos tienen una representación física y existe una relación geométrica entre ellos, la cual puede determinar la consecución, o no, de la acción propuesta por los agentes deliberativos. En CORTEX y más concretamente en el diseño de DSR, la idea es que la misma representación usada para almacenar la creencia en forma declarativa, como un grafo, contenga también la creencia espacial, como un grafo de escena, y que, además, la representación y sus funcionalidades no sean propiedad exclusiva de un solo agente. Esta decisión en DSR se basa en la importancia que tiene el denominado conocimiento *situacional* en Robótica, por el que todos los agentes acaban necesitando información del estado del robot en relación a su entorno, y así poder tomar decisiones eficientes para la consecución del objetivo común.

Por otro lado, en CORTEX es posible usar un módulo a cargo de la representación espacial que le permite operar contra la realidad o contra un entorno tridimensional simulado. Aunque su operación es local e independiente al resto de módulos, podría relacionarse con ellos mediante el uso de *buffers*. Estas características aproximan más a CORTEX a la idea propuesta en ACT-R/E [Trafton et al., 2009, Trafton et al., 2013] que a los otros ejemplos proporcionados.

También se extrae de las arquitecturas cognitivas que la combinación entre módulos para ofrecer un conocimiento refinado se consigue combinándolos explícitamente y aportando el resultado a un módulo central -*procedural module*- que también es alimentado por el resto de módulos. Este módulo central es *consciente* del estado actual. Toma las decisiones oportunas y coordina la ejecución de la tarea.

DSR pretende que la información procesada por el agente esté siempre disponible al resto, de forma que un agente encargado de una tarea diametralmente opuesta a la de otro, pueda servirse de esa información para completar su cometido, exhibiendo así un comportamiento oportunista. Sin esta elección de compartir una estructura entre los agentes, la posibilidad de comunicación entre ellos dependería de criterios de diseño apriorísticos difíciles de modificar. En este mismo sentido, la idea de una representación compartida como DSR, permite a la arqui-

tectura CORTEX aprender mediante las aportaciones perceptivas de los agentes que enriquecen así la creencia interna global acerca del estado actual del robot y de su entorno, y es propagada a todo el sistema.

Parte III

Deep State Representation

Capítulo 3

Deep State Representation

3.1. Introducción

La propuesta de *Deep State Representation* (DSR) que aquí se hace surge conjuntamente al nacimiento de CORTEX y culmina esta Tesis, aunque no su evolución y desarrollo. DSR está en el corazón de CORTEX y su objetivo es mantener una representación interna y única del robot y de su entorno, tanto a nivel simbólico como geométrico. Es el vehículo que articula el resto de partes y, a su vez, es una parte más de ella.

El capítulo se estructura de la siguiente forma: la sección 3.2 describe trabajos relacionados con la internalización del mundo exterior en una representación o modelo interno del mundo. Se analizan aspectos teóricos, también desde perspectivas ajenas a la Robótica y, finalmente, se estudian trabajos y soluciones concretos basados en la necesidad de un modelo que abstraiga la realidad.

En la sección 3.3 se argumentan las razones que nos han llevado a elegir la estructura computacional de un grafo dirigido multi-etiquetado (DAG) como la más idónea para albergar la representación interna del propio robot y su entorno.

En la sección 3.4 se describe brevemente la propuesta *Active Grammar Modelling* [Manso, 2013], dado que supone la solución de partida de DSR y es la base sobre la que se asienta su desarrollo. Asimismo, en esta sección se describe formalmente DSR.

En la sección 3.6 se describe la organización de la arquitectura cognitiva robótica CORTEX, y se ejemplifica su funcionamiento.

En la sección final 3.7 se analiza y discute la propuesta.

3.2. Trabajos relacionados

A nuestro modo de ver, si se pretende que los robots interaccionen con personas y realicen tareas de nuestra vida cotidiana, parece tan necesario construir razonamientos complejos a partir de una abstracción del mundo, como ser capaz de reaccionar rápida y eficientemente al dinamismo del entorno.

Aunque las arquitecturas basadas en tres niveles olviden en parte el contexto en aras de la compartimentalización, existe sin embargo un primer punto de inspiración en [Gat, 1998] cuando describe el papel del estado interno, es decir, la necesidad de que, de alguna forma y en algún sitio, se debe mantener y administrar la información del estado actual del robot. No obstante,

para Gat la información del estado interno es diferente en cada nivel, no necesita ser compartida, está implícita en la propia acción y obviamente no necesita una forma preestablecida de representación.

En nuestra opinión es ese estado interno quien proporciona el contexto, pero necesita ser modelado, compartido y conocido entre las partes siendo DSR una propuesta en esta dirección. Tal vez, la primera pregunta que nos surge sea: ¿cómo es ese modelo?, o incluso, ¿cuál es? Probablemente no haya una única solución válida y sea DSR una propuesta que atiende al realismo dependiente del modelo propuesto por S. Hawking [Hawking and Mlodinow, 2010], donde puede existir más de un modelo válido, siempre y cuando concuerde con las observaciones y pueda realizar predicciones. En palabras suyas:

> "El realismo dependiente del modelo se aplica no sólo a los modelos científicos, sino también a los modelos mentales conscientes o subconscientes que todos creamos para interpretar y comprender el mundo cotidiano. No hay manera de eliminar el observador — nosotros — de nuestra percepción del mundo creada por nuestro procesamiento sensorial y por la manera en que pensamos y razonamos. Nuestra percepción — y, por tanto, las observaciones sobre las cuales se basan nuestras teorías — no es directa, sino más bien conformada por una especie de lente, a saber la estructura interpretativa de nuestros cerebros humanos."

A esta observación de S. Hawking podríamos añadir que nuestra percepción no está solo conformada por la estructura interpretativa de nuestros cerebros, sino por las sucesivas acciones que decidimos realizar para acotar la incertidumbre en el ajuste de esos modelos. Este matiz trata de reajustar el valor de la acción en la interpretación y nos acerca a argumentos de gran interés propuestos por los teóricos dinamicistas [Eliasmith, 1999].

Si el observador fuese el robot con su limitada percepción y procesamiento sensorial, alojado en cada agente, la *estructura interpretativa* de nuestros cerebros no sería otra cosa que la estructura DSR propuesta en esta Tesis y que permite al robot "construir modelos que imaginan cómo es la realidad" y que además persigue lo expuesto por Holland [Holland, 2004]:

> "...at the heart of the mechanism is not just the body in the environment, it is a model of the body in a model of the environment."

Esta cita de Holland es otro punto de inspiración y forma parte de las llamadas Teorías de la Simulación de la Mente, como por ejemplo, del propio Holland [Holland, 2003] que defienden el uso de un modelo interno lo suficientemente sofisticado como para anticipar y simular el resultado de acciones virtuales en forma de percepciones virtuales.

Por tanto, tal vez se necesite un modelo que represente la realidad y se acerque al concepto descrito por Beetz et al. en [Beetz et al., 2010], al que denominó *deep-representation*:

> "representations that combine various levels of abstraction, ranging, for example, from the continuous limb motions required to perform an activity to atomic high-level actions, subactivities, and activities."

En el campo de la Robótica se pueden encontrar distintos enfoques para representar la realidad. La división más habitual se hace en términos de modelos métricos y modelos simbólicos. Dichos modelos han sido propuestos de diferentes maneras, al igual que para diferentes usos.

Tal vez, y en el nivel más alto de abstracción se encuentre la representación del conocimiento simbólico que ha sido el núcleo de la Inteligencia Artificial desde sus inicios [Russell and Norvig, 2009, Poole and Mackworth, 2010]. Esta representación abarca todo el espectro de las formalizaciones relacionales, tales como reglas de producción, los esquemas generales para representar el conocimiento, la lógica de primer orden, las redes semánticas, la toma de decisiones, etc. También se han creado lenguajes específicos, como por ejemplo: *Robot Learning Language (RoLL)* [Kirsch, 2009], que es usado en los modelos de aprendizaje sobre el comportamiento humano o el reconocimiento de la actividad humana. Incluso se han empleado modelos que tienen en cuenta al humano en su planificación, como en *Human-Aware Task Planner (HATP)* [Alami et al., 2006].

En cuanto a las representaciones del entorno inmediato, por un lado están los modelos topológicos y, por otro, los métricos. En el espacio intermedio los modelos híbridos combinan las ventajas de ambos tratando de minimizar las desventajas [Thrun, 2002]. Estos son los más habituales hoy en día.

Las representaciones que almacenan información cualitativa del entorno se conocen como representaciones topológicas. Suelen organizarse como grafos que recuerdan la configuración del mundo real teniendo en cuenta un conjunto de propiedades simbólicas y unas reglas de combinación lógica. Generalmente, los nodos representan conceptos abstractos, como objetos o lugares, y los arcos representan algún tipo de relación entre los nodos [Matarić, 1990, Angelopoulou et al., 1992]. El término simbólico se usa en el sentido de que las entidades pueden ser identificadas por un nombre y hacen referencia a entidades físicas o a otras entidades simbólicas (Fig. 3.1).

Las representaciones que almacenan medidas cuantitativas del entorno, se conocen como representaciones métricas. Almacenan valores métricos del entorno, normalmente en una estructura plana. Estas estructuras fueron diseñadas para dar respuesta a problemas relacionados con la navegación y, tal vez, su mayor exponente sean las rejillas de ocupación [Elfes, 1989], que consisten en un array de celdas que representan regiones contiguas del mundo real (Fig. 3.2).

La idea de crear y mantener una representación interna del mundo generó controversia en el pasado [Brooks, 1991, Noë, 2004], cuando se argumentó que crear una representación de la realidad en un modelo interno no tenía sentido, ya que la realidad es su mejor modelo. Esta cuestión del papel de la representación en la inteligencia es muy anterior a la creación de la Robótica y siempre ha ocupado un lugar central en los debates filosóficos sobre la inteligencia. Actualmente es cada vez menos frecuente encontrar posiciones extremas que defiendan tanto la ausencia total de representaciones internas, como la necesidad de una representación fiel de la realidad. Las mayoría de las propuestas actuales en Robótica plantean soluciones intermedias en las que hay una *tensión* entre el acceso activo al mundo para resituarse en él, y la creación y mantenimiento de una representación a diferentes niveles de abstracción que funcione como una **caché** del mundo, proporcionando estabilidad perceptiva, un acceso más eficiente a los procesos y una cierta capacidad de predicción. Volviendo a la propuesta de S. Hawking sobre realismo dependiente del modelo y llevándola a la Robótica, podríamos reformularla como la idea de construir un modelo suficiente como para mantener al robot en el mundo, ni muy separado de él por estar solo deliberando, ni muy atado a él por estar obligado a reaccionar ante cada evento. Extendiendo la idea de representación métrica que se ha comentado anteriormente, la representación del espacio es a menudo realizada usando árboles cinemáticos y grafos de escena, debido a que son habituales en motores gráficos y simuladores gráficos tridimensionales

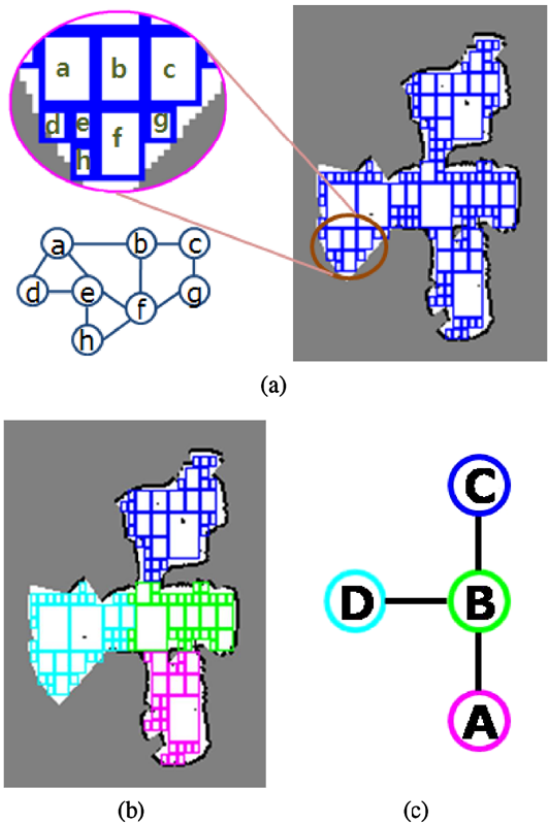


Figura 3.1: Ejemplo de modelado topológico *offline* (a) borrador inicial del modelo topológico, (b) grafo normalizado, cada cluster está representado con un color diferente , y (c) la topología extraída [Choi et al., 2011].

[1] [2] [3]. Son un importante punto de partida para alcanzar un conocimiento situacional que permita la representación espacial del estado del problema. También permiten al robot desarrollar, a partir de este conocimiento, simulaciones que le permitan anticipar situaciones, reacciones y decisiones, tal y como se indica en el trabajo propuesto por S. Wintermute [Wintermute, 2012].

> "... a highly detailed representation of the spatial state of the problem is maintained as an intermediate between the external world and an abstract representation. Actions can be simulated (imagined) in terms of this concrete representation, and the agent can derive abstract information by applying perceptual processes to the resulting concrete state."

Sin embargo, nuestra propuesta de *Deep State Representation* aboga por la representación integrada de la creencia del robot en una estructura articulada y unificada.

Hasta donde sabemos, los primeros trabajos que propusieron un grafo como una representación interna en una arquitectura robótica y centrados solamente en datos geométricos fueron: la librería de transformación de ROS, *tf: the transform library* [Foote, 2013], *BRICS Robot Scene Graph* [Blumenthal et al., 2013] y *RoboCog's InnerModel* [Bustos et al., 2013]. Todas ellas

[1] http://www.openscenegraph.org/
[2] http://wiki.ros.org/urdf
[3] https://www.khronos.org/collada/

Figura 3.2: *Occupancy grid map* [Thrun, 2002].

aparecieron en 2013 como respuesta a la necesidad de una estructura de representación centralizada de la cinemática del robot y del mundo. Aunque estas construcciones son importantes avances hacia mejores arquitecturas en Robótica, es necesaria una representación más rica y detallada para mantener el conjunto completo de creencias del robot.

Requiere aquí una especial mención la propuesta de Blumenthal et al. [Blumenthal et al., 2013], ya que es un referente significativo en el uso de los grafos de escena en el ámbito de la Robótica. Además tiene un elevado grado de similitud, desde el punto de vista conceptual, con la propuesta de DSR. Esto autores proponen la representación, mantenimiento y acceso compartido a un modelo del mundo tridimiensional usando un grafo de escena. Esta estructura central permite un acceso eficiente y unificado a la información almacenada por el resto de módulos del sistema.

Debido a que la mayoría de las arquitecturas actuales de control para Robótica están basadas en un conjunto de componentes distribuidos, el grafo de escena propuesto persigue evitar la fragmentación de los modelos del mundo en cada componente, a la vez que brinda un punto en común para que los resultados intermedios producidos por los componentes puedan ser aprovechados por los demás y no permanezcan aislados e inaccesibles en una parte del sistema.

Blumenthal et. al. proponen un grafo dirigido acíclico (DAG) de entidades geométricas que representa completamente la escena 3D. El significado de las relaciones padre-hijo entre los objetos del grafo es esencialmente geométrico y se almacena en unos nodos específicos llamados *transform node*. A su vez, los nodos pueden contener atributos semánticos o etiquetas usadas para identificar partes de la escena. Los nodos hoja pueden contener los datos en crudo proporcionados por el sensor. Esta característica es, en principio difícil de sostener desde el punto de vista de la eficiencia, ya que el volumen de datos de una, o varias, cámaras RGBD en tiempo real puede hacerlo intratable. Sin embargo, trabajos en curso de investigadores de nuestro grupo están proponiendo formas de solucionar este problema sin afectar al rendimiento global del grafo. Por otro lado, Blumenthal propone que sobre esa representación geométrica se pueden construir relaciones a más alto nivel. Estas serán utilizadas por los módulos de planificación

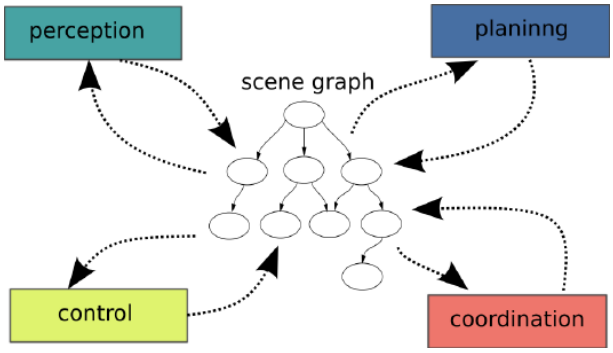


Figura 3.3: Vista general de un modelo del mundo basado en un grafo de escena. . Multitud de componentes pueden leer y escribir datos en el modelo del mundo [Blumenthal et al., 2013]

pero ese proceso no está descrito y queda como tarea futura.

Por tanto, nuestro objetivo era conseguir desarrollar una estructura compartida que nos permitiese representar la creencia del robot como una combinación de información simbólica y geométrica. Esta estructura representaría la creencia del robot sobre él mismo y el mundo que le rodea. Desde el punto de vista de la ingeniería debía ser flexible, escalable, adaptable y tenía que ser una parte fundamental de nuestro *framework* para Robótica orientado a componentes RoboComp [Manso et al., 2010], permitiendo una fácil integración con multitud de componentes ya existentes.

3.3. ¿Por qué un grafo como estructura compartida y de representación?

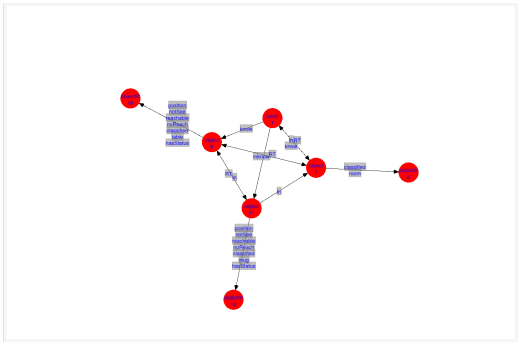
Formalmente, *Deep State Representation* es un grafo dirigido multi-etiquetado. Los nodos representan entidades genéricas y los arcos representan relaciones simbólicas y transformaciones geométricas. La arquitectura CORTEX se define estructuralmente como una configuración de agentes software conectados a esta representación, y la utilizan como su única vía de interacción.

La primera razón para usar un grafo como *Deep State Representation* es debido a que toda la información interna, que define el estado del robot y sus creencias sobre el entorno, necesita ser codificada en una *estructura muy genérica*. Esa estructura es un modelo de cómo se pueden interpretar y organizar los datos de los sensores. Como una estructura de datos general, los grafos pueden mantener cualquier conocimiento relacional compuesto de elementos discretos y las relaciones entre ellos. Dentro de esta amplia categoría se hallan casi todos los métodos de representación simbólica del conocimiento incluyendo *frames*, esquemas, reglas de producción, y también el conocimiento geométrico que el robot debe mantener sobre sí mismo y su entorno. Este conocimiento geométrico incluye instancias de los tipos de objetos reconocibles en el mundo, como por ejemplo, sillas, mesas, tazas u obstáculos genéricos de forma indefinida. También incluye el reconocimiento de cuerpos articulados, tales como el cuerpo humano donde se distinguen partes como brazos, cabezas, piernas, etc. Todas esas partes conforman una cadena cinemática, cuyos elementos se relacionan mediante transformaciones tridimensionales que, de hecho, se pueden codificar mediante un grafo, comúnmente llamado grafo de escena.

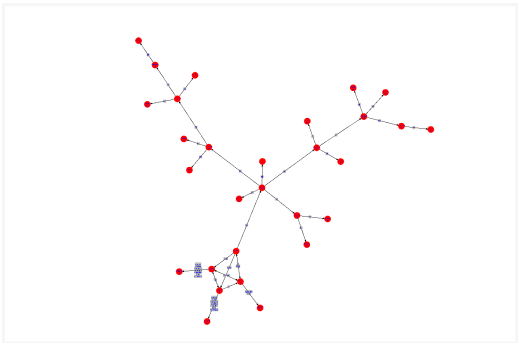
La segunda razón es que el grafo puede ser construido de manera que evolucione bajo

reglas generativas. Asumiendo que el tipo de nodos y arcos están predefinidos, el grafo puede evolucionar mediante la inclusión o el borrado de partes, causando cambios estructurales. También puede evolucionar modificando el valor de los atributos almacenados en los nodos y en los arcos. Los cambios estructurales pueden ser regulados por una gramática generativa que defina como el modelo inicial puede cambiar. Un ejemplo típico podría ser el de un robot que entra en una habitación desconocida y, tras explorarla, añade un nuevo nodo -que representa la habitación- al grafo. En este caso, la gramática impediría que el nuevo nodo se conectase a cualquier otro nodo que no fuera la puerta correspondiente.

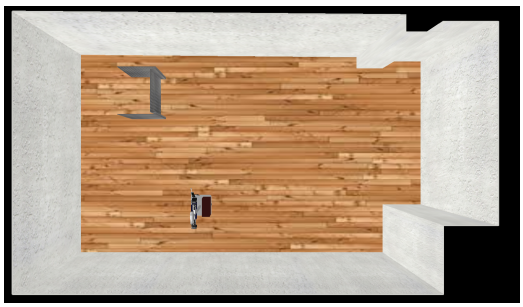
Entonces la estructura del grafo brinda la capacidad necesaria para almacenar elementos y sus relaciones, y combinado con una gramática que controle su evolución, permite un control coherente del crecimiento del modelo. La Figura. 3.4 muestra como el grafo evoluciona cuando una persona aparece en la escena. En el lado izquierdo solamente están representados el robot y la habitación. En el lado derecho, una persona entra en la habitación y el grafo la incorpora como un subgrafo correctamente relacionado con la estructura existente y con los atributos simbólicos que denoten lo que se conoce acerca de ella.



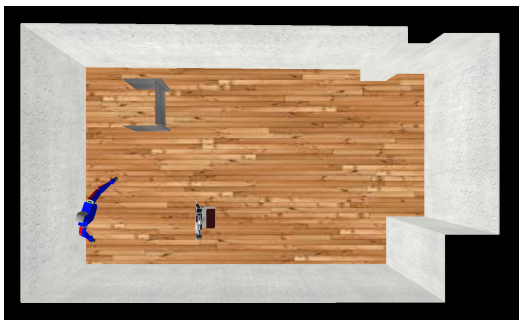
(a) Modelo inicial del mundo en DSR con el robot y la habitación.



(b) Una persona entra en la habitación. Cuando es detectada es insertada en DSR.



(c) Representación gráfica de la vista geométrica



(d) Representación gráfica de la vista geométrica cuando una persona es insertada en DSR.

Figura 3.4: Ilustración del cambio estructural en DSR. El nodo que representa el Robot ha sido colapsado para mejorar la visualización.

La tercera razón para usar un estructura de grafo es la posibilidad de traducirla en una instancia de PDDL [Manso, 2013]. Dependiendo de cómo se almacene en el grafo y de la versión de PDDL, habrá ciertas restricciones, pero una vez hecho da acceso a la mayoría de los algoritmos de planificación actuales. Estos algoritmos están en constante mejora por la comunidad especializada en la planificación simbólica y pueden ser incorporados fácilmente, proporcionando el acceso a una cantidad enorme de recursos que, de otra forma, habrían requerido un

enorme esfuerzo adicional.

La cuarta razón que da soporte a la elección de un grafo es su facilidad para visualizar su contenido usando cualquiera de la amplia variedad de herramientas y técnicas disponibles. La visualización en tiempo real es una característica crucial para evaluar el funcionamiento del robot, depurar el código de los agentes y proponer nuevas estrategias de desarrollo. La visualización de estas estructuras complejas se ha convertido en una herramienta imprescindible que debe evolucionar junto con la propia arquitectura. Actualmente, esta visualización 3D se extiende para hacerse interactiva y permitir al investigador literalmente interrogar e interactuar directamente con las creencias del robot.

3.4. Antecedentes de DSR

DSR se basa y extiende la *Active Grammar-based Modeling* (AGM) y la *Active Graph Grammar Language* (AGGL) propuesta en [Manso, 2013]. Este enfoque permite describir explícitamente las reglas gramaticales que pueden ser usadas para modificar el modelo del mundo del robot en un lenguaje específico del dominio. Manso propone en AGM un enfoque distribuido donde varios actores interactúan de acuerdo a una gramática que describe cómo puede ser creado y actualizado el modelo del mundo. Esta gramática es usada para guiar y controlar al robot y para representar su percepción del mundo.

La representación del modelo del mundo a alto nivel se realiza a través de un grafo semántico. Los nodos son símbolos que representan conceptos y las relaciones entre los nodos están etiquetadas en los arcos, como predicados. Los objetivos o misiones de los robots se expresan de la misma forma que los modelos -como un grafo-, pero en vez de expresar el modelo en su totalidad, expresan un patrón que debería encontrarse en un modelo que cumpla el objetivo. Esto impone una limitación: que todos los elementos relativos a los objetivos deben representarse de forma explícita en el modelo. El resultado de la planificación de la misión es una serie de reglas que habría que aplicar para que el robot evolucione el grafo hacia el estado final deseado. La Figura. 3.5 muestra una regla. La parte izquierda describe el patrón que debe existir en el modelo del mundo. Sí existe, el modelo mundo puede evolucionar conforme a los cambios que describe la parte derecha.

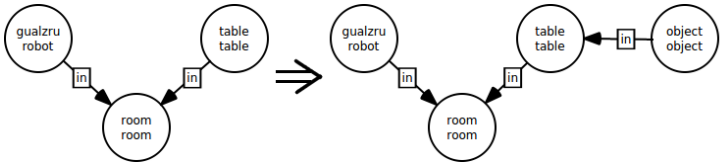


Figura 3.5: Ejemplo de una regla en AGGL: en el modelo del mundo del robot, los nuevos objetos pueden ser incluidos en la mesa sí el robot está en la misma habitación que la mesa [Manso, 2013].

El trabajo principal de esta Tesis ha sido mejorar la estructura anterior añadiendo la información geométrica. La nueva estructura puede mantener una representación interna unificada que incluye datos simbólicos y geométricos, donde ambos niveles pueden ser utilizados conjuntamente por los agentes para resolver las acciones del plan, consiguiendo así un modelo más útil y una descripción mas rica del estado del robot y del mundo.

3.5. Definición formal de DSR

Como dijimos, DSR es un grafo dirigido multi-etiquetado que almacena información tanto a nivel geométrico como simbólico. Los nodos almacenan conceptos que pueden ser simbólicos, geométricos o una combinación de ambos.

Los nodos contienen un número indeterminado de atributos en forma de una lista de pares *clave-valor* de tipo *string*. Dichos pares almacenan aspectos relacionados con la representación física y geométrica, tales como su malla tridimensional, la masa o el centro de gravedad, una textura o el nombre de una persona, las propiedades funcionales de un objeto, o la última frase dicha por el robot.

Los arcos representan relaciones entre nodos y poseen etiquetas que pueden almacenar un número indeterminado de atributos en la forma de una lista de pares *clave-valor* de tipo *string*.

Una definición formal de DSR puede ser expresada como $G = (V, E)$ donde V representa el conjunto de nodos $\{v_1, ...v_k\}$. Cada nodo v puede almacenar un número indeterminado de atributos como una lista de pares *clave-valor* de tipo *string* $v_l = \{< clave_1 : valor_1 >, ... < clave_n : valor_n >\}$.

E representa el conjunto de arcos $\{e_1....e_r\}$ donde e_i es definido como $< u, v, l >$ donde $u, v \in V$ y l es una lista de *string* que representa las etiquetas de cada arco. Cada etiqueta l puede almacenar un número indeterminado de atributos como una lista de pares *clave-valor* de tipo *string* $l_l = \{< clave_1 : valor_1 >, ... < clave_n : valor_n >\}$.

Las relaciones geométricas entre dos nodos $u, v \in V$ están representadas con una etiqueta fija RT y sus atributos contienen la matriz de transformación homogénea expresada como $RT_{uv} = \{rx, ry, rz, tx, ty, tz\}$. El sentido opuesto representa la matriz inversa $RT_{vu} = RT_{uv}^{-1}$.

Un camino en G entre dos nodos $u, v \in V$ representa un camino simbólico $C(u, v)$. Cuando es a través de etiquetas RT define la cadena cinemática $C(u, v)$. Una transformación entre nodos RT' puede obtenerse multiplicando las matrices de transformación de esta cadena. Téngase en cuenta que el camino inverso usaría las matrices inversas, por lo que se obtendría la transformación inversa.

En la Figura. 3.6, se muestra esta relación geométrica. En los arcos $< A, B, [at, RT] >$ y $< B, C, [RT] >$ existe la etiqueta geométrica RT que codifica la transformación rígida entre ellos. Estas transformaciones geométricas pueden ser encadenadas en sentido directo o inverso para trasladar el nodo al sistema de referencia deseado. En el arco $< B, D, [in] >$ y en el arco $< A, B, [at, RT] >$, las etiquetas in y at respectivamente, denotan predicados lógicos entre los nodos.

La estructura del grafo ha sido construida como una librería compatible con RoboComp [Manso et al., 2010]. Su clase principal contiene un vector de nodos y otro de arcos, además de las operaciones típicas para el manejo de nodos y arcos.

Las funciones más importantes están relacionadas con el acceso y la actualización de DSR por los agentes. Las tres operaciones que producen un cambio en la forma del grafo son: la inserción, el borrado y la modificación. A este tipo de operaciones se las denomina **cambios estructurales**. Alternativamente, la actualización de los atributos de un nodo o de un arco, es denominada **cambio no estructural**.

Los cambios estructurales se realizan primero en la copia local de DSR que posee el agente. Este propone su nueva representación de la realidad para ser incluida en DSR, la cual debe ser aceptada gramaticalmente.

Los cambios no estructurales afectan a los atributos de los nodos y los arcos, también se

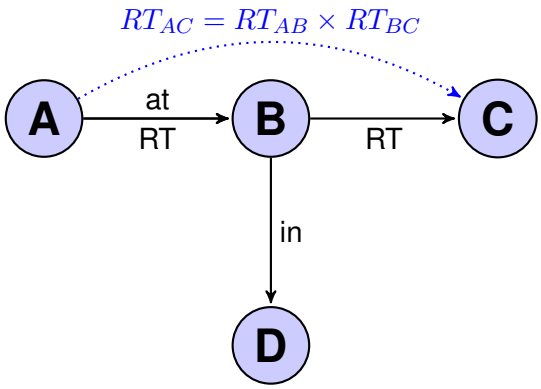


Figura 3.6: Representación unificada como un grafo dirigido multi-etiquetado en DSR.

realizan primero en la copia local, pero se actualizan en DSR cuando el agente quiere, sin necesidad de supervisión.

3.6. Organización de CORTEX alrededor de DSR

El funcionamiento de la arquitectura CORTEX, como un conjunto de agentes que interactúan a través de DSR, se explica fácilmente si nos lo imaginamos como un gran sistema dinámico. A partir de un estado cuasi-estacionario, los agentes de percepción tratan de mantener la representación interna sincronizada con el mundo. Pero cuando se solicita una nueva misión, entonces se genera un plan y se inyecta en el grafo. Esta alteración crea un desequilibrio para el que el sistema entero reacciona tratando de restaurar el equilibrio inicial. En este proceso el sistema expande su representación interna, capturando mas detalles del mundo exterior. Se incluye el nuevo conocimiento y la siguiente perturbación comenzará. Este proceso puede ser visto como una reacción endógena en DSR, provocada por un cambio exógeno en la realidad.

En la Figura. 3.7, las rodajas que rodean DSR representan a los agentes, que han sido desarrollados como redes de componentes software. Los agentes implementan las funcionalidades usadas habitualmente en Robótica como navegación, diálogo, detección de objetos, manipulación o planificación de tareas, y la comunicación entre ellos se hace a través de DSR. Podemos decir que la acción comienza cuando, a partir de un modelo inicial definido a priori, la representación interna es bombeada a todos los agentes en cada evento, interno o externo, que a través de un agente produce un cambio en ella.

Cada agente mantiene una copia local de DSR que puede ser usada y modificada a voluntad. El agente decide cuándo publicar los cambios introducidos en su copia local y el resto del sistema será consciente de ello. Otros agentes pueden percibir nuevos cambios como eventos externos, similares a los eventos producidos por sensores conectados directamente a él. Los agentes, en su comportamiento, pueden combinar ambos tipos de eventos para provocar cambios más conscientes del contexto. Un ejemplo típico es la navegación social, donde un agente de navegación sería capaz de elegir entre la forma de evitar a una persona y la de evitar a un objeto inanimado, sin necesidad de que el agente de navegación dispusiera de un detector de personas ya que dicha información está contenida en DSR.

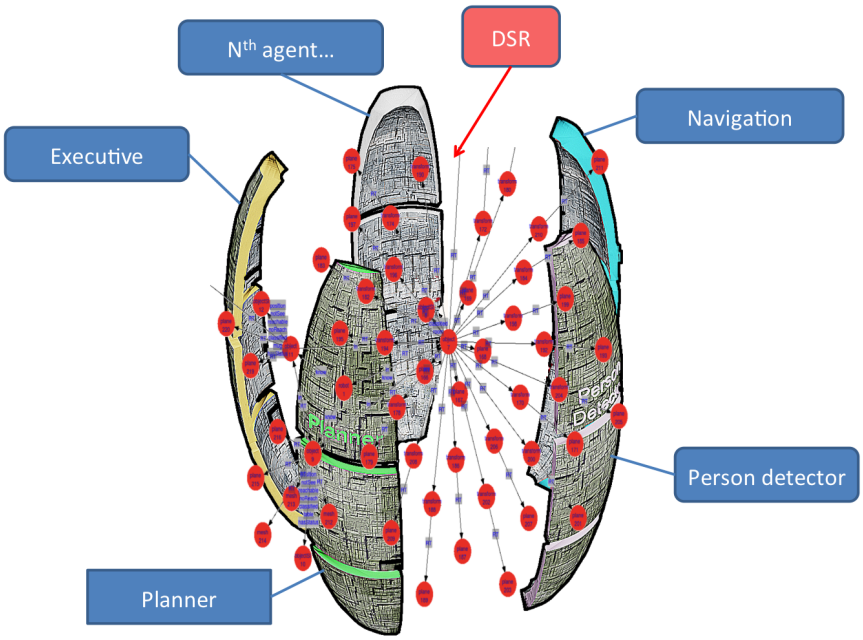


Figura 3.7: Visión general de DSR y su ubicación dentro de la arquitectura robótica cognitiva.

3.6.1. Ejemplo: un escenario simple

Para ilustrar el uso de DSR en la arquitectura, así como el proceso de actualización y propagación de los cambios, usaremos un ejemplo muy sencillo. En este ejemplo un grafo inicial ha sido creado con un mundo inicial donde solo existen dos elementos, un robot, representado con el nodo R, y el mundo representado con W (Figura. 3.8). En este escenario el robot tiene como objetivo detectar la presencia de una persona y, a continuación, aproximarse a ella hasta encontrarse a una distancia adecuada para comenzar una conversación.

$G = (V, E)$ where $V = \{W, R\}$, $E = \{< W, R, [RT] >\}$,

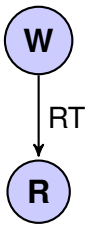


Figura 3.8: Estado inicial de DSR.

Para lograr estos objetivos, serán necesarios los siguientes agentes: (i) *Executive*, responsable de la coordinación de las acciones del plan y consciente de los cambios en DSR. Usa el conocimiento del dominio para comprobar si un cambio estructural es válido; si es así, el nuevo DSR es publicado. (ii) *Planner*, responsable de la planificación a alto nivel de la misión actual, integrando planificación y re-planificación, monitorizando y aprendiendo habilidades. (iii) *Person detector*, como responsable de la detección y el seguimiento de personas y (iv) *Navigation*, que está a cargo del movimiento del robot y su localización.

Para simplificar, en esta representación del mundo el robot es representado simplemente como un nodo y no se han dibujado los nodos y arcos que definen su estructura física. Podría interpretarse como que el subgrafo que lo representa está colapsado en o dentro de su nodo. La posición del robot esta definida en el arco $< W, R, [RT] >$, en concreto en la etiqueta RT, que representa la transformación entre el mundo W -el origen del sistema de coordenadas- y el robot R.

El conocimiento del dominio es modelado como un conjunto de reglas gramaticales. La primera permite modificar el grafo desde su estado inicial (Figura. 3.8) a un estado en el cual una persona ha sido detectada. Cuando una persona aparece en la escena -factor exógeno-, se provocará una reacción endógena. Esta reacción consistirá en insertar el modelo de un cuerpo humano en DSR de forma que se sincronice con la nueva realidad. La regla es descrita en la Figura. 3.9.

(a) Lado izquierdo. (b) Lado derecho.

Figura 3.9: Regla para detectar personas.

Como muestra la Figura. 3.9 el lado izquierdo es la precondición de la regla y el lado derecho la acción. En el lado derecho se especifican los cambios a realizar cuando una persona es detectada. Por tanto, el agente insertará el nodo persona P y creará un enlace geométrico entre W y P, y un enlace simbólico entre R y P con la etiqueta *detected* (Figura. 3.10). Esta propuesta deberá ser verificada por el agente *Executive* el cual aceptará el cambio si puede ser generado por la gramática. Si es aceptado, la siguiente acción del plan será solicitada al agente *Planner*

$$G = (V, E), \text{ where } V = \{W, R, P\}\ ,$$
$$E = \{< W, R, [RT] >, < W, P, [RT] >, < R, P, [detected] >\}$$

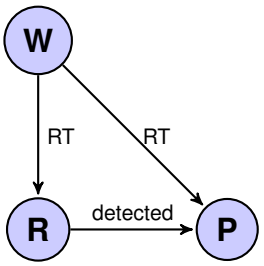


Figura 3.10: Estado de DSR cuando una persona es detectada

En este estado necesitamos otra regla que nos permita transitar hacia nuestro siguiente objetivo que es aproximarnos a las persona detectada. Está regla es definida como se describe en la Figura. 3.11.

(a) Lado izquierdo. (b) Lado derecho.

Figura 3.11: Regla: *approach person*.

La acción *approachPerson* necesita a ambos agentes funcionando a la vez. Por un lado *Person detector* mantiene la posición de la persona actualizada en la representación interna y,

por otro, el agente *Navigation* conduce al robot acercándolo hasta la persona, al mismo tiempo que actualiza su posición en el grafo. Como la actualización de estos valores geométricos son cambios no estructurales, éstos no son comprobados por el *Executive* y son propagadas inmediatamente al resto de la arquitectura.

$$G = (V, E), \text{ where } V = \{W, R, P\}\ ,$$
$$E = \{< W, R, [RT] >, < W, P, [RT] >,$$
$$< R, P, [detected, approachPerson] > \}$$

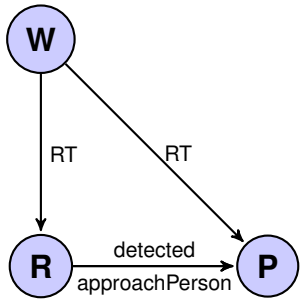
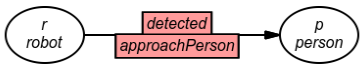
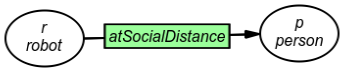


Figura 3.12: Estado de DSR cuando el robot se acerca a la persona.

Desde este estado se necesita otra regla para alcanzar el objetivo final: detenerse a una distancia prudente de una persona para interactuar con ella. La definición de esta regla es mostrada en la Figura. 3.13.



(a) Lado izquierdo.



(b) Lado derecho.

Figura 3.13: Regla: *at social distance*.

En este último paso se ejemplifican alguna de las bondades de la representación unificada. Simplemente necesitamos que un agente, por ejemplo el agente *Navigation*, proponga un cambio en DSR cuando la distancia entre el robot y la persona es la adecuada. El agente tiene acceso a ambos símbolos y a la rotación y traslación de la persona respecto del robot. Por tanto, puede calcular fácilmente cuándo la distancia entre el robot y la persona es la adecuada. A continuación, propone un cambio en el modelo interno reemplazando las etiquetas innecesarias *detected* y *approachPerson* por la de *atSocialDistance*. La Figura. 3.14 muestra el estado final de DSR.

$$G = (V, E), \text{ where } V = \{W, R, P\}\ ,$$
$$E = \{< W, R, [RT] >, < W, P, [RT] >,$$
$$< R, P, [atSocialDistance]\}$$

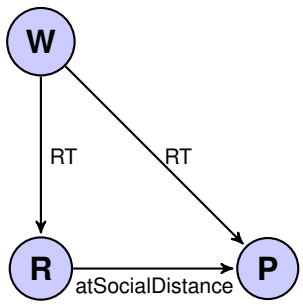


Figura 3.14: Estado de DSR cuando el robot se encuentra a la distancia adecuada de interacción.

Como se ha podido ver esta representación unificada permite a los agentes coordinarse a través de las acciones contenidas en el plan. Estas acciones están codificados en reglas que proporcionan tanto las condiciones necesarias que tienen que existir previamente en la representación, como las transformaciones que el modelo interno sufrirá tras completar la acción.

Las reglas pueden ser reutilizadas en diferentes dominios. Gracias al uso de diferentes agentes que utilizan DSR para compartir el contexto de la acción y combinan sus acciones para la consecución de la tarea, es posible abordar distintos tipos de escenarios e incorporar nuevas tareas fácilmente

Por último, el uso de DSR como una representación central compartida por los agentes ahorra tiempo de desarrollo, permite una integración más rápida de las habilidades y facilita el trabajo colaborativo.

3.7. Conclusiones

En este capítulo se ha explicado por qué DSR es el corazón de la arquitectura cognitiva CORTEX, y por qué organiza sus agentes alrededor suyo. Se ha definido y explicado cómo alcanzar el concepto de *Deep State Representation* propuesto, materializado en un estructura de representación interna de la realidad, en forma de grafo dirigido multi-etiquetado, su origen y trabajos relacionados, así como los mecanismos que le permiten ser compartido entre agentes. Se ha hecho también especial hincapié en su morfología, intrínsecamente concebida para ser usada como vehículo de razonamiento, al mismo tiempo que como representación de la creencia del robot.

Deep State Representation es el mayor logro de esta Tesis. Es su condición de representación unificada, que abarca el conocimiento geométrico y simbólico, que puede evolucionar en la escala de tiempos que el robot necesita para realizar la tarea, y que además lo hace de acuerdo a un conjunto de reglas del dominio, lo que consigue que sea una herramienta útil para el desarrollo futuro de la Robótica cognitiva. En consecuencia, las características más relevantes de DSR se pueden resumir en tres propiedades clave:

- *Estructural*: en DSR todo elemento, geométrico o simbólico que se inserta debe hacerlo en un lugar concreto y debe estar en relación con los otros elementos. Este enfoque difiere enormemente de los sistemas distribuidos más frecuentes que carecen de una representación central y en el que los agentes intercambian una gran cantidad de mensajes o *topics* para mantener una representación parcial de las creencias de robot.
- *Generativo*: DSR puede crecer o decrecer para adaptarse al entorno: objetos, personas, etc. son añadidos o borrados, pero siempre deben hacerlo de acuerdo al patrón de las reglas y a partir de un diccionario inicial de nombres.
- *Adaptativo*: DSR es continuamente modificado por los agentes, actualizando enlaces y nodos. Por ejemplo, cuando una taza es cogida por el robot y pasa de ser representada como *hija* de la mesa, a ser representada como *hija* de la mano, en el sentido cinemático del término. Esto permite reflejar los cambios en la estructura del entorno y actuar y predecir conforme a ellos. Los cambios en el grafo se propagan y son visibles para todos los agentes de la arquitectura, proporcionando la información necesaria para realizar operaciones sensibles al contexto.

La conjunción de estas tres propiedades definen las bases de nuestra propuesta de representación del conocimiento para su uso en las arquitecturas cognitivas. A partir de ella, el resto de elementos y funcionalidades pueden ser incorporadas. De la misma forma, prepara el camino

para la integración y desarrollo de los procesos relativos al razonamiento y la planificación espacial. En nuestra opinión esta forma de representación compartida situacional-simbólica del yo y del entorno, también facilita el avance hacia la creación de mecanismos computacionales de consciencia, basados en el meta-razonamiento sobre funcionamiento interno de la arquitectura.

Capítulo 4

El agente detector de personas: detección, reconocimiento y representación

4.1. Introducción

En la Robótica social, y más concretamente en el campo de la interacción humano-robot (HRI), es imprescindible que el sistema cuente al menos con un perceptor de personas, robusto, rápido y fiable con el que poder detectar, reconocer y representar a los humanos durante la interacción con ellos.

A lo largo del desarrollo de esta Tesis se han publicado trabajos derivados del objetivo central, aunque fuertemente vinculados y siempre inspiradores del resultado final. Este capítulo es un claro exponente de esta situación ya que se basa en el desarrollo de un agente, que originó el artículo más notable hasta la fecha del autor.

El artículo [Calderita et al., 2013b] propone un método para la captura del movimiento humano (*Human Motion Capture* o HMC), usando como sensor una cámara RGB-D. La estimación del movimiento se realiza en dos etapas: en la primera, se utilizan librerías comerciales para extraer las posiciones 3D de los segmentos corporales de la persona a partir de la información de la cámara. En la segunda, se propone un método basado en un modelo de persona, conocido de antemano y cinemáticamente restringido, para filtrar estas posiciones. Este modelo es representado por una cadena cinemática abierta en forma de árbol que representa la estructura cinemática de un humano, considerando los grados de libertad de cada articulación y sus proporciones medias. La comparación de las posiciones proporcionadas con las poses que el modelo puede adoptar permite eliminar posturas imposibles y filtra el ruido del sensor. Al mismo tiempo, el modelo adapta su forma para aprender las dimensiones reales del cuerpo detectado y permitir un filtrado más preciso.

Este uso de una representación interna de la realidad, que es contrastada con un modelo conocido y antropomórficamente correcto, así como el uso de un árbol cinemático donde los nodos representan las articulaciones del esqueleto de la persona, están íntimamente ligados al desarrollo posterior de DSR y su filosofía.

El resto del capitulo esta organizado de la siguiente forma:

La sección 4.2 contiene una breve revisión en el campo de la captura del movimiento humano utilizando cámaras RGB-D, analizando sus ventajas y desventajas, así como su funcionamiento.

La sección 4.3 muestra el método propuesto en [Calderita et al., 2013b], el cual mejora la

detección de la captura del movimiento humano (HMC) frente a los resultados de un software comercial, al aplicar un filtro a los resultados basándose en un modelo cinemático. El uso de un modelo proporciona claras ventajas frente a otras alternativas cuando se conoce de antemano lo que se está buscando. Esta restricción no es tan severa como pudiera parecer ya que en Robótica casi siempre hay un conocimiento previo disponible, innato o aprendido, que hace posible la interpretación de los datos sensoriales.

La sección 4.4 explica cómo el agente detector de personas construye la representación interna del cuerpo humano en DSR, mediante un subgrafo con las características cinemáticas correspondientes.

4.2. Detección y modelado de la figura humana usando cámaras RGB-D

Los sistemas tradicionales de captura del movimiento humano (HMC) se basan en un conjunto de marcadores colocados sobre el sujeto para capturar sus movimientos. Estos sistemas son caros, tienen que ser empleados en ambientes controlados, suelen ser invasivos para la persona y por lo general requieren tediosos procesos de calibración [Moeslund et al., 2006, Bandera, 2010].

En los últimos años se han investigado diferentes alternativas basadas en visión artificial, con el objetivo de solventar las carencias de los sistemas basados en marcadores. Estas iniciativas buscan un sistema de bajo coste capaz de capturar el movimiento *on-line*, a la par que pretenden lograr sistemas de HMC sin marcadores, con la suficiente calidad y dirigidos a entornos no excesivamente controlados. A pesar de los considerables esfuerzos realizados en esa dirección, el grado de éxito logrado es limitado [Moeslund et al., 2006]. Una parte de sus limitaciones se debe a los siguientes problemas específicos:

- *Detección de la persona*. La percepción de objetos y personas es complicada en un entorno ruidoso no controlado. Los algoritmos de detección de objetos basados en visión estéreo se veían afectados por las variaciones de iluminación, sombras y superficies sin texturas. Sensores más sofisticados, tales como sensores de profundidad basados en infrarrojos, también son afectados por la percepción parcial y las oclusiones. Los algoritmos de detección de personas, no sólo tienen que hacer frente a estos problemas, sino que también tienen que discriminar entre las personas y el resto de objetos.

- *Identificación de las partes del cuerpo*. Con el fin de capturar el movimiento de una persona es necesario identificar las partes de su cuerpo. Esto es una tarea compleja y si tiene que ser lograda sin marcadores, prendas de vestir o ropa especial, su complejidad aumenta considerablemente.

- *Seguimiento del movimiento*. Los sistemas HMC no solo pretenden detectar poses estáticas, sino también seguir el movimiento completo de un humano. Los movimientos naturales pueden implicar movimientos rápidos que superan la tasa de captura de la mayoría de los sistemas basados en visión artificial. Las pérdidas en el seguimiento de las articulaciones se convierten en un problema común para ellos.

- *Velocidad*. Con el fin de evitar el inconveniente anterior y proporcionar una descripción más precisa del movimiento, los sistemas HMC basados en marcadores comerciales son capaces de capturar el movimiento humano a altas velocidades. Las frecuencias de muestreo pueden superar los 60 *fps*. Sin embargo, requieren etapas de post-procesamiento para refinar los resultados y filtrar valores atípicos. Los sistemas HMC *on-line* tienen que abordar el problema de proporcionar una estimación robusta del movimiento a una velocidad lo suficientemente alta. Este requisito restringe la carga computacional de los algoritmos utilizados.

Para hacer frente a estas limitaciones, se han propuesto diferentes métodos no invasivos de captura de movimientos. Dichos métodos de estimación y seguimiento de los movimientos de un objeto articulado complejo se pueden dividir en dos categorías principales [Agarwal and Triggs, 2006, Poppe, 2007]:

(i) Basados en el aprendizaje o sin modelos (*model-free*), que utilizan métodos probabilísticos y de búsqueda para inferir la pose humana mediante características de la imagen.

(ii) Basados en modelos, que usan un modelo computacional o matemático de la persona para estimar su postura.

4.2.1. Enfoque *Model-Free*

Estos enfoques estiman directamente la postura del cuerpo en el espacio. Se basan en clasificadores entrenados sobre extensas bases de datos compuestas por conjuntos de pistas visuales -*image cues*- asociadas con las poses.

Estos procesos pueden requerir mucho tiempo de procesador y tener que lidiar con problemas de aparición de mínimos locales o de ambigüedades en la silueta [Agarwal and Triggs, 2006, Mori and Malik, 2002]. La generalización del conjunto de las posturas puede ser también difícil si este conjunto no es lo suficientemente completo [Demirdjian et al., 2005]. A pesar de estos problemas, muchos investigadores han utilizado estos enfoques debido a su capacidad para extraer correctamente posturas en escenas naturales complejas, incluso cuando se cuenta con una sóla imagen de la persona [Moeslund et al., 2006].

Las propuestas sin modelo (*model-free*) han experimentado un renovado y creciente interés desde la aparición de sensores RGB-D a un precio asequible, y de nuevos algoritmos rápidos y robustos en la estimación de la pose del cuerpo humano. Entre los sensores destacó el sensor PrimeSense, que proporciona un mapa de profundidad del entorno [Calderita et al., 2012] con una resolución de unos pocos centímetros, y a un precio muy barato. Respecto a los algoritmos para procesar esta información del sensor RGB-D, destaca la publicación de Shotton et al. [Shotton et al., 2013]. Esta contribución describe el método utilizado en el sensor Kinect de Microsoft para extraer la pose humana a partir de los datos de profundidad que proporciona el sensor RGB-D de PrimeSense. Sus principales características son las siguientes:

(i) Utiliza imágenes de profundidad en lugar de imágenes en color para inferir posturas humanas.

(ii) Utiliza un bosque -conjunto de árboles- de decisión para clasificar las características extraídas de la imagen de profundidad.

(iii) Los árboles de decisión han sido entrenados utilizando cientos de miles de poses obtenidas mediante modelos virtuales y registradas con cámaras RGB-D también virtuales.

Las principales características del método de Shotton et. al. son su robustez y velocidad. Es capaz de capturar el movimiento de todo el cuerpo -a 30 fps- en entornos no controlados, sin imponer ningún requisito sobre el sujeto. También es capaz de proporcionar estimaciones para las articulaciones ocluidas. Sus principales inconvenientes, por otro lado, son sus dificultades para inferir una pose correcta de los miembros inferiores, sus limitaciones en la captura de movimiento fino, como por ejemplo el movimiento de los dedos, y su falta de coherencia en la cinemática de las poses estimadas. El algoritmo proporciona, por tanto, posiciones 3D de diferentes partes del cuerpo, pero no comprueba la longitud de las extremidades, los límites, ni las colisiones.

4.2.2. Enfoque *Model-Based*

Estas propuestas se basan en el uso de un modelo humano para ayudar en los procesos de detección y seguimiento del cuerpo humano. Estos enfoques no requieren de largos procesos de aprendizaje sobre grandes repositorios de ejemplos y son, en principio, más robustos frente a las ambigüedades, posturas incorrectas y auto-oclusiones, ya que el modelo ayuda a inferir una posición válida para un dato percibido y proporciona estimaciones de las poses de las articulaciones ocluidas [Stenger et al., 2001, Sminchisescu and Triggs, 2003]. Por último, no necesitan depender de extensas fases de entrenamiento, ya que la información sobre la cinemática y la dinámica de los humanos se puede almacenar en el propio modelo [Agarwal and Triggs, 2006, Moeslund et al., 2006]. Entre los enfoques basados en modelos, la opción más común es el uso de modelos cinemáticos explícitos, en forma y apariencia, normalmente usando técnicas de análisis mediante síntesis [Moeslund et al., 2006, Azad et al., 2007]. Otros métodos sugieren la inclusión y creación de nuevos modelos a partir de observaciones [Starck and Hilton, 2008].

Si bien estos métodos son adecuados en términos de la calidad del seguimiento del movimiento y la velocidad de captura, tienen problemas para identificar a la persona y a las partes de su cuerpo. Se requieren, para iniciar el proceso de captura, algoritmos de ajuste del modelo al sujeto. Estos algoritmos están limitados por el nivel de detalle del modelo utilizado. Los modelos más toscos son generalmente más fáciles de identificar en el proceso de inicialización y, al mismo tiempo, reducen la carga computacional en el proceso de seguimiento. Sin embargo, son menos precisos en sus estimaciones. Por otro lado, hay parámetros del modelo -p. ej. longitudes y anchuras- que deberían adaptarse al sujeto en particular para capturar correctamente su movimiento.

Algunas contribuciones asumen modelos con dimensiones fijas, pero esta estrategia conduce a estimaciones inexactas de las posturas [Poppe, 2007]. Otros enfoques utilizan una etapa de inicialización [Sminchisescu and Triggs, 2003] o se basan en métodos de estadística a priori [Grauman et al., 2003]. Zhang et al. incluso propone utilizar la calibración manual del modelo para ajustarlo a la persona [Zhang et al., 2012]. El ajuste *on-line* de estos parámetros también se ha abordado, en sistemas que emplean múltiples cámaras [Mikić et al., 2003].

Es posible, sin embargo, utilizar modelos para reforzar estimaciones de pose previamente obtenidas. Así, un sistema capaz de enfrentarse con éxito a los problemas de identificación de personas y segmentos corporales podría verse reforzado por un modelo que proporcionara coherencia cinemática a los resultados. Además, los sistemas que utilizan un modelo para refinar poses percibidas pueden beneficiarse del uso de modelos complejos y realistas, ya que estos

modelos no tienen que ser inicializados en procesos de ajuste que suelen ser computacionalmente costoso debido a que, por lo general, usan métodos recursivos.

Las propuestas más recientes combinan, por un lado aproximaciones sin modelo en el proceso perceptivo *bottom-up* y, por otro, aproximaciones con modelo en el proceso *top-down* [Ganaphathi et al., 2010, Zhang et al., 2012]. Sin embargo, el uso de un modelo no sólo ayuda en la resolución de posturas ambiguas o en la obtención de los ángulos de la articulaciones, sino que como señaló Poppe [Poppe, 2007] en su estudio sobre HMC basado en visión, los enfoques basados en modelos pueden aplicar restricciones para reducir el espacio de posturas posibles y eliminar aquellas no factibles. Estas restricciones incluyen límites en el ángulo de la articulación, velocidad, aceleración y restricciones por colisión [Sminchisescu and Triggs, 2003, Deutscher and Reid, 2005, Yamamoto and Yagishita, 2000]. Aunque estas técnicas se han utilizado ampliamente en la última década en aplicaciones de HMC basadas en visión, en el año 2013, sólo se habían utilizado marginalmente con cámaras RGB-D.

4.3. Filtrado de los datos de la pose usando modelo

Aunque los beneficios de usar una aproximación sin modelos son claros, como se ha visto en el análisis bibliográfico, para el agente detector de personas se ha decidido utilizar un enfoque basado en modelos para reforzar los resultados, y mejorar la calidad de los datos de salida. Así, se ha desarrollado un algoritmo de HMC de dos pasos, en el que la primera etapa proporciona una estimación de la postura en bruto, mientras que el segundo paso refina esta estimación utilizando criterios basados en modelos. Concretamente, se propone un sistema que utiliza un modelo cinemático humano para mejorar los datos proporcionados por un dispositivo RGB-D.

Los algoritmos usados en la primera etapa usan un enfoque sin modelos, con lo que no se garantiza la validez de la postura proporcionada. Así, la longitud de las extremidades puede cambiar drásticamente debido a ruido y fallos en la detección (por ejemplo, un objeto en la mano es considerado como una parte del brazo generándose una estimación incorrecta de la longitud de éste). Los límites de las articulaciones tampoco son comprobados y un codo, por ejemplo, podría doblarse hacia atrás sin ningún impedimento. Las auto-colisiones no se comprueban y las ambigüedades en las posturas no se resuelven. Nuestra propuesta aborda este problema mediante el uso de un modelo de la cinemática de un ser humano, con el objetivo de garantizar la validez de las posturas obtenidas. El método consta de dos etapas, tal como se representa en la Figura 4.1. El primer paso captura los centroides 3D de las diferentes partes del cuerpo. El segundo paso utiliza un filtro basado en un modelo para estimar una pose válida del humano a partir de estos datos. Los datos de entrada capturados también se utilizan para adaptar el modelo a las dimensiones y proporciones particulares del humano. Ambos procedimientos, el chequeo de la posición y la adaptación del modelo, se ejecutan en paralelo y no reducen la frecuencia de muestreo del sistema original de HMC, que sigue siendo de 30 fps.

El detalle de funcionamiento de nuestro sistema de HMC se encuentra descrito en [Calderita et al., 2013b] y no se explicará con más detalle aquí. También se pueden encontrar en esa publicación los resultados de las pruebas a las que fue sometido para su evaluación. No obstante, nos gustaría destacar el siguiente resultado relativo al aprendizaje de las proporciones del humano, ya que es una característica importante del sistema propuesto. El sistema de HMC propone un algoritmo de aprendizaje de la longitud de las extremidades que es capaz de converger hacia valores precisos y estables para personas de diferentes alturas, género y proporciones.

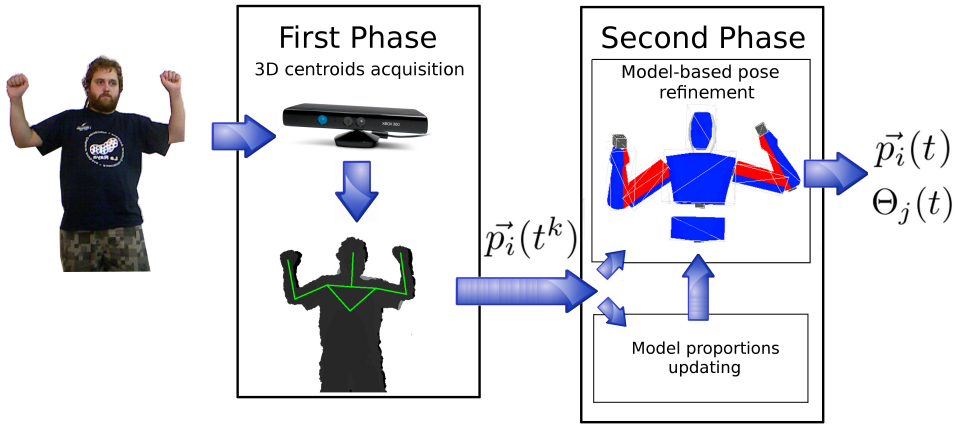


Figura 4.1: System overview.

El algoritmo de estimación de pose realiza dos tareas simultáneas, como muestra la Figura 4.1. En la segunda fase se estima la pose del humano usando un modelo, y se adaptan las dimensiones del modelo de acuerdo al individuo que está siendo capturado. Esta segunda tarea se logra usando un filtro adaptativo, el cual toma como entrada el conjunto de centroides 3D proporcionados por el dispositivo RGB-D $\vec{p_i}(t^k)$. Para adaptar el modelo del cuerpo humano a las dimensiones de la persona, dividimos el proceso en tres fases. *inicialización, aprendizaje y fijación*

Inicialización. El procedimiento de aprendizaje comienza usando las relaciones antropométricas publicadas en [Contini, 1972]. Estos datos se recavaron usando una amplia y variada muestra de personas. Proporcionan una media de las proporciones de hombres y mujeres de edad media relativas a H -la distancia desde el suelo a la parte superior de la cabeza-.

Aprendizaje. H se calcula, de acuerdo con el método de Contini, como $H_e/0{,}936$, H_e siendo la distancia del suelo al ojo. En nuestro algoritmo H_e se estima como la media del centroide de la cara. Una vez que la altura del modelo ha sido actualizada, las longitudes de sus extremidades, $\bar{l}_j(t^k)$, también son adaptadas a los datos percibidos por el humano durante las primeras N muestras usando un filtro *alpha-beta* con coeficientes variables en el tiempo -Ver algoritmo 1-.

Algorithm 1 Aprendizaje de las dimensiones del humano.

Require: $Limbs, CorrectedLimbs, k \in Frames \wedge (k > 0)$
1: $\alpha = \frac{k-1}{k}, \beta = \frac{1}{k}$
2: **for all** $l \in Limbs, \bar{l} \in CorrectedLimbs$ **do**
3: **if** $(k < N) \vee (|\bar{l} - l| > MaxDist)$ **then**
4: $\bar{l}_{t+1} \leftarrow \alpha * \bar{l}_t + \beta * l_t$
5: **end if**
6: **end for**
7: **return** $CorrectedLimbs$

las cuales pueden ser re-muestreadas como una media móvil acumulada:

$$\forall k \in [0..N] \quad \bar{l}_j(t^k) = \bar{l}_j(t^{k-1}) + \frac{l_j(t^k) - \bar{l}_j(t^{k-1})}{k} \tag{4.1}$$

donde $l_j(t^k)$ es la longitud de las extremidades instantáneas para el *frame* k, calculado como la distancia entre los centroides 3D de dos articulaciones consecutivas. Después de N muestras,

la medida se considera estable. El aprendizaje se detiene, y solo se reanuda, si se descubre una disparidad significativa entre el modelo y los datos.

Fijación. Tras actualizar las proporciones del modelo, si la distancia entre la longitud aprendida y la longitud calculada crece por encima de un cierto umbral -$MaxDist$-, la longitud de ese miembro se vuelve a evaluar mediante el ajuste del valor k a cero. Esto provoca que se realice la fase de aprendizaje de nuevo, pero sólo para ese miembro.

La Figura 4.2 muestra las longitudes del antebrazo derecho en cada *frame*, calculadas como la distancia euclídea entre los centroides proporcionados por el algoritmo de detección *model-free* de la primera etapa -en azul-. También muestra las longitudes del antebrazo del modelo, calculadas utilizando el algoritmo propuesto para el aprendizaje de las dimensiones de las extremidades humanas -en rojo-. Se representa la longitud del antebrazo medida manualmente -312 mm-, como una línea verde en la Figura 4.2, mostrando convincentemente cómo la longitud del antebrazo del modelo converge al valor real, a pesar de los ruidosos valores proporcionados inicialmente.

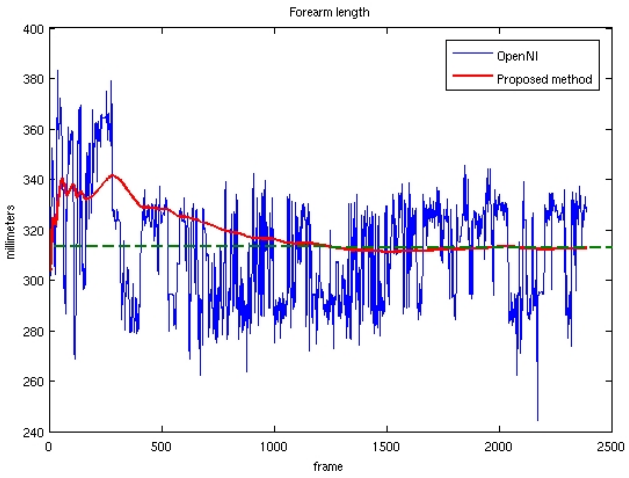


Figura 4.2: Longitudes del antebrazo proporcionado por el algoritmo de la primera etapa (*tracker* de OpenNI) y por el sistema adaptativo *model-based* propuesto [Calderita et al., 2013b].

4.4. Representación jerárquica del humano en DSR

Cuando tratamos con figuras humanas se debe definir un modelo adecuado en función de la aplicación prevista. Su precisión puede tener un profundo impacto en la validez de las predicciones que se derivan de su uso. No obstante, se deben asumir simplificaciones con el fin de tener un modelo manejable. Estas son aceptables siempre y cuando el usuario sea consciente de sus limitaciones. Por lo tanto, un buen modelo no es necesariamente un modelo muy complejo, sino un modelo que empareje las necesidades del problema con el conjunto mínimo de parámetros pertinentes. La mayoría de los modelos del cuerpo humano asumen que sus partes son rígidas, y no se pueden romper. Por supuesto, esto es una aproximación de la realidad. Para los modelos más precisos pero también mucho más costosos computacionalmente, el método de los elementos finitos [Williamson, 1980] es la técnica clásica más utilizada. Es aplicada al cálculo

de deformaciones debidas a las tensiones aplicadas de acuerdo con las leyes constitutivas del material. Estas leyes son muy complejas para los tejidos vivos [Baerlocher, 2001].

Por otra parte, el modelado virtual se inicia en los años sesenta, con los modelos desarrollados por la industria aeroespacial y de automoción. Su objetivo es diseñar y evaluar las cabinas de los pilotos. Aunque hoy en día son muchas las áreas que usan modelos virtuales para evaluar y simular, que abarcan desde fenómenos físicos y procesos industriales, hasta la evaluación de estructuras arquitectónicas. El problema, cuando se trata de modelos virtuales del ser humano, reside en la complejidad del propio cuerpo con su intrincado sistema musculoesquelético y su complejo sistema de control motor [Walker and Walker, 2001]. Afortunadamente, las partes del cuerpo de nuestro modelo no pretenden ir tan lejos, se representan como mallas fijas de triángulos que se ajustan a las proporciones humanas, y se utilizan para calcular las colisiones. Estas mallas se han modelado utilizando modelos de baja resolución poligonal para lograr mayor velocidad de cómputo -técnica habitual entre los desarrolladores de videojuegos-.

En general, las mallas que representan cada parte del cuerpo están unidas rígidamente a diferentes sistemas de coordenadas. El conjunto de sistemas de referencia y las mallas adjuntas está organizado jerárquicamente en una estructura llamada grafo de escena -*scene-graph*- [Martz, 2007]. Un grafo de escena es una estructura de árbol jerárquico utilizado en gráficos por ordenador para organizar los datos espaciales y la representación eficiente. Los grafos de escena son un concepto genérico que puede incluir diferentes conjuntos de vértices, tales como objetos, partes del cuerpo, fuentes de luz, materiales o texturas. Cada uno de estos nodos puede tener cero o más hijos. La transformación definida para un nodo padre también afecta a sus hijos, pero no al contrario. Por lo tanto, el grafo de escena facilita el establecimiento de dependencias entre los diferentes elementos del modelo. El cálculo de las transformaciones geométricas se propaga a través de estas dependencias. Es esta característica la que lo convierte en una herramienta interesante para representar el modelo humano en DSR, al mismo tiempo que también se puede utilizar para representar el robot y su entorno. Por ello, el agente detector de personas utilizará esta representación, insertándola en el DSR cuando una nueva persona sea detectada.

En nuestro modelo -Figura 4.3-, el nodo *person* representa a la persona. Internamente este nodo equivale al torso, -el nodo raíz del árbol cinemático del modelo del humano-. El agente transforma la pose del humano referenciada en el sistema de coordenadas de la cámara RGB-D al nodo raíz de la escena -world-. Los nodos descendientes se enlazan mediante arcos geométricos y representan la transformaciones rígidas tridimensionales entre el hijo y el padre. Las mallas tridimensionales y los planos se incluyen en la estructura como un nodo hijo asociado a ese nodo, por ejemplo el nodo *person* en la Figura 4.3, tiene un hijo, el nodo 15, que contiene su malla tridimensional (*mesh*). El enlace también está etiquetado geométricamente permitiendo adaptar la transformación de la malla, de forma que puedan usarse mallas tridimensionales de terceros[1], sin importar el sistema de referencia en el que fue creada, o su punto de giro.

Se puede observar también que, si las mallas son eliminadas de esta representación, la información acerca de las relaciones cinemáticas entre las diferentes partes del cuerpo sigue almacenada en la propia estructura. Esta simplificación normalmente se llama esqueleto o cadena cinemática.

El dispositivo RGB-D Kinect y el software comercial asociado (*Microsoft Kinect SDK*)

[1] http://www.sweethome3d.com/freeModels.jsp
http://www.turbosquid.com
http://tf3dm.com/

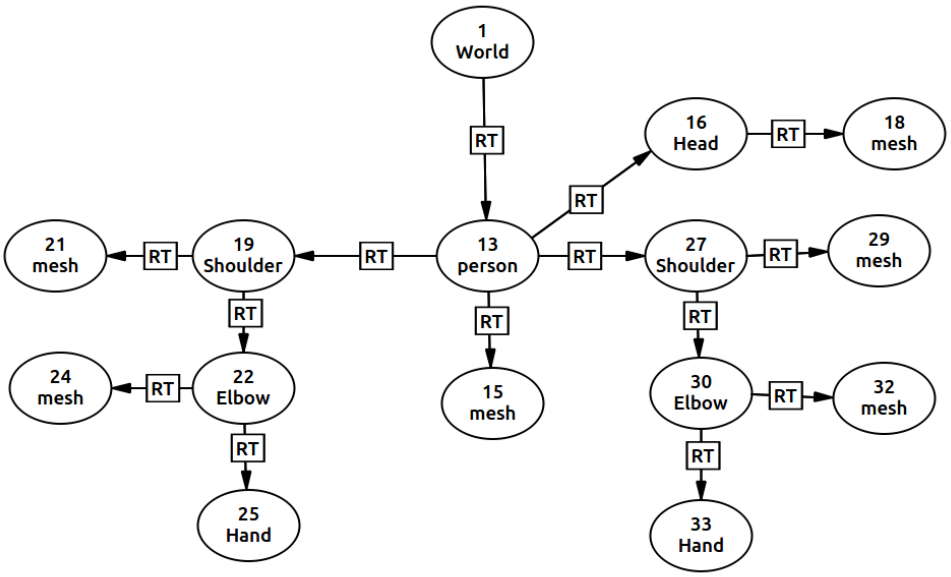


Figura 4.3: Parte superior del cuerpo en la vista 2D de DSR, enlazado al nodo raíz *World*.

pueden reconocer hasta seis usuarios en el campo de visión del sensor. De éstos, solo es posible obtener la posición de todas las articulaciones para dos usuarios. Este esqueleto proporciona información detallada acerca de la posición de veinte articulaciones del cuerpo humano. El agente solo utiliza las posiciones de catorce articulaciones. Las partes del cuerpo de nuestro modelo se distribuyen como se muestra en la Figura 4.4.

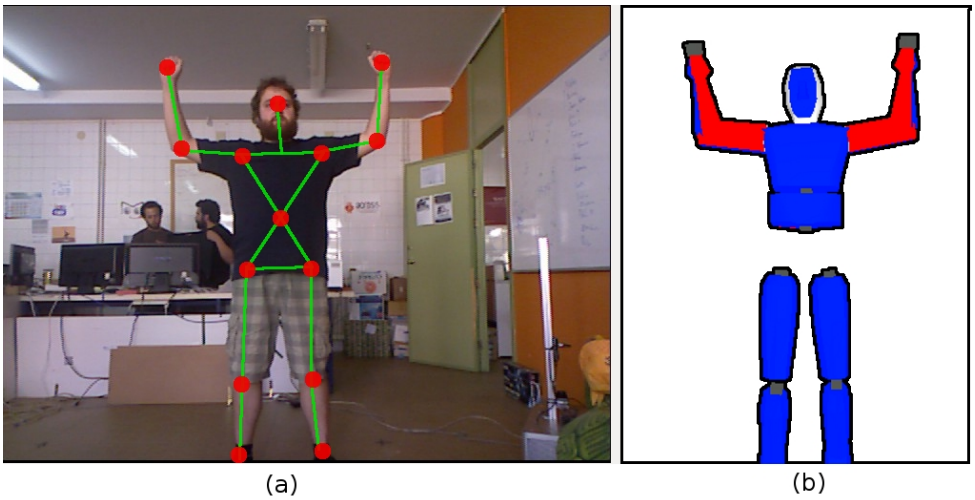


Figura 4.4: (**a**) Centroides 3D de las articulaciones -puntos rojos-; y (**b**) El modelo del humano con las mallas 3D.

Aunque a través del SDK se puede obtener la orientación de las miembros del esqueleto de forma jerárquica, su sistema de coordenadas y codificación de los datos no es directamente compatible con RoboComp. El agente posee métodos para crear la cadena cinemática del robot a partir de las coordenadas 3D de cada articulación. Las posiciones de las articulaciones están

referenciadas al sistema de la cámara y son transformadas al sistema de referencia del mundo. Es de esta forma como se obtiene, se representa e inserta el modelo del humano en DSR, garantizando la coherencia con el resto de elementos del grafo y con el resto de agentes de la arquitectura.

En la vista 2D se observa el nodo persona: el nodo raíz es del tipo *person* y a partir de él cuelgan el resto de partes. El nodo *person* en sí mismo representa a una persona desde el punto de vista cognitivo, pero también actúa como nodo raíz en la cadena cinemática que contiene la geometría de la persona. Así, al "colgar" de él el resto de partes que componen a un humano, y desde un punto de vista geométrico, proporciona a los agentes esa descripción más rica del mundo, en la que se ha insistido a lo largo de la Tesis, y que les permite lidiar con situaciones más complejas.

Por otro lado, es interesante destacar que los agentes son libres de elegir el sistema base al que referenciar los elementos que insertan en el grafo, pero la experiencia nos sugiere que aprovechando las relaciones geométricas el sistema de referencia más adecuado es el "mundo", el nodo raíz de la escena 3D. Esta elección nos proporciona una coherencia entre agentes y, además cuando es necesario, el agente puede cambiar el sistema de referencia de ese elemento al del nodo *robot* o al del cualquier otro nodo, propagando esa transformación a todos sus descendientes enlazados mediante enlaces RT.

El agente mantiene un vector -por razones de simplicidad- con los N árboles relativos a las personas que están siendo detectadas. Aunque, como se dijo, un dispositivo Kinect solo puede seguir todas las articulaciones de dos humanos, el agente ha sido diseñado para soportar la evolución de la SDK. Una iteración de su bucle general de funcionamiento podría resumirse en los siguientes pasos:

1. Actualizar el vector de N árboles relativos a las personas a las que actualmente se está siguiendo.
2. Obtener una lista de los identificadores de los nodos de personas que hay en DSR.
3. Compararlos con el vector de personas local.
4. Para los elementos que coinciden, actualizar el valor de los arcos de cada subgrafo que representa la persona en DSR, mediante la publicación de *cambios no estructurales*.
5. Eliminar o incluir en DSR aquellos subgrafos que no tienen su correspondiente en el vector local, mediante un *cambio estructural*.

Como se puede ver la representación del árbol cinemático es una parte más de DSR y constituye una integración transparente por la que cualquier agente puede acceder a la representación de las personas. Un ejemplo son las relaciones geométricas inferidas, como la distancia entre el robot y la persona, o la posición relativa de la mano. Por otro lado, el etiquetado simbólico -cerca, lejos, derecha, izquierda- atendiendo a razones geométricas, por ejemplo entre los elementos del mundo y la persona pueden ser evaluados y accedidos sin dificultad por el resto de la arquitectura. Un ejemplo de la flexibilidad que proporciona esta representación sería determinar por parte del robot qué objetos pueden ser alcanzados por una persona, y así generar una solicitud verbal dirigida al humano: "por favor, dame la herramienta que está a tu derecha". Este tipo de razonamientos llega a ser muy complejo sin una representación adecuada del robot y la persona en el entorno compartido, y la capacidad de adoptar geométrica y simbólicamente el punto de vista de la persona.

4.5. Conclusiones

Como primera conclusión es importante destacar que el agente que se ha descrito en este Capítulo tiene en el núcleo de su funcionamiento la idea básica que persigue la estructura DSR, y que no es otra que el uso de un modelo interno que representa la realidad, la cual es comparada con el modelo, una y otra vez.

Aunque ya hemos mencionado la función perceptora básica, y a la vez fundamental, del agente, que es la de detectar y mantener actualizada la información de las personas del entorno en DSR, es importante insistir en que sobre esa percepción básica se basarán las futuras tareas de interacción con el humano, y que estos comportamientos tienen como fuente principal el DSR. Es el análisis de su estado por otros agentes lo que desencadenará las acciones necesarias para conseguirlo, siendo extensible al resto de nuevos agentes que se incorporen. Es decir, en el futuro se harán necesarios más agentes que de alguna forma accedan a DSR y deduzcan relaciones entre los nodos, o modifiquen valores que les permitan, a ellos mismos o a otros agentes, responder adecuadamente a los eventos de su entorno.

En el campo de la interacción humano robot la representación geométrica del humano permite crear en el robot una cierta consciencia situacional de su presencia. Esta puede ser enriquecida con innumerables enlaces y atributos simbólicos, lo que permitiría anotar y mantener conceptos como preferencias, gustos, creencias, actitudes o aspectos propios de la personalidad humana. Este es sin duda un objetivo perseguido y deseado en este camino hacia la interacción más social, empática y próxima entre el robot y el humano. Probablemente los agentes relacionados con el diálogo jueguen un papel crucial en la extracción de esa información simbólica al humano.

Otros tipos de percepciones, como las relacionadas con el estado emocional del humano, deberían también formar parte de la representación. Este es un aspecto de DSR que se está empezando a investigar actualmente. De forma similar, DSR se presenta como un espacio adecuado para tratar con aspectos relativos a las cualidades de los objetos o ambientes, que permitan al robot realizar una acción. Sin duda esto recuerda al término *affordance* originalmente definido por Gibson, y más tarde llevado al campo de la HCI por Donald Norman [Norman, 2013], pero tal vez acercándolo más a la línea descrita por autores actuales [Zhang and Patel, 2006] que consideran que se pueden explicar representacionalmente.

En cualquier caso se observa la necesidad de disponer de un conjunto de agentes que enriquezcan DSR creando cambios estructurales que abstraigan poco a poco los datos crudos de los sensores. En el próximo Capítulo se verán algunos de los agentes existentes actualmente en CORTEX.

Capítulo 5

Alimentando DSR: agentes adicionales

Los agentes están organizados alrededor de DSR y este es el esquema general de CORTEX. Están encargados de una subtarea y pueden estar formados por más de un componente. Los agentes han sido desarrollados y verificados individual e independientemente por especialistas antes de su integración, lo que supone una ventaja añadida a la propuesta. Desde un punto de vista funcional es excelente para el trabajo en grupo y distribuido, incluso geográficamente. Si se quiere añadir un nuevo agente solo hay que conocer el "lenguaje común" de DSR para que su funcionalidad pueda ser usada por el resto de CORTEX

El capítulo se organiza con una sección por cada agente proporcionando su descripción y detalles relevantes. También se han recogido para cada uno de ellos las pruebas de verificación que se consideran más significativas.

5.1. Agente Navegación

La navegación en interiores es un componente clave en las arquitecturas actuales de control de robots autónomos. Como otros módulos complejos que forman parte de estas arquitecturas, la navegación se suele descomponer en varios elementos débilmente acoplados que forman un sistema distribuido. Los elementos típicos de navegación son (i) evitar colisiones percibiendo en el entorno; (ii) (re) planificación de rutas óptimas y seguras; (iii) (re) localización.

En CORTEX existen dos agentes diferentes a cargo de esta tarea:

- El alcance del primero, se limita a moverlo dentro de la habitación, el módulo de navegación utiliza un algoritmo reactivo, que permite al robot alcanzar un objetivo evitando posibles obstáculos y también incluye un sistema de localización básica basado en AprilTags [Olson, 2011]. Es usado en la plataforma robótica diferencial de nuestro robot Gualzru, del que se hablará en la Sección 6.2.
- El segundo se ha desarrollado para nuestro robot manipulador Shelly (Sección 6.3) como un nuevo agente de navegación para manipuladores móviles, que se basa en la idea de bandas elásticas [Quinlan and Khatib, 1993]. Los algoritmos se basan en la representación compartida del robot y el entorno disponible gracias a DSR. El objetivo futuro es integrar la navegación y la manipulación en una misma acción.

5.1.1. Navegador reactivo

El primero se basa principalmente en el algoritmo R-ORM. Es una evolución del *Obstacle Restriction Method* (ORM), propuesto por Chamorro y Vázquez-Martín [Chamorro and Vázquez Martín, 2009]. Este algoritmo divide el problema de alcanzar un determinado objetivo, evitando las colisiones, en un conjunto de subproblemas. Cada uno de estos subproblemas consiste en alcanzar un determinado subobjetivo. La distribución de los obstáculos percibidos alrededor del robot definen las posiciones en las que se colocan estos subobjetivos. El método genera comandos de velocidad y de giro que permiten al robot alcanzar el objetivo final navegado a través de una secuencia de subobjetivos.

El R-ORM tiene varios problemas cuando se aplica a robots reales, que pretenden ser empleados en experimentos de larga duración y entornos dinámicos. A continuación se detallan las modificaciones que se han incorporado al algoritmo R-ORM para resolver estas deficiencias.

Una vez que el R-ORM establece un subobjetivo, tiene que llegar a él antes de calcular uno nuevo. En experimentos reales, esta decisión provoca situaciones ineficientes, donde una oclusión temporal del objetivo final obliga a la selección de subobjetivos distantes que atraen al robot incluso cuando el objetivo es visible de nuevo. Este problema se ha resuelto haciendo que el algoritmo se olvide de inmediato del actual subobjetivo y se dirija al objetivo final cuando se percibe como alcanzable. Con el fin de reducir la oscilación producida por cambios rápidos en el subobjetivo actual, se ha incluido un factor de inercia. Una vez que se selecciona un nuevo subobjetivo, se mantiene como el objetivo actual durante un cierto tiempo, que depende del valor de inercia. En el mismo sentido se puede ajustar su frecuencia de muestreo, de forma que pueda ajustarse el flujo de comandos de giro y avance a las características hardware de la base robótica.

Aunque R-ORM es bastante robusto en general, en algunas situaciones poco comunes, puede conducir al robot a una posición comprometida y "atascarse". En esta posición, no es capaz de moverse hacia el subobjetivo, ni establecer uno nuevo. Se ha incluido un parámetro de "paciencia" para hacer frente a estas situaciones. Si el robot se encuentra en la misma posición durante un tiempo superior a este valor, -y no se ha llegado a ningún objetivo aún- girará sobre si mismo durante un cierto tiempo hasta que pueda establecer un nuevo subobjetivo [Calderita et al., 2014a].

Por otra parte, como el algoritmo R-ORM se basa sólo en la odometría para ubicarse, cuando el robot se ha estado moviendo durante un tiempo, es imposible alcanzar el destino debido a los famosos errores sistemáticos y no sistemáticos. El sistema de navegación necesita elementos adicionales para evitar estos errores de posicionamiento. Para evitarlo se han colocado marcas de AprilTags [Olson, 2011] en una ubicación predefinida. Por lo general tres son suficientes. Una de ellas cerca de una posición inicial predefinida, y dos marcas más situadas en lugares visibles. Las posiciones de estas marcas se fijan con anterioridad en DSR. Son elementos conocidos en el ambiente, por lo que son parte del modelo interno del robot. Son un nodo más con el identificador de la marca, ubicado en una posición relativa al origen de coordenadas de las escena. Gracias al agente detector de objetos (detallado en la Sección 5.8) estas marcas son detectadas y sirven para corregir la odometría.

El autor de esta Tesis ha sido el responsable de desarrollar y probar este agente, cuyos resultados se muestran en [Romero-Garcés et al., 2015]. El algoritmo de navegación fue percibido por las personas que interactuaron con el robot como seguro y fiable, aunque sus movimientos han sido evaluados como no demasiado naturales. El navegador, por otro lado, genera

en general movimientos un poco lentos, ya que el primer objetivo fue la seguridad. Sería interesante mejorar los algoritmos empleados y establecer la velocidad del robot en función de las circunstancias. Una solución podría ser utilizar algoritmos de interpolación de tercer orden [Huang et al., 2001] para cambiar los valores de velocidad -en lugar de la interpolación lineal empleada actualmente-. Este cambio debe producir cambios de velocidad más naturales y suaves. Probablemente será capaz de aumentar la velocidad máxima, aunque por supuesto, debe mantener el nivel de seguridad alcanzado.

5.1.2. Un agente de navegación para manipuladores móviles

Este agente se basa en la idea de las bandas elásticas. Fueron introducidas por primera vez por Quinlan and Kathib [Quinlan and Khatib, 1993] como un método para cerrar la brecha entre la planificación de rutas y la navegación sin colisiones en tiempo real. Una banda elástica, es una construcción teórica obtenida por un planificador, que consigue una conexión entre la ruta planificada y el mundo real, a través de la interacción con el sensor láser. La banda funciona como un pegamento, entre la representación interna de la ruta y las limitaciones impuestas por la física del mundo. El camino puede ser "roto" por el cruce de un humano, por ejemplo, y restaurado después. El controlador local, simplemente "ve" una pequeña perturbación que podría implicar un cambio en la velocidad. La Figura. 5.1, muestra una representación esquemática del robot, en una tarea de navegación, que termina cogiendo la taza que está sobre la mesa.

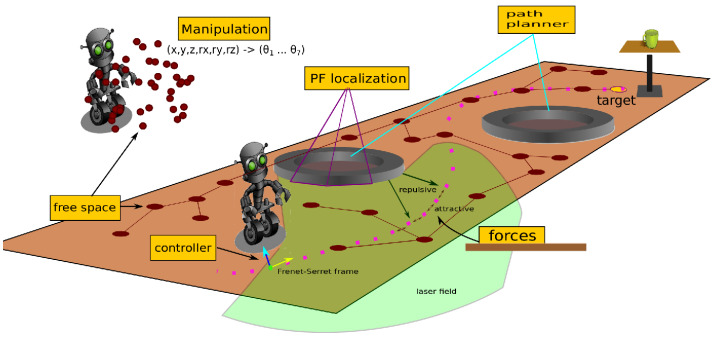


Figura 5.1: Vista esquemática del agente de navegación.

La técnica de las bandas elásticas no ha recibido demasiada atención en la comunidad robótica, tal vez, debido al mayor interés en métodos de control local más especializados, o en la planificación de trayectorias. Recientemente, la industria del automóvil ha mostrado interés en las bandas elásticas como una forma de calcular maniobras evasivas y servir de ayuda a los conductores. Más tarde, una extensión de la idea, propuesta como una cadena elástica *-elastic strips-*, fue presentada en [Brock and Khatib, 2002]. Su objetivo era adaptarla para robots con muchos grados de libertad. Todos los grados de libertad del manipulador son incluidos en el proceso de evitar obstáculos, al mismo tiempo que se obtiene la postura deseada. Una última generalización de la idea original fue publicada como hojas de ruta elásticas *-elastics roadmaps-* [Yang and Brock, 2009], incluyendo el proceso de planificación en el bucle.

Con objeto de simplificar la explicación, abordaremos aquí, y de manera somera, sólo la parte relativa al sistema de navegación dejando para la sección 5.6 la parte relativa al brazo. No obstante el esqueleto del sistema es el mismo y consta de las siguientes partes:

- La representación interna geométrica en DSR, permite introducir nuevos nodos que representan la creencia que el robot tiene acerca de su espacio de configuración libre *-free C-space-*.
- Sobre ese *free C-space* se obtiene un camino seguro. El camino es creado primero buscando el punto más cercano en el grafo a la posición actual de la base o la mano. A continuación, se añade el punto más cercano a la posición de destino en el grafo. Por último, se crea un camino a través del grafo que une ambos puntos.
- Para finalizar, ese camino es ejecutado, por la base o el brazo del robot, que tiene en cuenta lo imprevisible de la realidad.

5.1.2.1. Localización

La localización es una tarea crucial. Debe actualizar la posición estimada de la base del robot en su modelo interno. Algoritmos probabilísticos basados en el conocimiento parcial de la información geométrica del entorno, constituyen el foco de esta tarea. Se utiliza una variante del filtro de partículas CGR, propuesto en [Biswas et al., 2011]. Esto permite obtener una estimación de la pose del robot con respecto a un sistema de referencia global inicial. La representación del mapa del entorno, es realizada como una lista de líneas correspondientes a la parte inferior de las paredes y objetos conocidos. Este algoritmo es muy eficiente en términos de ciclos de *CPU* y en número de partículas, debido a que la función de usada es analítica y derivable. Una pre-ordenación de las líneas de oclusión y un bucle interno de minimización, mejoran el posicionamiento de las partículas usando el Jacobiano de la medida.

El localizador funciona a bordo de Shelly 6.3. Antes de su integración al agente de Navegación, fue probado intensamente en el *RoboLab's Autonomy Lab* -escenario de los experimentos de Shelly-. Un apartamento de $70m^2$ que está siendo acondicionado para la investigación robótica social. El primer experimento fue diseñado para medir la robustez del algoritmo de localización CGR en el mundo real, bajo condiciones altas de perturbación. El robot fue enviado, a una serie de lugares al azar. Se almacenó la posición de *ground truth* usando un conjunto de marcadores AprilTags. La Figura 5.2 muestra la evolución del error acumulado durante 90 metros de navegación ininterrumpida. La media es de $8cm$ y la desviación estándar es $3cm$.

Por tanto, se concluye que el localizador es robusto y puede estimar posiciones de la base desde distintas fuentes. Mantiene una ubicación fiable durante períodos largo de tiempo. Además las condiciones del experimento son similares a las de un apartamento.

Aunque su integración junto a los brazos, no está conseguida aún, su desarrollo podría ser alcanzado a medio plazo. No obstante lo más relevante desde el punto de vista de está Tesis. Es que la utilización y el flujo entre agentes de la representación interna permite, o al menos provee, de los elementos necesarios para abordar este tipo de tareas complejas.

5.2. Agente Conversación

El agente de conversación dota al robot con la capacidad de comunicarse con nosotros a través del diálogo. El agente procesa la información proporcionada por el canal auditivo, y genera las frases por medio de un *Text To Speech* convencional.

CORTEX no dispone, de ningún agente para utilizar la información acústica en un sentido geométrico, por ejemplo, para estimar la dirección de procedencia del sonido. Esto permitiría

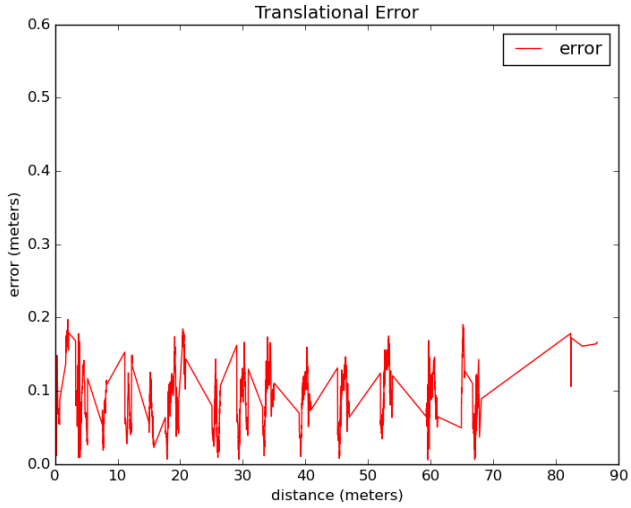


Figura 5.2: Error acumulado después de $90m$ de navegación ininterrumpida.

dirigir el robot hacia la fuente, consiguiendo, tal vez, una mejora en la captura del audio.

El proceso de reconocimiento de voz, se compone de dos etapas separadas: transcripción y comprensión de la transcripción -Figura. 5.3. El primer paso procesa la fuente de audio y obtiene la transcripción a texto más fiable. La generación de la transcripción se lleva a cabo mediante el uso de dos elementos internos: la acústica y los modelos de lenguaje.

El primer modelo representa la probabilidad de obtener un enunciado de entrada x, dado una secuencia de palabras w -trascripción-. El segundo modelo puntúa la transcripción w, utilizando la probabilidad conjunta de la secuencia de palabras. La probabilidad para cada palabra depende de la lista de las palabras anteriores $wi-1, ...w_{i-n}$. El modelo de lenguaje se genera siguiendo el n-*gram model* propuesto en [Jelinek, 1997], donde n define el número de palabras consideradas en la probabilidad conjunta. El agente utiliza *3-grams model* generado a partir del corpus COLA[Hofland et al., 2005]. Dicho corpus fue elegido porque incluye maneras informales de hablar. Esta gramática se utiliza para generar nuevas transcripciones para cada entrada de audio recibido, que pueda ser modelado con la gramática. Eso permite que el ruido ambiente sea desechado.

El proceso de comprensión es realizado por el agente utilizando la entrada de audio. La transcripción resultante es almacenada en una etiqueta semántica. Esta información semántica se integra en el agente, solo si la información es interesante para el proceso conversacional, es decir, si el audio reconocido está relacionado con el diálogo. En otro caso es descartada.

El proceso de clasificación es realizado usando el método de *Bag of Words (BoW)*[Wallach, 2006] en conjunción con un clasificador Bayesiano. El diccionario para el procedimiento *BoW* se obtiene con un proceso de selección variable que elimina palabras inútiles -como artículos o conectores-. Para el entrenamiento y la validación del clasificador, se han utilizado 750 frases, realizadas por una muestras de más de 25 personas. A partir de estas frases iniciales, se ha construido un modelo de gramática para cada una de las clases. Este modelo incluye variaciones aleatorias. Finalmente, 1800 frases diferentes se generaron para el entrenamiento y otras 600 adicionales para la validación.

Cuando un usuario habla, se obtiene la transcripción más fiable. La transcripción se transforma utilizando la *BoW*. Si el conjunto obtenido no incluye un número mínimo de palabras

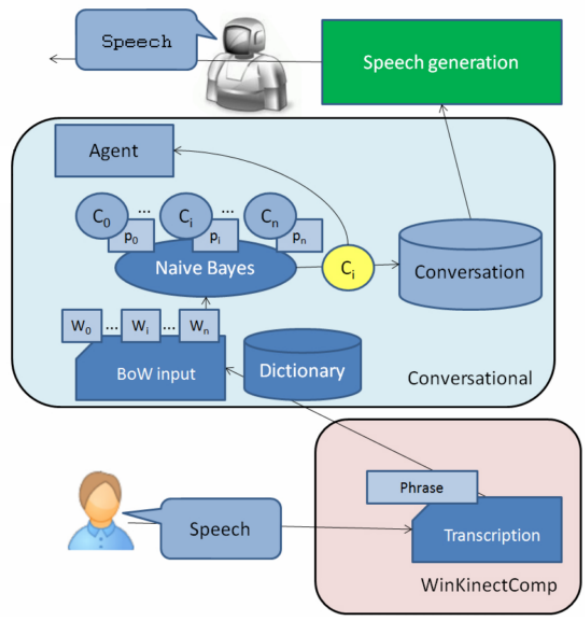


Figura 5.3: Agente de conversación. Reconocimiento y generación del habla.

clave, -incluidas en el diccionario-, la frase de entrada se considera sin sentido y se descarta. Si no es así, el conjunto de palabras de entrada se procesa utilizando un clasificador *Naive Bayes* y el conjunto de probabilidades de la salida se estudia clasificando la frase de entrada sólo si, C_i cuando $P(C_i|w)$ claramente supera el resto de las probabilidades $P(C_j|w)_{i!=j}$.

Como resultado de la clasificación, se obtienen dos escenarios diferentes: la pregunta del usuario y la decisión del usuario. En ambas situaciones el agente requiere de la generación de voz para responder con una frase apropiada. Sin embargo, las decisiones de los usuarios implicarán cambios en DSR, por ejemplo, una respuesta afirmativa desencadenará en el robot una acción diferente, a la que conseguiría una negativa. Pudiendo, por ejemplo, etiquetar la relación entre el robot y el humano, como interesado o no interesado. De la misma manera la consciencia espacial de DSR es usada en el agente, por ejemplo, para abortar la conversación aunque no haya terminado, si la persona ha desaparecido.

El desarrollo de este agente fue realizado el Dr. Jesus Martínez. Ha ido evolucionado a lo largo del tiempo, sus cambios más notables se han producido en la forma en que se modela el audio recibido de acuerdo al lenguaje. En la primera aproximación se modelo a través de llamadas a la API de Google con unos resultados razonables. Posteriormente se utilizó la SDK de Microsoft a partir de una gramática generada, que consiguió mejores resultados. En este punto, se evaluaron dos tipos de hardware de entrada con la intención de mejorar el proceso, de un lado, el array de micrófonos del dispositivo Kinect y del otro un micrófono direccional[1] manteniendo el resto de elementos. Se consiguieron mejores resultados con este último micrófono que con el dispositivo Kinect. El agente continuó mejorando y fue evaluado en el RockIn14 [2] donde resultó vencedor en la categoría *Functionality Benchmark Best Team (Speech understanding)* a bordo del robot Ursus.

[1] http://www.audio-technica.com/cms/wired_mics/cae8c23cfe000574/index.html
[2] http://rockinrobotchallenge.eu/rockin2014.php

5.3. Agente Reconocedor de Emociones

El fin último de la interacción entre el ser humano y una máquina, es conseguir que la comunicación fluya a través de una interfaz natural e intuitiva entre ambos interlocutores. Dado el avance y sofisticación de las máquinas actuales, las técnicas en experimentación y desarrollo en el área de la HRI y, de la HMI en general, persiguen que ésta vaya más allá de la interacción clásica basada en elementos hardware, p. ej. botonería, paneles de mando, controles remotos, ratones, teclados...

En este sentido, sería importante para una comunicación más natural que la interfaz permitiera a la máquina la detección del estado emocional de la persona, para lo cual, dos de los canales que proporcionan más información son: las expresiones faciales y el audio. Las expresiones faciales suelen ser obtenidas visualmente a partir de una señal de vídeo, en el enfoque tradicional [Liu and Chen, 2003], o obtenidas -o reforzadas-, con la información tridimensional de los nuevos dispositivos y algoritmos [Mayer et al., 2009]. Combinadas con las expresiones léxico-fonéticas obtenidas de una señal auditiva, son una fuente de entrada interesante en la detección de emociones. Incluso, pueden extenderse en un futuro al contacto mediante análisis de patrones corporales como la temperatura, el pulso cardíaco o la sudoración.

En los últimos años, la importancia que ha experimentado la detección de emociones en tiempo real. Surge de que éstas modulan prácticamente todos los canales de comunicación entre personas. Así, en mayor o menor medida, condicionan entre otras, nuestras expresiones faciales, gestos y posturas, nuestro tono de voz, e incluso, las palabras que empleamos. Pese a su importancia, no existe una teoría unificada que describa apropiadamente todo el amplio rango de emociones.

Clasificarlas en un número pequeño de categorías primarias -emociones puras- es claramente limitado, por lo que resulta normalmente preferible caracterizarlas usando dimensiones continuas -p. ej. negativo versus positivo- [Cowie, 2009]. De esta forma, la representación resultará más flexible y generalizable. Además, dado que el objetivo final de la detección de emociones, es determinar las posibles acciones a llevar a cabo por parte de la máquina, ésta no tiene que conocer forzosamente el estado emocional, sino sus implicaciones sobre la acción.

Varios autores [Cowie et al., 2001, Jaimes and Sebe, 2007] hacen hincapié en el hecho de que la emoción humana está generalmente expresada a través de diferentes modalidades en una interacción natural. Otros estudios [Zeng et al., 2009, Prado et al., 2012, Kessous et al., 2010] han realizado análisis multimodal de las expresiones a partir de la información de vídeo y audio. En la primera, analizando las expresiones faciales, los gestos, la postura del cuerpo humano en la interacción. En la segunda, analizando la señal de audio originada por el ser humano durante la conversación. Este análisis multimodal lleva, en general, a unos resultados más precisos que el uso de uno solo.

En CORTEX el agente dedicado a la detección de emociones utiliza un único canal para el análisis de emociones. El agente detecta el estado emocional del usuario usando información facial. Emplea un clasificador bayesiano para determinar el estado emocional del sujeto basado en el modelo de Unidades de Acción *-Action Units -AUs-* [Ekman and Friesen, 1978]. Como muestra la Figura. 5.4, el agente utiliza la malla de puntos clave *-Candide-3 model-* ofrecida por el sensor Kinect. A partir de este modelo, obtiene las características necesarias para determinar la emoción facial. Básicamente, estas características se establecen como distancias euclídeas entre los puntos clave de la malla -p. ej. la distancia entre el contorno superior de las cejas y el borde inferior de los ojos, o la que existe entre el contorno superior y el borde

inferior de los labios-. La evolución de estas distancias en secuencias de vídeo de corta duración se asigna directamente con las AUs. La red bayesiana dinámica implementada, utiliza propiedades antagonistas de alguna AU, para aumentar el rendimiento y reducir el número de variables a considerar en la red. El sistema propuesto, sólo utiliza 11 AUs, para reducir y optimizar el procesamiento de la información. Estas 11 AUs se agrupan de acuerdo a propiedades antagónicas y exclusivas, en un conjunto de sólo siete variables. De acuerdo con el trabajo de Ekman [Ekman and Friesen, 1978], cinco posibles estados emocionales son estimados por el algoritmo: feliz, triste, enojado, temeroso, y neutral.

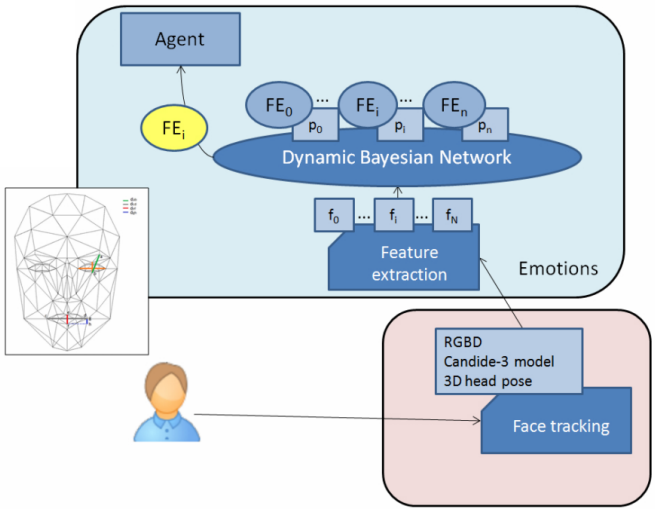


Figura 5.4: Proceso del reconocimiento de emociones.

Como dijimos, la detección de emociones juega un papel crucial a la hora de conseguir aumentar la empatía del robot con la persona. El agente cuenta con la capacidad de detectar el estado de ánimo del sujeto. Obviamente, del vasto conjunto de emociones humanas es capaz de reconocer un conjunto mínimo de emociones básicas: *neutro, tristeza, miedo, enfado, felicidad*.

Además, los valores no tienen porqué ser totalmente fiables, por lo que la salida será un vector cuyos valores tienen la forma de dupla -emoción, probabilidad-. En concreto, el enfoque propuesto logra una extracción de características faciales robusta y en tiempo real, mediante la aplicación consecutiva de filtros a la imagen RGB-D. El algoritmo base está descrito en [Cid et al., 2013]. Aunque ha sido reforzado en el agente. Para conseguirlo, se ha añadido la malla facial tridimensional al conjunto de datos de entrada. De forma que sea posible procesar las deformaciones de la cara asociadas a cada expresión facial utilizando el sistema FACS -*Facial Action Coding System*- [Cid et al., 2014]. Observamos el conjunto de emociones básicas reconocibles en la Figura. 5.5.

Una descripción más profunda del proceso se encuentra en el trabajo del Dr. Cid. [Cid et al., 2014] y es también el fundamento de su tesis. Estos trabajos fueron trasladados al agente por él y el Dr. Nuñez, dotando a CORTEX de una prometedora fuente de información para conseguir una mejor HRI. El agente, al igual que otros perceptores básicos, escribe en DSR la salida más probable del vector, como la cadena de texto asociada -neutro, tristeza, miedo, enfado, felicidad-.

La detección de emociones aportará un papel muy relevante en el futuro de nuestros robots, por ejemplo en Gualzru 6.2, podrá permitir conocer si la persona está disfrutando con la expe-

riencia y las reacciones que provoca, así como, ayudar a encauzar correctamente el proceso de diálogo.

De la misma forma es importante en el campo de la rehabilitación [Calderita et al., 2015], -y más si los pacientes son niños-. Tener la certeza de que la emoción del niño no es de miedo o enfado, y que, por contra tiende hacia un estado de felicidad, servirá para comprobar lo adecuado del ejercicio, ya que la motivación es fundamental en el tratamiento. Lo anterior repercute en beneficio para el profesional médico, quién dispondrá de más datos para la preparación y diseño de las sesiones.

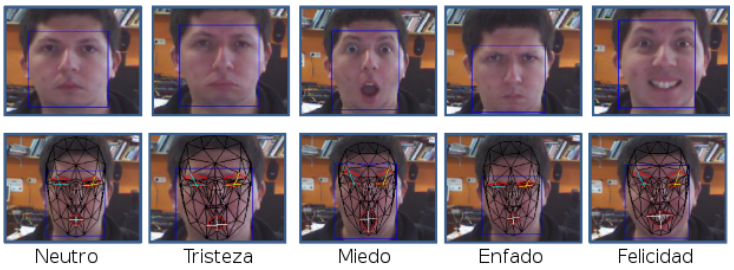


Figura 5.5: Conjunto de emociones básicas reconocible.[Calderita et al., 2015]

5.4. Agente Planificación

El agente de planificación es una instancia del framework PELEA [Alcázar et al., 2010]. PELEA, está formado por varios módulos, planificación, ejecución, monitorización, replanificación y aprendizaje. La Figura. 5.6 muestra una vista general de su integración en el agente. Se compone de cuatro módulos principales y el planificador de alto nivel. DSR es convertido a PDDL y es usado por los distintos módulos.

Monitoring Module

Este módulo es la parte principal del agente. Es responsable del control de los avances del plan durante su ejecución mediante la recepción del problema, el dominio y el estado, y su función es devolver la siguiente acción del plan. Al recibir el nuevo estado desde DSR, comprobará si ese estado era el que espera recibir, en caso afirmativo devolverá la siguiente acción. Si el estado recibido es distinto, tendrá que replanificar para afrontar la nueva situación.

Execution Module

El módulo de ejecución está a cargo de la interacción entre PELEA y el entorno. El entorno puede ser un simulador software -por ejemplo, MDPSim [Younes et al., 2005] ampliamente utilizado por la comunidad-, un dispositivo hardware -robot-, una aplicación de software, o un usuario. En particular, es responsable de iniciar el mundo de PELEA mediante la recepción del dominio y problema particular a resolver.

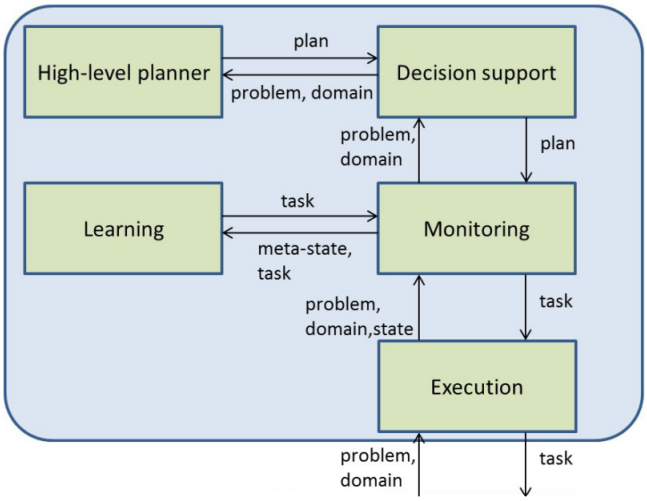


Figura 5.6: Representación esquemática del agente PELEA.

Decision Support Module

Este módulo es el encargado de recibir el dominio y el problema desde el módulo de monitorización y devolver un plan tras invocar al planificador de alto nivel. Cuando detecta una discrepancia, entre el estado observado y el esperado en el plan, el módulo, invocará al planificador para que actualice el estado del plan al nuevo. Este módulo puede ser configurado para ejecutar dos tipos de planificadores. En el agente, solo se usa el planificador Fast-Downward [Helmert, 2006].

Learning Module

En este módulo se infiere conocimiento a partir de la experiencia acumulada por el planificador durante la ejecución del plan. EL plan puede ser utilizado, incluso, para actualizar el dominio mejorando el proceso de planificación, por ejemplo, mediante el aprendizaje heurístico. También puede ser usado para reducir el tiempo de respuesta de la tarea, mediante el uso de una política de aprendizaje basada en el plan *-plan-based policy-learning-* [Manso et al., 2014].

Cada vez que sea requerido por el agente Ejecutivo, el agente planificador ofrece la siguiente acción a ejecutar teniendo en cuenta el plan inicial generado y el estado actual de DSR. Si el estado actual es uno esperado, el agente devuelve la siguiente tarea de acuerdo con el plan inicial. Si el estado actual es inesperado, -p. ej. porque el nivel de la batería ha disminuido rápidamente-, se genera un nuevo plan -desde cero o revisando el antiguo-. Esto genera una nueva acción que tiene en cuenta las nuevas condiciones[Pulido et al., 2014].

5.5. Agente Ejecutivo

El agente ejecutivo se encarga de la ejecución del plan, es decir, publica cada acción del plan al resto de agentes. Cada acción del plan es una regla. Estas reglas, como se vio en el capítulo 3, están compuestas de dos partes. Su lado izquierdo es la condición inicial de la acción, un patrón en forma de subgrafo que existe en DSR. El lado derecho es el resultado. La diferencia entre

ambos subgrafos, indica la modificación necesaria, a realizar por el agente en su DSR, para completar la acción.

El nombre de la regla permite al agente conocer si está implicado en esa acción y que elementos necesita. Es decir, si por ejemplo, el agente detector de personas recibe la acción *detectPerson* la regla contiene los identificadores y nodos de DSR que le permiten ejecutarla. A partir de ese momento el agente -se podía decir que se "activa"- tratará de satisfacer las condiciones. Por tanto, el agente sabe cual es la salida de la acción y trata de conseguirlo mediante su computación local. Cuando lo consiga, publicará ese nuevo estado de DSR, que contiene el subgrafo descrito en el lado derecho de la regla.

El ejecutivo es el encargado de propagar los cambios en DSR, a todos los agentes, tras un cambio estructural. Técnicamente hablando, todos los agentes están suscritos a un tópico. El ejecutivo publica en ese tópico cada nuevo estado en DSR. Esta función ha sido realizada en el ejecutivo por razones de programación e integración en RoboComp [Manso et al., 2010]. Aunque puede ser traslada a la librería que implementa la estructura de DSR. Por las mismas razones, se han añadido dos funcionalidades extra al agente: la traducción de DSR al lenguaje de planificación y la validación del cambio estructural.

Por tanto, el agente Ejecutivo es el responsable de comprobar, que los cambios propuestos por el agente, son coherentes con la gramática y el estado actual en DSR. Es decir, el agente Ejecutivo comprueba que es posible alcanzar ese nuevo estado de DSR, de acuerdo a la gramática, validando, o no, la propuesta. En caso afirmativo publicará este nuevo modelo. Este tipo de comportamiento relativos al *model checking* también se puede encontrar en [Meijer, 2012], donde el lenguaje de planificación PDDL es combinado con una herramienta de *model checking* basada den grafos llamada GROOVE[3].

La limitación, al beneficio de la coherencia interna de DSR, viene derivada de la obligación de definir de antemano todo el conjunto de reglas gramaticales que codifican los cambios estructurales válidos que DSR puede sufrir.

A modo de resumen general, podríamos personificar las funciones del agente Ejecutivo en la arquitectura CORTEX en: el *guardián*, el *traductor* y el *administrador* de DSR. El primero de ellos, porque preserva o protege la coherencia del estado interno permitiendo sólo algunos cambios estructurales, el segundo, porque traduce la estructura a PDDL y el último, porque administra las acciones del plan -las reglas-, publicándolas al resto de agentes.

5.6. Agente Manipulación

Uno de los objetivos perseguidos por la robótica es conseguir dotar al robot del mayor grado de autonomía posible. Para ello se necesitan herramientas que nos permitan lidiar con la cinemática del robot. Si bien la cinemática directa no entraña demasiadas complicaciones, no ocurre lo mismo en el campo de la cinemática inversa, donde existen una variedad de propuestas que dependen de la naturaleza del problema. Probablemente, la distinción más clara para eligir la solución, sea si el robot se encuentra en un entorno fijo controlado o no. En cualquier caso, si se pretende que el robot interactúe con humanos, el método de cinemática inversa, necesita encontrar la solución en nuestra escala de tiempos. Es decir, el robot debe resolver el problema básico de la manipulación: ¿Cómo conseguir que un robot mueva su brazo para alcanzar una

[3] http://groove.cs.utwente.nl/

taza, un lápiz -o cualquier objeto o posición en el espacio-, en un tiempo y con una carga de cómputo razonable?

5.6.1. El núcleo: *Cinemática Inversa*

El núcleo del agente reside en los algoritmos de cinemática inversa. Estos han sido envueltos en un componente. Su objetivo es conseguir un método general, para que a partir de una cadena cinemática inicial -que representa, al robot o la parte del robot, que se quiere desplazar- y la pose final deseada del efector $P_e = [tx, ty, tz, rx, ry, rz]$, se devuelva la configuración final que lo satisface. Expresada como un *array* de motores, siendo cada motor un grado de libertad del robot, y cuyos valores representan el giro a realizar para alcanzar la posición final deseada.

Gracias a la información geométrica contenida en la representación interna, se puede acceder a la cadena cinemática que representa el robot o parte de él, como por ejemplo los brazos. De esta forma, además, se consigue que el mismo proceso sea independiente del modelo de robot, ya que su configuración esta descrita en DSR. También, DSR permite tener en cuenta el resto de elementos representados a la hora de encontrar la solución, que satisface la postura final deseada.

En la búsqueda de ese método ágil que nos permitiera abordar la tarea requerida, el elegido ha sido el algoritmo de Levenberg-Marquardt, sugerido originalmente en [Levemberg, 1944] y posteriormente en [Marquardt, 1963], es una técnica iterativa que encuentra un mínimo local de una función multivariable expresada como la suma de los cuadrados de las funciones de valores reales no lineales. El algoritmo Levenberg-Marquardt, ha demostrado ser uno de los de más éxito, debido principalmente a su facilidad de implementación y al uso de una estrategia de amortiguación eficaz, que le confiere la capacidad de converger rápidamente desde una amplia gama de conjeturas iniciales [Hiebert, 1981].

La estructura general de la implementación del algoritmo de Levenberg-Marquardt se basa en la propuesta de [Lourakis and Argyros, 2009]. La Figura. 5.7 muestra el pseudocódigo.

La articulación de un robot puede tener uno o varios motores. Por lo tanto, al calcular los incrementos para los ángulos de los motores pertenecientes a una cadena cinemática, se plantea un problema técnico, que escapa del campo teórico de la matemática para enfrentarnos al mundo real de la ingeniería, los motores pueden, y normalmente deben, tener unos límites máximos y mínimos de giro, que no se pueden sobrepasar. Es decir, los motores tienen un rango en el que pueden girar o moverse. Normalmente, estos límites son impuestos por razones físicas o de diseño, por ejemplo, el motor ubicado en el codo de un robot, sólo podría girar hasta colisionar con el antebrazo, o tal vez, el cableado le impida alcanzar un determinado ángulo, ya que puede ocasionar su rotura. Para estas y otras situaciones, es necesario que existan límites de giro.

Por tanto, y debido a que el algoritmo de Levenberg-Marquardt, calcula unos incrementos angulares para cada motor que luego serán añadidos a los ángulos originales. Entonces, ¿Qué ocurre cuando el algoritmo de Levenberg-Marquardt calcula unos incrementos que superan los límites mínimo y máximo de un motor?

Para afrontar esta situación se ha seguido el método propuesto por Baerlocher y Boulic en [Baerlocher and Boulic, 2004]. Este método propone añadir al de Levenberg-Marquardt, un bucle interno que se encargue de comprobar si algún ángulo calculado supera los límites del motor, o articulación que le corresponda y, de darse el caso, bloquear ese motor y eliminarlo de los cálculos.

Input: A vector function $f : \mathcal{R}^m \rightarrow \mathcal{R}^n$ with $n \geq m$, a measurement vector $\mathbf{x} \in \mathcal{R}^n$ and an initial parameters estimate $\mathbf{p}_0 \in \mathcal{R}^m$.
Output: A vector $\mathbf{p}^+ \in \mathcal{R}^m$ minimizing $||\mathbf{x} - f(\mathbf{p})||^2$.
Algorithm:
$k := 0$; $\nu := 2$; $\mathbf{p} := \mathbf{p}_0$;
$\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_\mathbf{p} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
stop:=$(||\mathbf{g}||_\infty \leq \varepsilon_1)$; $\mu := \tau * \max_{i=1,\ldots,m}(A_{ii})$;
while (not stop) and $(k < k_{max})$
  $k := k + 1$;
  repeat
    Solve $(\mathbf{A} + \mu\mathbf{I})\delta_\mathbf{p} = \mathbf{g}$;
    if $(||\delta_\mathbf{p}|| \leq \varepsilon_2(||\mathbf{p}|| + \varepsilon_2))$
      stop:=true;
    else
      $\mathbf{p}_{new} := \mathbf{p} + \delta_\mathbf{p}$;
      $\rho := (||\epsilon_\mathbf{p}||^2 - ||\mathbf{x} - f(\mathbf{p}_{new})||^2)/(\delta_\mathbf{p}^T(\mu\delta_\mathbf{p} + \mathbf{g}))$;
      if $\rho > 0$
        stop:=$(||\epsilon_\mathbf{p}|| - ||\mathbf{x} - f(\mathbf{p}_{new})|| < \varepsilon_4||\epsilon_\mathbf{p}||)$;
        $\mathbf{p} = \mathbf{p}_{new}$;
        $\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_\mathbf{p} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
        stop:=(stop) or $(||\mathbf{g}||_\infty \leq \varepsilon_1)$;
        $\mu := \mu * \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$; $\nu := 2$;
      else
        $\mu := \mu * \nu$; $\nu := 2 * \nu$;
      endif
    endif
  until $(\rho > 0)$ or (stop)
  stop:=$(||\epsilon_\mathbf{p}|| \leq \varepsilon_3)$;
endwhile
$\mathbf{p}^+ := \mathbf{p}$;

Figura 5.7: Pseudocódigo del algoritmo Levenberg-Marquardt [Lourakis and Argyros, 2009].

La adaptación y desarrollo de los algoritmos matemáticos mencionados, condujeron por una parte a la creación y desarrollo de nuevos componentes dentro de RoboComp [Manso et al., 2010]. Los cuales son el núcleo de este agente. Por otra parte, y no menos importante, a la realización y defensa del Trabajo fin de grado *Cinemática Inversa en Robot Sociales* que fue calificado con la máxima nota y, del cual, el autor de esta Tesis es co-director.

5.6.2. El concepto de *emulación interna*

Como se explicó anteriormente para el agente de navegación 5.1.2. El esqueleto del sistema para la "navegación" del brazo, es el mismo que para la base, ya que la idea futura es que ambos sean fusionados en uno solo. De forma que no haya que subdividir la tarea en subpartes, es decir, el objetivo es conseguir que mientras el robot se desplaza vaya acomodando su postura a la acción de manipulación. Por ejemplo, mientras el robot se acerca a una mesa para coger una taza, acompase el movimiento de sus brazos -o el cuerpo entero-, junto con el desplazamiento de la base.

De momento, en un paso a previo a incorporar la base como un grado de libertad más, se ha expandido el espacio de configuración para el brazo robótico. De esta forma, se comprueba la validez de la propuesta cuando extendemos el espacio de configuración, a los siete grados de libertad del brazo.

Sin olvidar que el futuro de la HRI exige que las acciones repetitivas mejoren con el tiempo. Si el robot coge una taza de la misma mesa varias veces, la persona que está con él, esperará que el robot reduzca su tiempo de ejecución a un periodo humano -aprenda y mejore-. Hacer lo

contrario, probablemente, repercutiría en la confianza del humano acerca de la utilidad de la interacción con el robot. Para tratar de conseguirlo, se adopta una solución similar a la expuesta en la sección 5.1.2, se añaden los grados de libertad del brazo robótico.

La principal diferencia radica en la construcción del espacio de configuración libre *free C-space*. En este caso, en lugar de un grafo de posiciones aleatorias sobre el espacio libre, se usa una rejilla regular -*grid*- tridimensional del espacio de trabajo del brazo. Cada elemento en ese volumen codifica una posición euclidiana de seis dimensiones para la postura de la mano y el conjunto de configuración del *free C-space* que permite alcanzarla. Este espacio tridimensional es el espacio de trabajo del brazo robótico. Esta rejilla discretizada, como veremos en la sección 6.3, no es una limitación en el sentido de que el objetivo tenga que encontrarse en una de esas posiciones para poder ser alcanzado, si no que al contrario, permite hacer un movimiento rápido del efector hasta una de estas posiciones para, a continuación, proceder con un último estadio de refinamiento visual que permita alcanzar el objetivo, con la orientación y posición correcta.

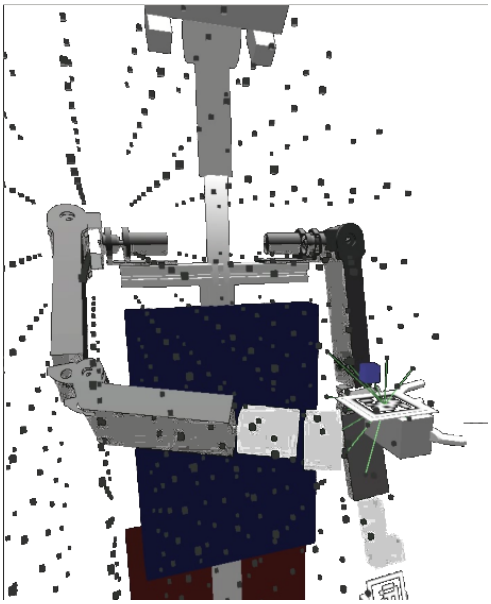


Figura 5.8: *Free C-space* para el brazo.

Pero antes de continuar, es necesario definir el concepto de *emulación interna* ya que es una propiedad muy interesante de DSR, y este agente muestra claramente sus beneficios. Aunque es extensible al resto de agentes. El concepto es sencillo, el agente usa su copia local para emular la acción, y así concluir cual es la decisión óptima para alcanzar el objetivo. Es decir, el agente utiliza su copia de la representación de la realidad para generar una percepción simulada del resultado de la acción. El uso de la palabra emulación significa aquí, que esa copia local puede ser adulterada con elementos que no estaban presentes en la realidad, por ejemplo, el espacio libre de configuración que ha sido representado como cubos de color negro o el cubo lila que representa la posición final que debe alcanzar el efector.

Entonces, si la acción del robot es alcanzar la posición del cubo lila. El concepto de *emulación interna* permite ejecutar el proceso completo internamente. Usando el espacio de configuración libre creado anteriormente. Se elige un camino libre de colisiones con el entorno y con el robot mismo, y que asegura que la postura del brazo, no sobrepasa los límites físicos de

ningún motor, siguiendo una ruta a través de los cubos negros. La *emulación interna* permite, por tanto, realizar predicciones a corto plazo y evaluar las consecuencias de las acciones a un nivel situacional o sensomotor.

5.7. Agente Propiocepción

La propiocepción en pocas palabras, es el sentido que permite percibir nuestro estado interno. Es el *sentido* que informa al organismo de la posición de los músculos, es la capacidad de sentir la posición relativa de partes corporales contiguas. La propiocepción regula la dirección y rango de movimiento, permite reacciones y respuestas automáticas, interviene en el desarrollo del esquema corporal y en la relación de éste con el espacio, sustentando la acción motora planificada.

Uno de las propiedades claves del concepto de *Deep State Representation* es la representación interna del robot y su entorno, este agente es el exponente de esa filosofía -desde el punto de vista geométrico-. Su función es la de mantener constantemente actualizada toda la estructura del robot.

El robot más completo de todos los robots disponibles en RoboLab, es Shelly 6.3. El grafo que lo define internamente está formado por más de 100 nodos enlazados mediante enlaces geométricos, etiquetados con RT.

El bucle general de funcionamiento de este agente consiste en: obtener en cada ciclo el valor de todos los motores del robot real y compararlos con los valores de la lectura anterior. Por razones de eficiencia computacional, solo se actualiza cuando existe una variación apreciable entre las lecturas, o cuando el periodo sin actualizar algún valor en la representación es considerable. El umbral elegido para actualizar el valor de la representación, al valor real del motor, ha sido de medio grado. Por razones históricas la posición de la base es actualiza en DSR mediante el agente navegación, lo que puede ser considerado como una falta de coherencia en la arquitectura. Aunque a efectos prácticos es irrelevante. Lo verdaderamente importante es mantener la representación del robot actualizada, ya que se puede obtener información extra, gracias a los enlaces no explícitos e inherentes a las propiedades de las cadenas cinemáticas.

Una pregunta interesante que ejemplifica de manera clara y sencilla este agente es: ¿Es necesario incluir la información de los datos en crudo de los sensores en DSR? Blumenthal et. al. [Blumenthal et al., 2013], ya se planteaba esa cuestión. Argumentaba que esos datos debían ir en las hojas de su grafo. La mayoría de nuestro equipo de investigación -incluidos el autor de esta Tesis-, eramos reticentes a esta idea, o en honor a la verdad, nos parecía que sería sobrecargar el sistema. Demasiada información, demasiado ancho de banda moviéndose entre DSR y los agentes, tal vez, la estructura y con ella la arquitectura entera podrían colapsar. Pero pensando en un sistema ideal de recursos ilimitados, tenemos que reconocer que no es mala idea. Por ejemplo, ahora hay una abstracción entre la nube de puntos proporcionada por un dispositivo Kinect, y las apenas catorce coordenadas que simbolizan a un humano en DSR. La razón es entendible porque la nube de puntos es pesada. Pero tal vez, esos datos en crudo, podrían ser relevantes a un tercer agente que los necesitase y no tendría porque implementar una conexión con otro componente, simplemente obtenerla en DSR. Como ocurre con el agente Propiocepción, los datos del motor, son los datos que se están almacenando en DSR. Se están compartiendo con el resto de agentes. Generando buenos resultados en la operatividad de la arquitectura. Por ejemplo, el agente de manipulación visto en la sección 5.6, los utiliza, para

calcular la postura del robot más adecuada.

Por razones de la tecnología actual, no estamos seguros de que podamos incorporar los datos a las hojas de DSR. Aunque si que estamos más cerca de estar de acuerdo con Blumenthal en que es una buena idea. La solución propuesta para salvar las posibles limitaciones tecnológicas y -que se explica en más detalle en los trabajos futuros 8, es incluir en esa hoja un *proxy* al componente que proporciona los datos. De forma que el agente pueda acceder a los datos del dispositivo físico, sin que estén realmente en DSR. Y sin que el agente necesite especificarlo en el momento de su creación o ser regenerado si los requiere.

5.8. Agente Detector de Objetos

El agente detector de objetos implementado en CORTEX es muy limitado. El campo de la detección de objetos es una de las áreas más activas de la robótica y los retos que plantea son innumerables.

Por el momento y hasta contar con un sistema robusto, tal vez, con la inclusión de los trabajos de algún especialista en el agente. La arquitectura cuenta con un agente perceptor de objetos que usa AprilTags [Olson, 2011]. Asocia a distintos objetos del entorno una marca única de AprilTags. Cuando la marca es percibida, se reconoce como ese objeto. En ese momento un nuevo nodo es añadido y ubicado, en un lugar preciso dentro de DSR. Además, el nodo, contiene una relación de atributos asociados con el objeto. Entre ellos destaca la representación tridimensional asociada en forma de malla. Esto permite, además de una visualización representativa para el desarrollador, la capacidad de que pueda ser usada para evitar colisiones con el entorno.

Los beneficios de usar una marca AprilTag son varios. Permite evitar confusiones en la detección de los elementos. Ofrece una precisión suficiente en la obtención de su posición y orientación en el espacio. Aspecto crucial en la estructura DSR. Al mismo tiempo permite avanzar en la tarea de la coordinación de actividades, en la manipulación de objetos y en su representación. Una vez inserta en DSR permite ser usada por el resto de agentes. De esta forma conseguimos completar la misión o tarea, añadir nuevos objetos y gestionar todo el proceso de una forma sencilla y ágil. Cuando la robustez y capacidad de un futuro perceptor de objetos universal, sea un hecho. Simplemente se deberá sustituir la fuente de datos de entrada del agente. Reemplazando el perceptor de AprilTag por el nuevo perceptor de objetos.

5.9. Agente Misión

El agente misión es un agente especial que se encarga de activar la misión del robot y permite monitorizar el desarrollo de la misma. El dilema en el desarrollo de la robótica viene muchas veces, en relación a las herramientas. Básicamente hay una difícil decisión, en cual es la cantidad de tiempo y esfuerzo que se debe dedicar a las herramientas -que facilitan el desarrollo de las tareas-, y cuanto al diseño, estudio, creación y desarrollo de nuevos algoritmos. Extendiéndolo, a la investigación pura.

Al igual que en otros frameworks robóticos [Quigley et al., 2009]. RoboComp [Manso et al., 2010] ha decidido dedicar bastantes de esos esfuerzos al desarrollo de herramientas simplificadoras de los futuros desarrollos. El autor de esta Tesis ha dedicado mucho esfuerzo a que este agente especial diera un paso más hacia su usabilidad. Como se ha

explicado la estructura DSR se encuentra en todos los agentes, por tanto, también está en el agente misión.

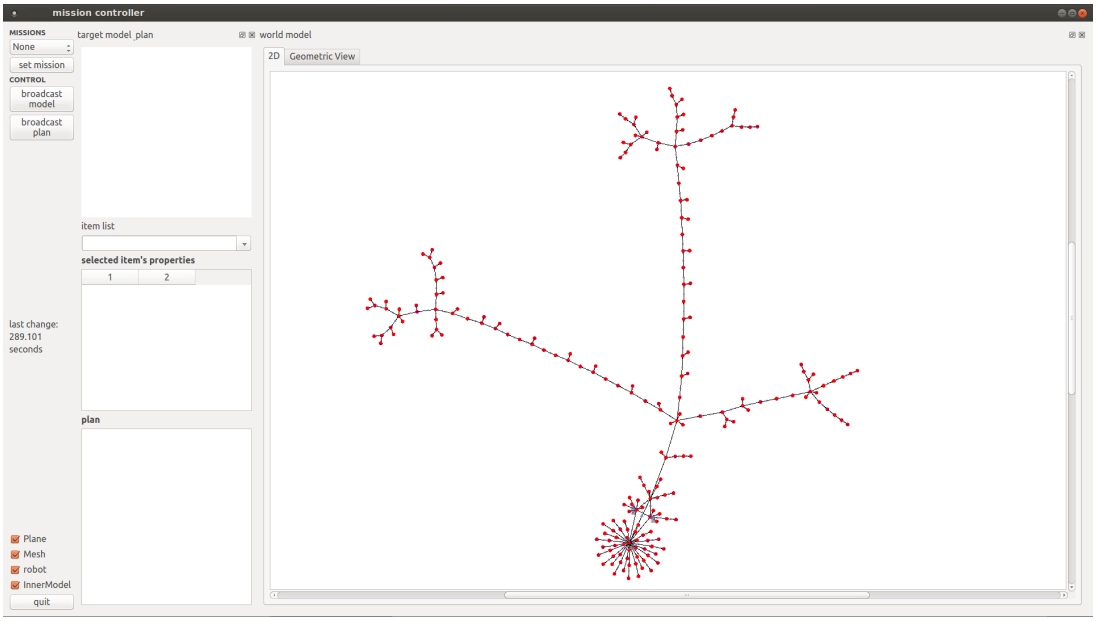


Figura 5.9: Agente misión. Vista bidimensional al inicio de una misión.

La vista bidimensional del grafo implementada en este agente -Figura. 5.9-, permite visualizar y casi materializar el concepto de este grafo unificado dentro de la arquitectura. Lamentablemente, arroja la reveladora verdad, de que es difícil de comprender. Obviamente, la parte geometrica del robot comprenderá la mayor parte de los nodos y arcos. Pero al menos, se puede afirmar que para un mismo robot no crecerá en exceso. Por tanto, y como la idea es que las tareas sigan aumentando en complejidad, implicará un aumento del resto de nodos y arcos. Sin duda, esto hará aún más difícil comprender la información del estado interno del robot y el entorno. Es decir, el concepto de DSR, se vuelve demasiado complejo de comprender para el desarrollador ¿de que le sirve, tener esta traza del plan, sí no puede comprender que está ocurriendo internamente entre los agentes?

De otra parte, la inclusión en la representación de elementos en la escena, gracias a reconocedores de objetos, detectores de personas o la creación y actualización de mapas, conlleva una carga cognitiva, pero a su vez también una carga geométrica para representar esos nuevos elementos. Por ejemplo, detectar a un humano, como se vio, simbólicamente hablando, consistía en crear un enlace entre la habitación y la persona e etiquetarlo con in. En cambio desde el punto de vista geométrico consistía en la inclusión de tantos nodos como articulaciones, sus arcos y sus mallas 3D asociadas.

En este contexto, y con la intención de acelerar los tiempos futuros del desarrollo de las tareas, se ha tomado la decisión de implementar en el agente, la vista geométrica a través de una representación tridimensional mediante un motor gráfico. En este caso el elegido como motor gráfico, es el conocido *Open Scene Graph*. Reafirmando, una vez más, la elección del grafo como estructura de representación de la creencia del robot. La Figura.5.10 muestra la representación tridimensional.

La vista geométrica de la representación ofrece la posibilidad de entender de un vistazo, fácilmente, que está sucediendo internamente, o porqué el robot se está comportando de una u

otra manera en la realidad.

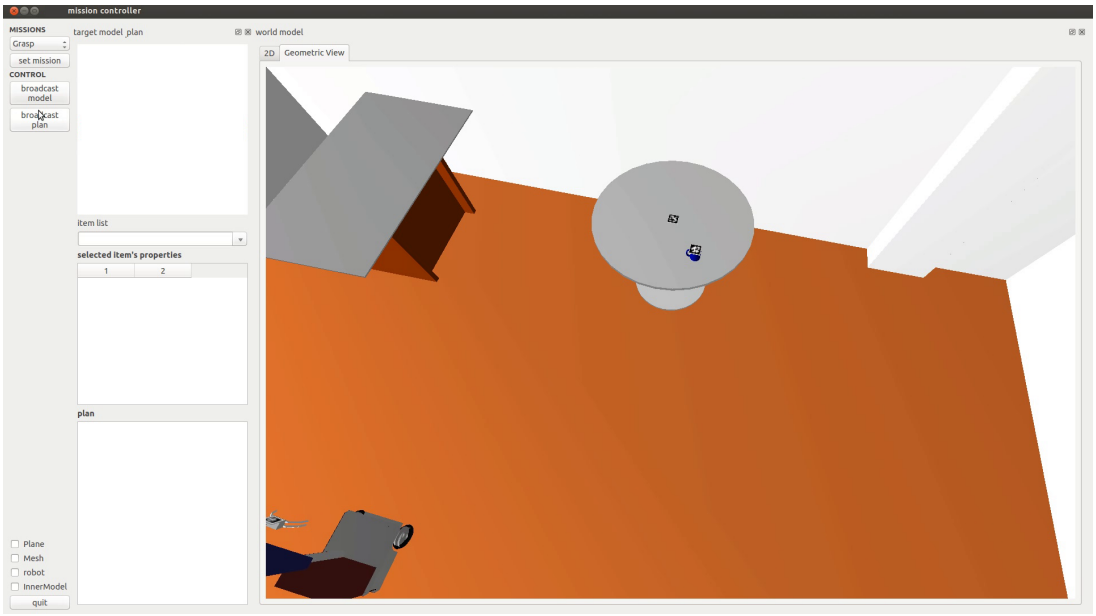


Figura 5.10: Agente misión. Vista tridimensional al inicio de una misión.

5.10. Conclusión

El objetivo del capítulo ha sido ejemplificar lo extensible de la arquitectura CORTEX y DSR, y que el lector comprenda la facilidad de incorporar, de una forma elegante y transparente, el trabajo de otros grupos o especialistas dentro de la arquitectura en la forma de agentes. De otra parte, esto favorece el trabajo distribuido y la cooperación entre grupos de investigación.

La morfología y funcionalidad del agente desarrollado hasta ahora, cumple el rol de un ente que envuelve el trabajo de un especialista y que proporciona información a la representación. A partir de este momento, comienzan a volverse nítidos, otros tipos de de agentes que trabajen solo con la representación, infiriendo propiedades y relaciones de la realidad representada. Por ejemplo, entre los objetos y el robot. Imaginemos un agente que pueda ir relacionando, que objetos están cerca o son alcanzables por el robot, mientras este se mueve por la habitación. O incluso otros que sean capaces de deducir conceptos como encima, debajo o al lado.

La emulación interna, sin duda, se presenta como una característica muy relevante de la arquitectura CORTEX y DSR, ya que dota a los agentes de la capacidad de emular acciones en su copia local, trasladando a la representación compartida solo los resultados adecuados de la acción.

Obviamente, si la emulación se prolonga demasiado en el tiempo, puede afectar al resto de la arquitectura. Pero es interesante pensar, en los posibles beneficios de la predicción en un espacio corto de tiempo. Si se pudiera pre-ejecutar, cada mínima acción internamente, a un nivel sensomotor, sin pérdida para el resto del sistema, existiría una validación adicional que dotaría de mayor robustez al sistema.

Parte IV

Escenarios experimentales

Capítulo 6

Escenarios experimentales

Este capítulo pretende mostrar, por un lado, la evolución de nuestros robots en estos años y la evolución del software que los gobierna. Por otro, mostrar un experimento final de una tarea compleja, traer una taza, realizaba por un robot manipulador móvil.

En la sección 6.1. Se muestra a Ursus -en sus distintos diseños-. Su objetivo era comprobar, si un robot social podría ayudar, al paciente, en la adherencia en los procesos de rehabilitación. Al mismo tiempo que ayudar al terapeuta en la monitorización del paciente.

En la sección 6.2. Se muestra a Gualzru, un robot vendedor, cuya misión es convencer a potenciales clientes para que le acompañen a un stand publicitario. Al mismo tiempo que se trataba de comprobar, si causa o no rechazo, o inseguridad, o si es percibido como social. En su última versión software incorpora DSR y CORTEX.

En la sección 6.3. Se muestra como DSR ayuda y simplifica, una tarea compleja, que requiere un traspaso de información fluido y coordinación entre agentes. La tarea es traer una taza.

Para finalizar 6.4 se discute y analiza, desde la perspectiva del software que los gobierna, lo expuesto en el capítulo.

6.1. Desde Ursus a Therapist

El robot Ursus, ha ido evolucionando en sus distintas versiones [Naranjo-Saucedo et al., 2011] hasta llegar a Therapist [Calderita et al., 2014b]. Ursus fue uno de los primeros robot en abandonar RoboLab para ser usado en un entorno real -lo que siempre supone un desafío tecnológico-. Concretamente en un entorno clínico y con el objetivo de ayudar en la neuro-rehabilitation motora de pacientes. La Figura.6.1, muestra su último diseño y una captura en una sesión piloto en el Hospital.

Este proyecto se realizó conjuntamente con el Hospital Universitario Virgen del Rocío (HUVR). El aspecto exterior y el tamaño del robot tienen un papel muy importante en este tipo de entornos. Para conseguir que tuviera un aspecto amigable. El robot estaba cubierto con el tejido de la piel de un oso de peluche. Era un robot semi-autónomo equipado con un sensor RGB-D que registraba los movimientos del paciente. Ursus disponía de dos brazos, ambos con cuatro grados de libertad (DOF), acoplados a un torso fijo.

Los pacientes observaban como Ursus realizaba el ejercicio de rehabilitación motora y trataban de imitarlo. Disponía de unos altavoces en la base del robot que animaban, con mensajes

Figura 6.1: (Izquierda) El robot Ursus. (Derecha) URSUS dirigiendo una sesión de rehabilitación en el HUVR[Calderita et al., 2014b].

de apoyo, al niño durante la sesión. Para conseguir un mayor realismo el movimiento de la boca se sincroniza con el audio. La boca estaba compuesta de tres piezas planas de aluminio y un motor. La pieza de aluminio superior estaba fija, mientras que el labio inferior es movido por el motor. La Figura. 6.2 muestra en detalle la boca robótica.

La amplitud del movimiento de la boca, fue ajustado para oscilar entre 0 y 45 grados. El algoritmo descrito en [Cid et al., 2011] es el encargado de sincronizar el movimiento. La idea básica del algoritmo, es estimar el grado de apertura, en base a la entropía del audio sintético generado por un sistema *Text-To-Speech* (TTS).

Figura 6.2: Vista detallada de la boca robótica [Cid et al., 2011].

El objetivo principal del proyecto fue mejorar la adherencia a las terapias de neuro-rehabilitación, ya que tradicionalmente son repetitivas y tediosas. El resultado es una sesión aburrida, día tras día. Este aspecto reduce la motivación y adherencia al tratamiento limitando los beneficios para los pacientes. Por tanto, el objetivo era conseguir que las sesiones fueran más amenas, sin perder el rigor de la terapia.

Además, las sesiones aburridas repercuten más negativamente en un niño que en un adulto, lo que aumenta aún más su desacoplamiento con el tratamiento. Sin embargo, la ejecución de estos movimientos repetitivos es realmente necesaria, ya que altera las propiedades de nuestras neuronas, incluyendo su patrón de conectividad. Conducido correctamente, este proceso finalmente permite mejorar la funcionalidad neuronal. Por tanto, la cuestión era: ¿Cómo podemos mejorar la motivación y la inmersión de los pacientes en la terapia? La solución fue, convertir la terapia aburrida en una divertida, que ayudará al niño y al terapeuta.

Para este fin, las tecnologías asistidas por ordenador se han utilizado ampliamente en la última década. Se propuso la inmersión del niño en un escenario de realidad aumentada 6.3.

Este tipo de juegos son conocidos como *Serious Games* y han demostrado sus beneficios en el área de la rehabilitación [Rego et al., 2010]. Además, la adherencia a la sesión aumenta gracias a la incorporación de nuestro robot social como un compañero de ejercicios. Ursus podría ser visto como un entrenador personal, o un compañero de juegos, que lleva a cabo la sesión en el mundo real con el paciente[Calderita et al., 2013a]. Para ayudar al terapeuta, el robot era capaz de registrar los datos de los movimientos del paciente en cada sesión. Estos podían ser utilizados posteriormente por los especialistas en rehabilitación para el seguimiento y/o la adaptación de la terapia a las necesidades del paciente.

Figura 6.3: (Izquierda) Un paciente en una sesión de rehabilitación con Ursus (Derecha) El juego de realidad aumentada usado al mismo tiempo.

Desde la primera versión de Ursus, el robot mostró que la predisposición del paciente en su tratamiento neuro-rehabilitación se puede mejorar con su incorporación. Uno de las principales carencias del robot era que para operar necesitaba estar bajo la supervisión de un ingeniero. Lo que invadía el espacio clínico y la intimidad de la sesión, sin incorporar ningún beneficio. Aunque, no fue objeto de la investigación, para nosotros supuso entender que la arquitectura de control necesitaba ser redefinida, sobre todo, si se quería hacer frente a escenarios diferentes y más complejos. Así que, había que dirigirse hacia arquitecturas de control más adaptables. La arquitectura de control original de Ursus, codificada cada sesión terapéutica como una máquina de estados jerárquica fija. Así, era difícil hacer frente a nuevas situaciones, o incorporas variantes fácilmente, ya que era preciso reescribirla una y otra vez.

6.1.1. Therapist: La evolución de Ursus

En estos dos artículos [Calderita et al., 2015, Calderita et al., 2014b] se detalla la evolución de Ursus hacia Therapist un robot socialmente interactivo, aunque las fechas de publicación describen una línea temporal distinta, lo cierto es que el orden correcto es, en primer lugar el artículo [Calderita et al., 2015] que fue publicado en Enero de 2015, y en segundo lugar el artículo [Calderita et al., 2014b] que fue publicado en Octubre de 2014, la realidad es que a veces los plazos de revisión se prolongan.

Siguiendo esta correcta línea temporal describiré lo más notable de esta evolución de la arquitectura software e incluiré algún dato relevante sobre los resultados y evolución del robot. Lo más importante del primer artículo, desde el punto de vista de esta Tesis, es la inclusión,

de lo que podía considerarse, el germen de la arquitectura actual CORTEX, y a la que llamamos RoboCog [Bustos et al., 2013], se conseguía integrar diferentes modalidades sensoriales a través de las denominadas *CompoNet* -una red de componentes-, y se conseguía una coordinación básica a través de un componente *Ejecutivo*. Todo había sido implementado dentro del framework RoboComp. Aunque no como estructura compartida, se incluye una representación interna del modelo geométrico -en forma de árbol cinemático-. El modelo incluye al robot y al niño. Este es usado de forma local por una de las *CompoNet*. El objetivo era que con su incorporación, el software podía exhibir un comportamiento más pro-activo, al ser más consciente de su entorno.

De otra parte, se concluía que se necesitaba una nueva versión hardware de Ursus que acabaría desembocando en Therapist. Fundamentalmente se necesitaban construir unos brazos más robustos que pudieran soportar un hipotético experimento a largo plazo dentro del Hospital. Lo que no fue posible debido a razones económicas.

No obstante la Figura. 6.4 muestra como eran. A parte de su construcción se desarrollaron componentes que los controlaban. Afortunadamente el trabajo no fue en balde y estos mismos son ahora parte del robot Shelly6.3.

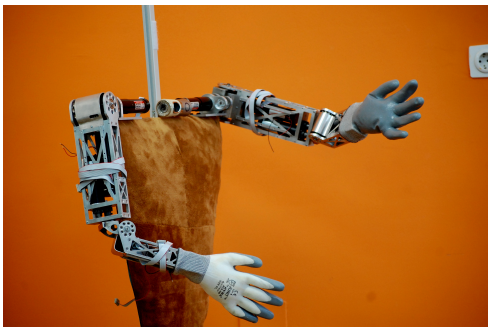


Figura 6.4: Brazos construidos para Therapist

Aunque el pilotaje no llego nunca ha realizarse, el desarrollo del robot Therapist fue completo. El robot superó con éxito un experimento piloto que contó con más de 35 niños generando el segundo de los artículos mencionados anteriormente [Calderita et al., 2014b].

Su principal evolución en cuanto a la arquitectura software es la inclusión de, la recién nacida, AGM [Manso, 2013]. Lo que le permitía usar funciones de alto nivel, como por ejemplo, la inclusión de componentes a cargo de la planificación. Además le permitía tener una representación simbólica del entorno.

El robot cambia de aspecto, se refuerzan algunos de los módulos y se generan otros nuevos. El objetivo del robot sigue siendo el mismo: exhibir un comportamiento que refuerce la adherencia al tratamiento. En esta ocasión, el ejercicio elegido por el hospital Universitario Virgen del Rocío, era distinto y exigía, un hardware más potente, y nuevos módulos software.

El juego de rehabilitación se denominó: *touch the draw*. Consistía en que el robot tocara un dibujo -una marca de AprilTags [Olson, 2011]-. El dibujo era mostrado por el niño y el robot trataba de tocarla. El robot no solo interactuaba de esta manera, sino que mantenía una conversación simple y restringida al entorno. A la vez que expresaba un conjunto básico de emociones, gracias a una *tablet* colocada a modo de cara expresiva. La Figura. 6.5 muestra el robot y una secuencia del juego.

El juego, consiste en una simple interacción entre un humano y un robot. Esto nos permi-

Figura 6.5: El robot Therapist jugando a *touch the draw* [Calderita et al., 2014b].

tió evaluar sus capacidades sociales. El robot se presenta y le pregunta al niño si quiere jugar a un juego. Tras la aceptación, el niño muestra un dibujo al robot. El robot comienza a seguirlo, dirigiendo su mirada hacia el dibujo. El seguimiento de la marca es realizado mediante el sensor RGB-D colocado en la cabeza. La cabeza dirige la mirada hacia la marca gracias mediante cinemática inversa. El dibujo puede escapar del campo de visión, debido a que, el robot está fijo y solo puede mover el cuello y la cabeza. También puede ser ocultado por el niño tras su espalda, en estas situaciones el robot pedirá al niño que se lo muestre de nuevo. Tras una indicación verbal, el robot intentará alcanzar el dibujo acercando su mano hasta él. Si lo logra emitirá una frase de alegría. Tras una pequeña pausa de cortesía, colocará su brazo en una posición de reposo. El robot anima al niño a mover el dibujo durante el transcurso de la partida, e incluso, a ocultarlo.

La figura 6.5 muestra algunas instantáneas de la sesión. El robot no usa ninguna carcasa específica. Aunque el aspecto mecánico podría sesgar la percepción del robot de los niños hacia una entidad artificial en lugar de hacia una entidad social, la aceptación fue general [Calderita et al., 2014b].

El juego fue evaluado inicialmente en el laboratorio y luego se probó con niños dentro de la población objetivo -niños entre 4-9 años de edad-. En pocas palabras, el objetivo era poner a prueba la interacción niño-robot y comprobar las actitudes del robot. Este experimento se llevó a cabo en julio de 2014. La población total consistió en 35 niños -16 niñas y 19 niños-. La edad media fue de 7,37 años -SD 1,47-. Los participantes fueron reclutados en una escuela de verano. Su profesor acompañaba al niño a la zona de juegos. Solamente con la instrucción de jugar con el robot durante un rato. En el área de juego, el robot se presentaba y el juego comenzaba. Las sesiones se grabaron en vídeo para ser analizadas posteriormente. Sin embargo, la interacción robot-niño se analizó principalmente a través de un cuestionario, que era rellenado por el niño inmediatamente después de haber jugado con el robot. Un segundo cuestionario evaluando el comportamiento del robot fue rellenado por un observador externo, normalmente la madre del niño.

Nuestra hipótesis de partida es que los pacientes de pediatría podrían incrementar su inmersión en la terapia a través de la una interacción, no necesariamente física, con un robot. A corto plazo, esto facilitará el diseño de nuevas terapias que deberían mejorar la motivación del paciente al enfrentarse a las tediosas terapias de rehabilitación. Sin embargo, en este trabajo, nos centramos en el estudio y análisis de la predisposición de los niños a aceptar el robot como

un elemento de la terapia y a entenderlo como una entidad social. Esperábamos que los niños disfrutaran jugando con el robot y quisieran repetir la experiencia.

Ambas cuestiones son básicas para el establecimiento de un vínculo social entre el robot y el niño. Este vínculo, es el que permitirá al robot, ayudar al niño a alcanzar con éxito, los objetivos de las terapias de rehabilitación a largo plazo. Por su puesto, que para conducir esta interacción de forma eficaz, el robot debe ser capaz de emanar respuestas en la escala de tiempos humana. Por lo tanto, nosotros también usamos el experimento para estudiar una segunda hipótesis, si con la explotación de las posibilidades RoboCog, estábamos más cerca de que THERAPIST consiguiera involucrarse interactivamente en el proceso de rehabilitación del niño.

Por tanto, en el artículo [Calderita et al., 2014b], mediante el uso de cuestionarios y a través de la anotación semántica manual de la grabación de los vídeos, se pretende demostrar dos hipótesis:

La primera, si los niños perciben al robot como una entidad social, más que como una simple máquina. Para ello se utilizó un cuestionario. Además de la evaluación cuantitativa de los datos de vídeo. Esto permite la evaluación de la respuesta emocional y el comportamiento interactivo de los niños.

La segunda, comprobar el comportamiento del robot y su actitud. Se utilizó otro cuestionario diseñado para ser contestado desde el punto de vista de la persona que observa el comportamiento del usuario frente al robot.

Me gustaría señalar aquí los resultados del cuestionario Q2, ya que está directamente relacionado con la evaluación del comportamiento general, y esto, en definitiva, depende de su arquitectura software. El resto de cuestionarios y resultados se encuentran en [Calderita et al., 2014b]. El cuestionario Q2, consta de cuatro bloques, cada uno de los cuales incluye dos preguntas. Estos cuestionarios evalúan el comportamiento y naturalidad del robot -velocidad de respuesta, movimiento natural, etc- de la interacción. Las preguntas fueron contestadas en una escala *Likert* de 5 puntos. La Tabla. 6.1 resume los resultados. Se concluye que la interacción entre el robot y el niño es por lo general fluida, y todos los canales -verbales y no verbales- eran apropiados. Cabe señalar, desde el punto de vista software, que el robot no se quedó bloqueado durante las sesiones. El robot estuvo funcionado sin interrupción durante dos mañanas, con una duración aproximada de cuatro horas cada una.

	Media	**Desviación estándar**
Are the robot's movements natural?	4.66	0.40
Has the child stepped away from the robot?	0.23	0.55
Have you understood what the robot told the child?	4.76	0.43
Does the robot understand the child?	4.20	0.72
Does the robot get blocked?	0.00	0.00
Was the interaction fluent?	4.00	0.69
Do you think the child enjoyed the experience?	4.52	0.56
Do you think the child wants to repeat the experience?	4.41	0.55

Cuadro 6.1: Media y desviación estándar para el cuestionario Q2 [Calderita et al., 2014b].

En resumen, la hipótesis inicial fue validada por el estudio experimental. Se concluye que la interacción con el robot, es positiva y esa actitud colaborativa podía mejorar los resultados de los pacientes de pediatría.

También se verificó que RoboCog permite al robot interactuar con los pacientes a una escala tiempo de humana. Sin embargo, en el camino hacia el comportamiento pro-activo y autónomo del robot, se necesita que este sea más consciente de sí mismo y de su entorno. El desarrollo de nuevas terapias de rehabilitación sin intervención requerirá la conjunción de muchas líneas de investigación, aún en desarrollo. Esta conclusión, se podría extender al siguiente robot. Esperamos que la propuesta de esta Tesis contribuya al desarrollo e integración de las diferentes líneas de investigación.

Por otra parte, y desde el punto de vista del comportamiento humano, se comprobó, que, debido a la tendencia inherente de los seres humanos a personificar objetos animados [Feil-Seifer and Matarić, 2011], cuando las habilidades del robot no satisfacen completamente las expectativas, se genera cierto grado de frustración. Por ejemplo, cuando el robot no entendía bien a un niño o cuando la trayectoria de la mano para tocar el dibujo era poco natural.

Por último, era necesario tener en cuenta una serie de consideraciones acerca de la apariencia física del robot, ya que estudios recientes han demostrado, por ejemplo, que el tamaño tiene un impacto considerable en la percepción de sus habilidades de conducta e interacción [Feil-Seifer and Matarić, 2011].

El proyecto terminó y ahora el testigo ha sido depositado en otros miembros del consorcio. La Universidad Carlos III ha sustituido a Ursus por el famoso robot Nao[1], generando muy buenos resultados. Desde un punto de vista personal, es una satisfacción que esta línea de investigación continúe abierta, ya que es un campo muy prometedor. Por otra parte, y desde el mismo punto de vista, la experiencia de trabajar con niños, junto con la de acercar el mundo de la Universidad a la vida real, serán un recuerdo imborrable.

6.2. Gualzru: el vendedor robótico

El robot Gualzru ha sido desarrollado dentro del proyecto ADAPTA [2]. En su última versión incorpora DSR y la arquitectura CORTEX. El desarrollo de Gualzru ha estado íntimamente ligado al desarrollo de la propuesta de esta Tesis.

La misión de Gualzru es esperar al lado de un stand publicitario interactivo, mientras observa a la gente pasar y, en algún momento, selecciona a un candidato y se acerca a él para iniciar una conversación. Mediante un pequeño diálogo, tratará de convencer al candidato para que le acompañe de vuelta al stand publicitario. Una vez allí dejará el resto del proceso de venta al software interactivo incrustado en el stand.

El aspecto exterior de Gualzru se ha cuidado mucho. La Figura. 6.6a muestra el prototipo. Se puede observar que el diseño es simple, seguro y elegante, evitando elementos afilados o piezas articuladas. Incluso se han ocultado las ruedas dentro de la carcasa de fibra de vidrio.

La Figura 6.6b, muestra su configuración hardware interna. Básicamente está compuesto por un router que conecta diferentes elementos de procesamiento. Dos ordenadores integrados. Uno de ellos, encargado de procesar los datos visuales y de audio, utiliza el sistema operativo Windows. Este equipo está conectado al sensor Kinect y al micrófono empleado para capturar el audio.

El otro ordenador embebido utiliza el sistema operativo Ubuntu y ejecuta el resto del software. Está conectado al dispositivo láser empleado para la navegación, a los motores de la base

[1] https://www.aldebaran.com/en

[2] Innterconecta Programme 2011 project ITC-20111030 ADAPTA

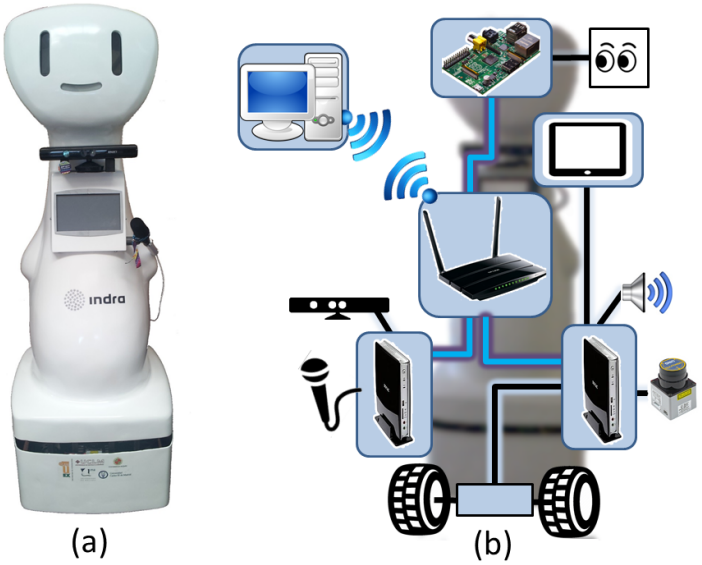


Figura 6.6: (a) Gualzru, el vendedor robótico; y (b) Estructura hardware.

y a los altavoces. Todo está a bordo de una evolución de la plataforma robótica *-open-hardware-* Robex [Mateos et al., 2010]. Es una plataforma diferencial. Las ruedas motrices están colocadas en la parte delantera y una rueda giratoria -o rueda loca- en la parte posterior. Las ruedas tienen un radio lo suficientemente grande para absorber las irregularidades del suelo -en entornos de interior-. A la vez, proporcionan una buena transferencia de potencia y velocidad. Los motores están equipados con codificadores incrementales ópticos para proporcionar una odometría aceptable. Aunque finalmente, ha tenido que ser reforzada con un sencillo localizador basado en marcas de AprilTags [Olson, 2011].

Hay dos elementos más incluidos en el robot. Uno de ellos es un dispositivo Raspberry Pi, que se encarga de controlar el movimiento de los ojos y el nivel de la batería. El otro es una pantalla táctil instalada en el pecho del robot -incorporada en la última fase del proyecto-. Esta pantalla se utiliza para visualizar las frases pronunciadas por el robot y para proporcionar una alternativa al canal auditivo, permitiendo una comunicación más robusta con el humano.

La siguiente sección describe la evolución de Gualzru. Los problemas y las necesidades que surgieron en su desarrollo, han sido una fuente más de inspiración en la propuesta de esta Tesis.

6.2.1. Evolución del robot Gualzru

En cuanto a la evolución de Gualzru nos centraremos en su evolución software siguiendo la evolución del proyecto ADAPTA. Tras un acuerdo conjunto, entre los grupos del consorcio incluidos en el proyecto, este se pretendía usar como una prueba para la propuesta inicial de la Universidad de Extremadura: organizar toda la arquitectura del software en torno a un modelo interno y centralizado del mundo exterior. Esta propuesta, surge tras comprobar lo estático y complicado de adaptar el comportamiento del primer robot, Ursus, a otros escenarios y robots. Los componentes relacionados con el Hardware eran sencillos de reutilizar. En cambio, los encargados del alto nivel o del procesado intermedio, no.

El proyecto ADAPTA planteaba un escenario realista -aunque controlado- para probar la idea inicial. Inicialmente, contábamos con la arquitectura RoboCog, la cual disponía de una

representación interna del robot en forma de árbol cinemático, llamada InnerModel. Era una representación geométrica que servía también para iniciar el simulador 3D de RoboComp, llamado RCIS.

Por lo tanto, InnerModel era el punto de partida perfecto para desarrollar la idea de una representación compartida del robot y su entorno. Tanto InnerModel como RCIS eran gradualmente extendidos para soportar más tipos de objetos, al mismo tiempo que, se desarrollaban funcionalidades útiles, como transformaciones de coordenadas, cálculos métricos, operaciones para el manejo en la inserción, modificación y eliminación de nodos, funcionalidades de visualización, simulación de elementos hardware, cambios de perspectivas, etc.

Para dotar de acceso al resto de componentes, a una estructura compartida que representase al robot y al mundo, se creó una interfaz para el simulador RCIS que permitía acceder a InnerModel. Al extender las capacidades del simulador, y dotarlo de acceso en caliente a las funciones de InnerModel, cualquier componente podía conectarse, compartir y mantener allí, su representación del mundo.

Sin embargo, dentro del proyecto ADAPTA, había problemas en la gestión del alto nivel. En un principio, la codificación del caso de uso, fue implementada en una máquina de estados finita. Pero pronto esta opción se volvió impracticable. El número de estados y transiciones, crecía cada vez que se suavizan las restricciones del caso de uso. Las restricciones se suavizaban con el objetivo de hacerlo más flexible y acercarlo un poco más a la realidad.

Para solucionarlo, se decidió mejorar el control del alto nivel recurriendo a un sistema de planificación simbólica. Por tanto, había que incorporar toda la información simbólica necesaria por los elementos deliberativos, a la arquitectura RoboCog. Los requisitos eran, que la percepción y la acción debían actualizar un conjunto fijo de atributos simbólicos y predicados, y que la representación seleccionada se pudiera traducir -de manera eficiente- a PDDL. Para poder usar el *framework* de planificación PELEA [Alcázar et al., 2010]. Esto permitía aprovechar el conocimiento y trabajo de otros miembros del consorcio, ya que eran especialistas en este área y esta tecnología. Nuestra elección sobre cómo proceder no fue exactamente un error, pero sin duda, estuvo cerca.

Se decide integrar, *Active Grammar Modelling* [Manso, 2013], ya que permitía manejar una representación simbólica del mundo. Esto repercute en que la representación del mundo se duplicaba. En consecuencia, el aspecto simbólico y geométrico del robot y del entorno, habitan en dos representaciones diferentes no conectadas. La integración de ambas no era sencilla. AGM era reciente y estaba en desarrollo. No obstante, ya era capaz de mantener correctamente la representación simbólica y convertirla a PDDL.

Se decide mantener esta configuración por razones operativas -plazos y auditorías del proyecto-. Pero, las dificultades e incoherencias de la arquitectura, se empiezan a manifestar internamente, tanto desde el punto de vista software, como dentro del equipo. No obstante, y desde un punto de vista externo, el robot Gualzru avanza en su desarrollo. Supera los hitos propuestos y consigue realizar su tarea como vendedor. Además, es probado y evaluado con éxito, en los diferentes escenarios del proyecto [Romero-Garcés et al., 2015]. Por tanto, como mencionamos anteriormente, fue un error de diseño, en el sentido de que la división generó otros problemas, pero no en el sentido ingenieril, ya que permitió cumplir con el proyecto. La siguiente lista sintetiza estos problemas:

1. La complejidad al actualizar y acceder a ambas representaciones por separado.

2. La dificultad en el mantenimiento de una división clara entre ambos niveles de abstracción.
3. La dificultad en el mantenimiento de un estado interno común y coherente cuando es necesario acceder a ambas estructuras simultáneamente.
4. No se logra un acceso eficiente a los datos cuando las consultas necesitan acceder a -y en ocasiones esperar la respuesta de- ambas representaciones.
5. La dificultad al introducir nuevos conceptos, o nuevos atributos a los existentes, si estos tienen una naturaleza simbólica y geométrica.

Todos estas deficiencias, no hicieron otra cosa que reforzar nuestra idea de DSR, como solución elegante, coherente y con futuro. De hecho, se podía decir que se estaba empezando a diseñar la arquitectura CORTEX -basada en los principios cognitivos como dijimos- entendiendo el proceso cognitivo, como la habilidad que nos permite tratar internamente con la información sobre nosotros mismos y el mundo exterior. Claramente, esta habilidad está sujeta a la existencia de una representación interna activa que gestione toda esta información. La existencia de una *Deep State Representation*, del mundo exterior y del propio robot, es el concepto clave de CORTEX.

6.2.2. Experimentos con Gualzru

Gualzru ha sido evaluado, con éxito, en varias ocasiones dentro del proyecto, y ha completado decenas de horas de pruebas en entornos reales, interactuando con personas que no conocían previamente el sistema. Los resultados de estas pruebas a lo largo del proyecto se han ido utilizando para detectar deficiencias, corregirlas y mejorar en definitiva el sistema hasta llegar a su versión final [Romero-Garcés et al., 2015].

A modo de síntesis, incluiremos aquí la prueba más relevante realizada en diciembre de 2014. La Figura.6.7 muestra varias imágenes mostradas durante la prueba. El robot fue evaluado en un escenario real de trabajo. Todos los elementos del proyecto[3] fueron desplegados en el hall de la Escuela de Ingeniería en la Universidad de Málaga. La zona en la que el robot trabajaba era de unos 70 metros cuadrados. Había obstáculos fijos, como una columna y algunas mesas, pero la mayoría de la zona era libre y amplia para que el robot se moviese fácilmente. El recinto estaba lleno de estudiantes. Las pruebas duraron dos días y medio. El robot trabaja sin intervención humana y salía al encuentro de la gente que pasaba cerca. Estas personas no tenían *a priori* ningún conocimiento sobre el robot ni su funcionalidad.

Cincuenta personas interactuaron con el robot. Al terminar se les pidió rellenar un cuestionario. Este estaba diseñado siguiendo la escala *Likert*, aunque utiliza seis niveles, de 0 a 5, para eliminar la opción neutral -punto medio-. Es similar a la empleada por Joosse et al. [Joosse et al., 2013] para generar la base de datos BEHAVE-II. Su principal diferencia es que no se ha creado desde el punto de vista de la persona que observa el comportamiento del usuario frente al robot, sino desde el punto de vista del mismo usuario que interactúa con el robot. En este sentido, podemos considerar que recoge influencias de los cuestionarios del modelo original *Almere* [Heerink et al., 2010]. En particular, el cuestionario incluye una colección de preguntas organizadas en cuatro bloques: navegación, conversación, interacción y sensación ge-

[3] ADAPTA también incluía un holograma interactivo

Figura 6.7: Imágenes tomadas durante los experimentos realizados en la Universidad de Málaga

Cuadro 6.2: Resultados de los 50 cuestionarios [Romero-Garcés et al., 2015].

Question	$\bar{x}$	σ
Navigation		
1.1 *Do you feel safe when the robot approaches you?*	4.31	0.95
1.2 *Does the robot invade your personal space?*	0.96	1.37
1.3 *Do you think robot movements are natural?*	2.62	1.23
1.4 *Have you stepped away from the robot?*	0.96	1.46
Conversation		
2.1 *Have you understood the robot?*	3.57	1.28
2.2 *Has the robot understood you?*	2.70	1.30
2.3 *Was the conversation coherent?*	2.96	1.38
2.4 *Do you like the voice of the robot?*	3.13	1.29
Interaction		
3.1 *Did the robot get blocked?*	1.39	1.72
3.2 *Was the interaction natural?*	3.11	1.11
3.3 *Was the conversation fluent?*	2.85	1.22
3.4 *Did the robot seem to be tele-operated?*	0.87	1.44
Overall sensation		
4.1 *Did you enjoy the experiment?*	4.31	0.88
4.2 *Do you think the exp. was not interesting?*	0.70	1.32
4.3 *Would you like to repeat?*	4.28	1.32
4.4 *Would you recommend it to other people?*	4.52	0.86

neral. El usuario rellena el cuestionario dando un valor para cada respuesta entre 5 -totalmente de acuerdo- y 0 -totalmente en desacuerdo-. Estas preguntas se enumeran en la tabla 6.2.

Estos datos muestran que el sistema de conversación es el punto débil del robot. Algunas personas no entendieron correctamente el robot debido al ruido ambiente. La voz del robot parece no demasiado agradable. Pero la cuestión más importante se relaciona con la capacidad de comprensión de Gualzru. Incluso, cuando se utiliza un micrófono de tipo *shotgun* -diseñado para estas situaciones-, la comprensión auditiva se mantiene limitada. Aunque mejora respecto a las versiones anteriores de Gualzru en la que utilizábamos el *array* de micrófonos del dispositivo Kinect [Romero-Garcés et al., 2015]. El sistema es muy sensible al ruido ambiental y al eco. Además su rendimiento decae cuando hay varias personas que hablan alrededor del robot. Esta situación es más común de lo esperado debido al interés que el robot produce. Otras cuestiones, como los diferentes acentos, los volúmenes de voz, etc. Añaden aún más dificultades a la situación. A pesar de estas habilidades de conversación limitada, Gualzru logró su principal objetivo -desde el punto de vista del marketing-, captar la atención de la gente. La mayoría de ellos disfrutaron de la experiencia, la recomendarían a amigos y les gustaría repetirla.

No obstante, se recuerda que el propósito de Gualzru es: *conseguir un si o un no como respuesta de un cliente potencial tras un breve diálogo*. Si la respuesta es afirmativa, el robot conducirá al cliente hasta el stand publicitario. En caso negativo, Gualzru se despedirá amablemente de la persona y buscará un nuevo candidato.

Por tanto, si la habilidad de conversación se ve comprometida, todo el sistema se ve comprometido. La solución a estos problemas fue la inclusión de una una pantalla táctil en el torso del robot que reforzase el proceso de conversación. La pantalla muestra las frases al mismo tiempo que el robot las dice. Esto facilita su entendimiento por parte de los usuarios, sobre todo en entornos bulliciosos. También permite al usuario, elegir la respuesta tocando en en la pantalla. Esto consigue que la comunicación entre ambos mejore, ya que existen dos vías de interacción, la pantalla táctil y la voz.

En este punto me gustaría argumentar que ocurriría sino se dispusiera de una representación compartida. La funcionalidad de la pantalla sería la misma y es obvio el refuerzo al sistema de conversación. Pero desde el punto de vista ingenieril, imaginemos un sistema sin representación y orientado a componentes. Añadir está nueva interfaz de datos requeriría alguna de las siguientes opciones: (a) Incrementar las funcionalidades del componente encargado del proceso de conversación. Para ello habría que modificarlo e incluir acceso al hardware de la pantalla. (b) Crear un nuevo componente con que se encargará de acceder a la pantalla táctil y conectarlo con el componente a cargo de la conversación.

Cualquiera de las opciones, también implica, la modificación de la lógica de control del componente de conversación, ya que debería ser modificado para tener en cuenta las nuevas entradas y salidas.

Si se añadiera una tercera fuente de información para el proceso de conversación, por ejemplo, la detección del gesto afirmativo con la cabeza de la persona. Una nueva conexión con el componente de conversación debería ser creada.

Este tipo de incremento en la información desde fuentes heterogéneas, escala con facilidad usando DSR y CORTEX. El proceso sería crear un agente a cargo de la pantalla táctil. Cuando Gualzru dice una frase, la escribe simultáneamente en un nodo de la representación. El agente a cargo de la pantalla lee el mismo nodo y la muestra. La repuesta es escrita y propagada a través de DSR, lo que puede provocar, que se ejecute la siguiente acción del plan. Lo realmente interesante de este enfoque, es que se puede mejorar todo el sistema sin apenas apenas modificar el resto de la arquitectura.

Una implementación inicial de la arquitectura CORTEX y DSR, han sido incorporadas a Gualzru, con el objetivo de validarla experimentalmente y comprobar los beneficios de esta integración. Los resultados de la experiencia se han recogido en el artículo, aún no publicado, *The cognitive architecture of a robotic salesman*. Para comprobarlo, treinta tres personas accedieron a rellenar un cuestionario tras haber realizado el caso de uso con Gualzru. El diseño del cuestionario era el mismo que la vez anterior. La Tabla. 6.3 muestra las dos preguntas relacionadas directamente con el proceso de comprensión entre el humano y el robot. Las columnas más a la derecha muestran los resultados antes de incluir DSR y CORTEX y la pantalla táctil. Las columnas más a la izquierda muestran los resultados tras incluirlos.

Por tanto, DSR representa la percepción de la realidad del robot, pero ahora el mismo estado -la respuesta del humano-, puede ser alcanzado desde dos fuentes diferentes. Además el resto de agentes de la arquitectura son conscientes de eso y pueden actuar en consecuencia. Con lo cual es posible añadir fácilmente nuevas entradas y salidas. Por ejemplo, permitir respuestas usando el lenguaje corporal o los gestos, mediante la implementación de nuevos agentes a cargo de las

Question	$\bar{x}$	σ	$\bar{x}_{prev}$	σ_{prev}
2.1 Have you understood the robot?	4.27	1.23	3.57	1.28
2.2 Has the robot understood you?	3.72	1.28	2.70	1.30

Cuadro 6.3: Detalle de los resultados para los 33 cuestionarios.

mismas partes del grafo. En consecuencia, y sin modificar la implementación de los agentes, se refuerza todo el sistema y se mejora el contexto de la acción en el resto de agentes.

6.3. Shelly: un robot manipulador móvil autónomo

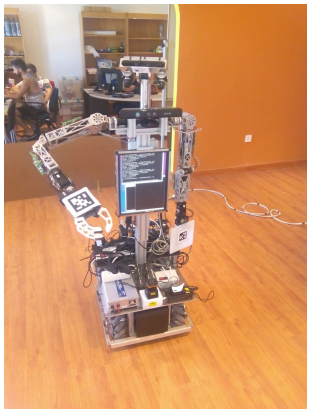
Shelly -Figura. 6.8a- es un robot antropomórfico construido en *RoboLab*, en la Universidad de Extremadura. Es un robot social diseñado para ayudar en las tareas domésticas. Se compone de una base omnidirecional, dos brazos, cada uno con 7 grados de libertad y dos dedos en forma de pinza -Figura. 6.8d. Tiene una cámara RGB-D encima de una estructura que le permite el giro sobre los tres ejes. Esta cámara se usa para la detección de objetos en el entorno. Tiene otra camará RGB-D en la parte superior del torso. Su cometido es detectar y capturar el movimiento de las personas. También dispone de un sensor láser para la navegación.

El propósito del experimento es describir una tarea compleja, pero muy común en nuestras vidas: El robot debe recoger un objeto de una mesa -una taza- y dársela a la persona. El experimento valida el trabajo interno de CORTEX y DSR, ya que la tarea requiere la interacción de las habilidades de planificación, percepción, manipulación, agarre, navegación y monitorización. Dentro del contexto de un robot móvil manipulador autónomo.

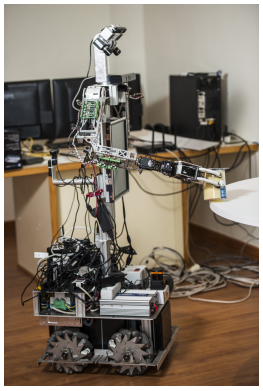
Los siguientes agentes han sido implementados y desarrollados para el experimento:

- *Executive* coordina las acciones del plan y comprueba la validez de los cambios estructurales en DSR, basándose en el conocimiento del dominio. Si el cambio estructural es aceptado el nuevo DSR es publicado.
- *Planner* es responsable de la planificación de alto nivel de la misión. Integra las habilidades de planificación y replanificación, monitorización y aprendizaje.
- *Person detector* detecta y captura el movimiento de las personas.
- *Navigation*, su cometido es desplazar al robot y ubicarlo en el entorno.
- *Object detector* detecta la orientación y la posición de los objetos.
- *Proprioception* mantiene actualizada la representación interna del robot con la información sensorial de su postura.
- *Grasping* coordina y ejecuta la acción de agarre de los objetos.

El experimento ha sido restringido a las funcionalidades actuales de los agentes del robot. Por ejemplo, usamos marcas de AprilTags [Olson, 2011] para detectar de forma fiable los objetos. La Figura. 6.8c, muestra la habitación con una mesa etiquetada con una marca de AprilTags

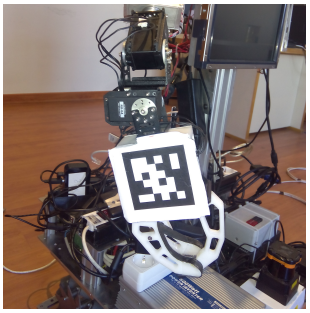


(a) Vista general de Shelly.



(b) Shelly cogiendo la "taza".

(c) La mesa y la "taza". Las marcas de AprilTag indentifican y ubican los objetos.



(d) Vista de la muñeca y dedos. La marca ayuda en la localización de la muñeca del robot.

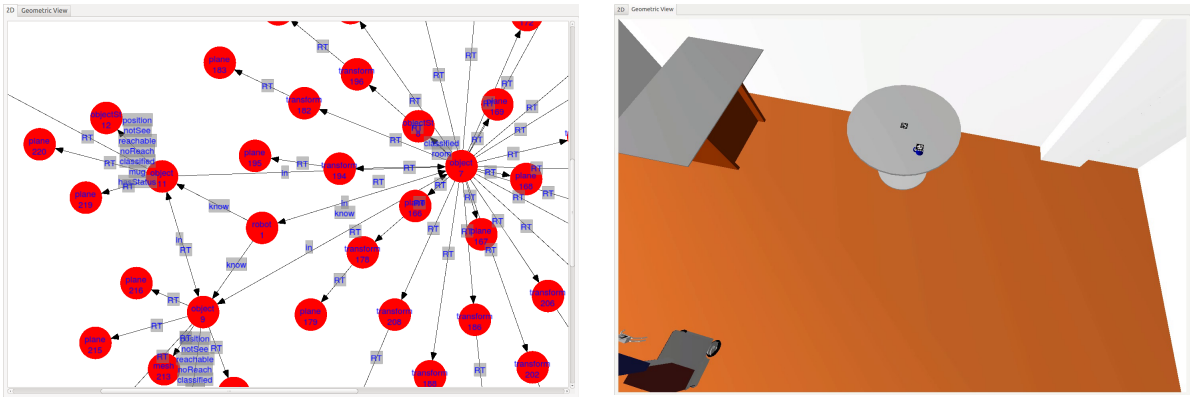
Figura 6.8: El robot móvil manipulador en *RoboLab*.

en su centro. Sobre la mesa hay un objeto cilíndrico -una taza- también etiquetado con una marca. Este objeto puede ser cogido y levantado por el robot sin forzar su hardware -Figura. 6.8b-.

La misión del robot comienza tras recibir una orden verbal del humano. Seguidamente se desplazará hacia la mesa, recogerá la taza y se la llevará a la persona. Antes de empezar DSR incluye una serie de elementos. Estos elementos representan el conocimiento *a priori* del entorno. La Figura. 6.9 muestra el estado inicial. Los nodos y arcos geométricos incluyen las paredes, la mesa, la taza y la estructura de Shelly. Existe también un obstáculo fijo en el entorno, una mesa sin marca. Este objeto no puede ser distinguido por el agente *Object detector*, pero si puede ser usado por el agente *Navigation* para la auto-localización del robot. De la misma forma que es esquivado cuando el robot navega.

Como muestra la Figura. 6.9b, el robot sabe que hay una mesa y una taza en la habitación, pero, por ejemplo, la posición de la taza en DSR es diferente a la de su posición real -Figura. 6.8c-, debido a que aún no ha sido detectada por el agente *Object detector*.

Como mencionamos anteriormente, el experimento comienza cuando el robot recibe una orden verbal del humano. Una vez que la misión está activa el agente *Planner* recibe un patrón objetivo y el estado actual de DSR. Tras componer un plan, como una secuencia de acciones o reglas, que transformarán el estado actual de la representación al estado deseado descrito en el patrón. El agente *Executive* enviará la primera regla al resto de agentes. La Figura. 6.10 muestra la regla.



(a) DSR vista 2D.

(b) DSR vista geométrica.

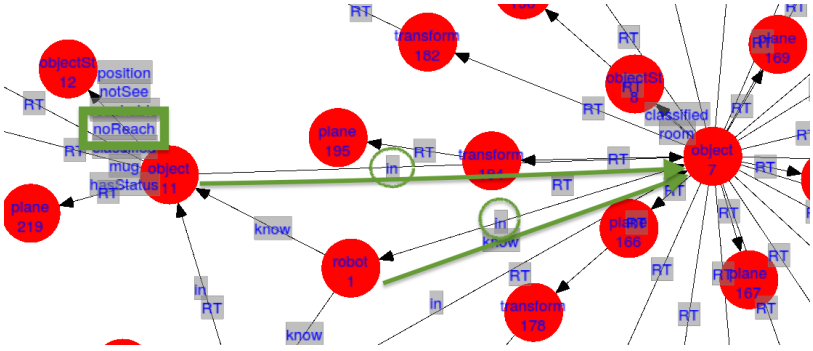
Figura 6.9: DSR visto en detalle. Vista bidimensional y geométrica.

Esta regla describe las condiciones del estado de DSR -Figura. 6.10a- y las operaciones a realizar cuando se cumplan -Figura. 6.10b-. La operación es sencilla. El agente *Grasping* comprueba DSR. Usa la información geométrica para calcular la distancia entre la posición del robot -actualizada por el agente *Navigation* mediante cambios no estructurales- y la posición de la taza -actualizada por el agente *Object detector* mediante cambios no estructurales-. Cuando la distancia es adecuada, el agente *Grasping* evolucionará el estado en DSR cambiando la etiqueta *noReach* por *reach*.

Se observa que el lado izquierdo de la regla -Figura. 6.10a-, es un patrón que ya existe en el grafo -Figura. 6.10c-, esto conseguirá que los agentes trabajen para satisfacer el lado derecho de la regla -Figura. 6.10b-. Al mismo tiempo el agente *Object detector* y el agente *Person detector* pueden estar actualizando DSR incluyendo nuevos objetos en el entorno.

(a) Lado izquierdo.

(b) Lado derecho.



(c) Solo se dibuja el lado izquierdo de la regla en DSR para una mejor visualización.

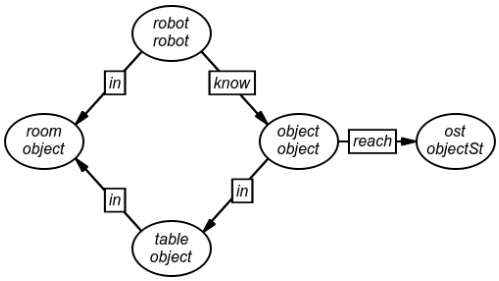
Figura 6.10: Regla: *SetObjectReach*. El agente *Navigation* acerca el robot a la taza y el agente *Grasping* modifica la etiqueta *not reach* por *reach* cuando el robot esta a la distancia adecuada.

Una vez que el robot está cerca de la mesa, la siguiente regla es lanzada -Figura. 6.11-. El robot está listo para la tarea de agarre. Además, el agente *Grasping* ya ha calculado el *grid*

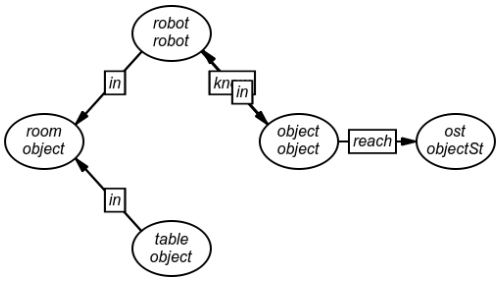
tridimensional que codifica el espacio alcanzable por el brazo. El agente comprobará diferentes rutas hacia el objetivo -el cubo lila en la Figura. 6.12a-, y selecciona el cubo negro más cercano. Por tanto, DSR permite realizar predicciones a corto plazo, así como, evaluar las consecuencias de las acciones a un nivel situacional o sensomotor.

Esos cambios no afectan al resto de agentes y nos sirven para introducir el concepto de *internal emulation*, en un caso real. El concepto es sencillo de explicar, el agente usa su copia local para emular la acción, y así concluir cual es la decisión óptima para alcanzar el objetivo. Es decir, el agente utiliza su copia de la representación de la realidad para generar una percepción simulada del resultado de la acción. El uso del término emulación significa aquí, que esa copia local puede ser extendida con elementos que no estaban presentes en la realidad. Por ejemplo, el espacio alcanzable del brazo -los cubos negros-, o el cubo lila que representa la posición final que debe alcanzar el efector. El concepto de *internal emulation*, permite ejecutar localmente el proceso completo en DSR, antes de ejecutarlo en la realidad.

Por otra parte, los cubos negros representan la cinemática directa de esa posición. Se codifican como un vector de los valores de los motores para esa pose. Esto sirve para conseguir una respuesta más rápida, un sacádico del brazo, que en el caso que nos ocupa, conseguirá colocar el efector dentro del campo de visión de la cámara RGB-D de la cabeza.



(a) Lado izquierdo.



(b) Lado derecho.

Figura 6.11: Regla: *graspObject*. El agente *Grasping* en coordinación con el agente *Object detector* son los actores principales de esta acción.

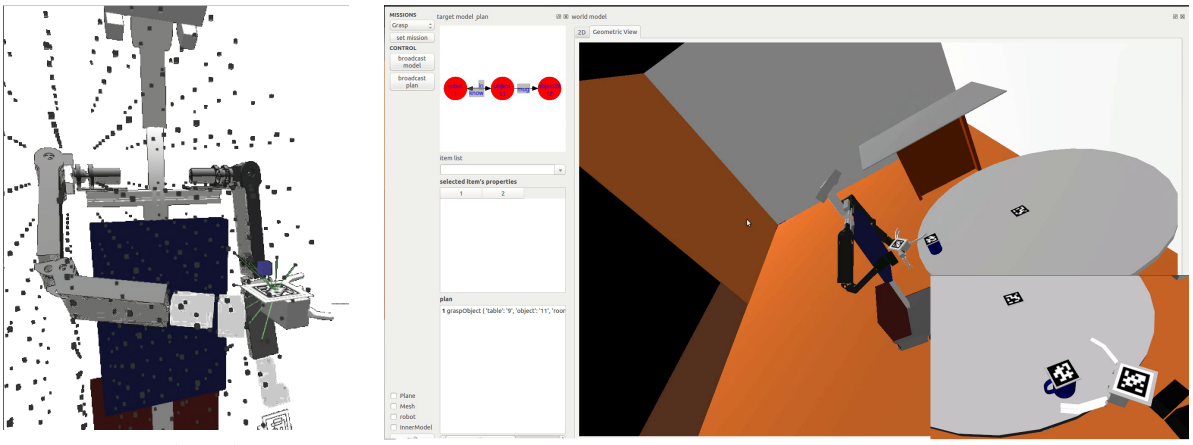
Debido a holguras, motores baratos o errores de calibración, es muy probable que el brazo entero tras este movimiento sacádico termine en una posición diferente a la esperada. Desafortunadamente, atendiendo al valor almacenado en los motores, el agente *Grasping* no puede saberlo. La pose en la realidad era una cadena de valores y esos valores están actualmente en los motores. No puede comprobar, lo correcto o incorrecto, de su pose sin otra fuente de información.

Afortunadamente, el agente *Object detector* está actualizando en DSR las posiciones de las marcas de AprilTags, referenciadas en el sistema de coordenadas del mundo. Al mismo tiempo, el agente *Proprioception* ha actualizado la pose del robot en el mismo sistema de coordenadas. Por lo que el agente *Grasping*, en caso de error entre la posición final de la muñeca y la percibida, dispone de un marco común para corregirlo. Para ello usará la cinemática inversa, enviando la muñeca a la posición de la marca almacenada en DSR. Esta acción moverá el brazo de Shelly en la realidad. Acción que será reflejada inmediatamente por el agente *Proprioception* en la representación unificada, lo que permitirá cancelar el error entre ambas percepciones y, también permitirá al agente *Grasping* continuar con la ejecución de la acción del plan. Es decir, como el agente aún no ha terminado la acción *graspObject*, no puede publicar un cambio estructural,

porque no satisface la regla. Es este periodo entre cambios estructurales o estados del grafo, el que permite a los agentes cooperar para realizar la acción. Usando su propia percepción y acción, junto a la información contenida en DSR -en estos periodos, el trasiego de información suele ser esencialmente geométrico-.

Es también, gracias a la combinación de los procesos anteriores, realimentación visual, corrección externa, realimentación interna, como se pretende conseguir un ajuste fino, que permite a la marca de la muñeca alinearse con la marca de la taza. Esta vez, usando técnicas de *visual servoing*. El agente *Grasping* dispone de toda la información necesaria en su DSR. Las posiciones de las marcas son mantenidas por el agente *Object detector*. Con esta información ordena los movimientos del brazo en la realidad. Hecho, que a su vez, provoca la actualización del brazo en la representación mediante el agente *Proprioception*. Este ciclo se repite hasta alcanzar la posición deseada.

Una vez que la mano alcanza la posición correcta, la pinza se cierra y coge la taza y, acto seguido, el agente actualiza en DSR el cambio estructural definido en la regla -Figura. 6.11-. La operación consiste en eliminar el arco etiquetado con *in*, que ubicaba al objeto taza en la mesa, y crearlo entre el objeto taza y el robot.



(a) *Internal emulation*.

(b) DSR después de la ejecución en la realidad.

Figura 6.12: Primer movimiento hacia una posición cercana a la taza. En la esquina inferior derecha se muestra una vista sintética desde la cámara RGB-D.

Con la taza cogida y DSR actualizado, el robot entregará la taza al humano. El agente *Person detector* es el encargado de la detección de personas en los alrededores del robot, así como, de actualizar su localización en DSR. El agente *Navigation* se dirigirá hacia la posición de la persona. Cuando el robot este cerca de la persona, le ofrecerá la taza -Figura. 6.13-. Tras una orden verbal de la persona, el caso de uso finalizará, forzando al robot a soltar la taza.

6.4. Conclusiones

En gran medida, la propuesta de *Deep State Representation*, se propone como mejora o solución, a los problemas, las dificultades encontradas y los errores cometidos a lo largo de estos años.

Como se ha comprobado, estos trabajos han generado contribuciones científicas en los campos donde se aplicaban. Aunque el objetivo de estas contribuciones era comprobar si el robot

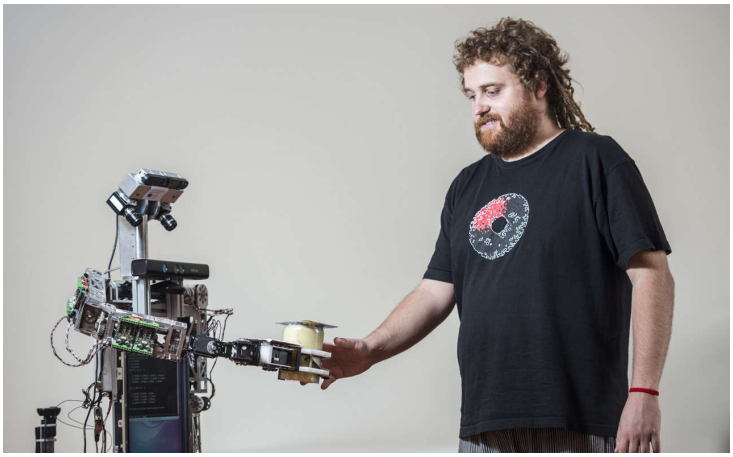


Figura 6.13: El autor recibiendo la “taza”.

podía ser adecuado para las distintas tareas, éstas han reforzado la idea de que el robot diseñado o construido sólo para un propósito, y con un software que no le permite desarrollar o incorporar nuevas tareas fácilmente, tiene poco recorrido.

De la misma forma, nos ha llevado a advertir que cuando la tarea precisa de una tecnología que excede a lo que puede ser controlado por una única persona, son necesarios el trabajo colaborativo por distintos equipos, así como, “hablar” un lenguaje común. En nuestra propuesta, ese lenguaje es DSR, el cual, permite la fluidez y comunicación entre las partes.

Además, nos ha dirigido hacia la opinión, -y digo opinión porque, *a priori* y sobre el papel, el software siempre puede ser realizado de otra forma-, de que las arquitecturas basadas en componentes alcanzan un punto de colapso. No colapsan en el sentido de que la arquitectura no pueda, en teoría, resolver un problema en cuestión, sino desde el punto de vista de que se hace inmanejable para el desarrollador o el equipo de desarrolladores.

No podemos afirmar, que con una cantidad ilimitada de recursos y personal sea posible. Pero lo que si podemos intuir por la experiencia de los trabajos aquí expuestos, es que el uso de esta estructura central que sirve para representar al robot y su mundo, al mismo tiempo que puede ser usada para razonar, simplifica el trabajo de los programadores. Les permite desarrollar de forma semi-independiente la tarea a su cargo. Les brinda una información extra generada por terceras partes, que muchas veces sería descartada al no existir una conexión directa entre componentes. Y, además, tiene el valor añadido de que la regla define, de forma general, la programación local del agente facilitando su implementación. Sin olvidar, que la información proviene del propio agente o está en DSR. Sigue unas reglas de creación y de relación, lo que permite ser usada y modificada, con conocimiento de causa, por todos y cada uno de los agentes.

Por último, me gustaría terminar el capítulo con una reflexión y pregunta abierta.

Como se comentó en la sección sobre el robot Gualzru, concretamente en la sección 6.2.2, se ha trabajado mucho en la mejora de su capacidad de reconocimiento del habla, ya que era su punto más débil. Un mes antes, -noviembre de 2014- de la prueba -diciembre de 2014- en la Universidad de Málaga, que concluyó la necesidad de reforzar el sistema de conversación con una pantalla táctil, el software a cargo de esta tarea ganó la *Functionality Benchmark Best Team (Speech understanding)* en el RoCKIn14 [4]. Era el mismo software, pero las condiciones del entorno eran diferentes, provocando que el sistema fallase. Ambos escenarios eran complica-

[4] http://rockinrobotchallenge.eu/rockin2014.php

dos, pero no dejaban de ser una situación real y posible. Por supuesto, estamos muy orgullosos del premio del Rockin14. Porque muestra la capacidad del sistema y premia el trabajo realizado para mejorarlo y adaptarlo a cualquier entorno. Pero la experiencia nos ha enseñado, que la robustez del algoritmo en una situación, se puede ir al traste con cualquier imprevisto de la realidad. Lo hemos sufrido con la navegación, la detección de objetos y personas, o el *grasping*. Conscientes de que aunque el robot haya sido capaz de realizar la tarea en diferentes localizaciones, puede que en la siguiente, de condiciones aparentemente iguales, no sea capaz de lograrlo. Este tipo de retos, o pruebas de estrés, a las que hemos sometido a los diferentes robots, transladándolos del laboratorio a la calle, son inevitables si queremos acabar consiguiendo que los robots coexistan con nosotros. El contexto anterior, nos emplaza a la bonita reflexión sobre la evaluación global de la robótica actual [Nehmzow, 2006], ¿Cómo se mide la calidad del reconocimiento del habla de Gualzru? ¿Con el prestigio del resultado del Rockin14 o con las pruebas de la Universidad?

Parte V

Conclusiones

Capítulo 7

Conclusions

This Tesis has presented the CORTEX as an agent-based cognitive architecture that uses a central, shared representation named DSR as its core representation and communication support. DSR has been defined, formalized and described in detail, explaining its multi-graph structure, its origin and the related scientific papers that have been published about it. This hybrid representation has been shown to work efficiently as a vehicle for reasoning, as well as for the representations of beliefs.

DSR is the major achievement of this Thesis, and it is its condition as an unified representation encompassing geometric and symbolic knowledge that can evolve in the robot's time according to a set of domain rules, what makes it an useful tool for the future development of cognitive robotics. The most relevant features of DSR can be summarized in three key points:

- *Structural*: every element, geometrical or symbolic, is inserted in a particular place and must be related to other elements. This approach wildly differs from the more frequent distributed system lacking a central representation and in which agents exchange a great amount of messages or topics to keep a partial representation of the robot beliefs.
- *Generative*: the size of DSR changes as space, objects, people, etc. are added or removed, but this dynamics is constrained by a set of rules and patterns, all sharing a common dictionary of names.
- *Adaptive*: DSR is continuously changed by the agents, updating links and nodes, as when the mug is grasped and it becomes the child of the Hand. Changes in one side of the graph are propagated and made visible to all agents in the architecture, providing the necessary information to perform context-aware computations.

The CORTEX architecture, with DSR in its core, represents a step towards a more comprehensive, coherent, functional and elegant software architecture for an intelligent robot. It also facilitates the reuse of complex code, saving time and money. For instance, many of the plan's actions, or even entire missions are repeated in different settings with minimal changes to the architecture and to most of the code base developed so far.

The CORTEX architecture also opens many possibilities and questions for further research. For example, regarding the validation of the changes using the grammar, how will geometric changes be validated? How to verify that they are "legal" in a given context? In the current implementation each agent is responsible to validate its proposals, but a more advanced architecture will include an agent specialized in geometric computations and planning, as [Alili et al., 2010] or [Kaelbling and Lozano-Pérez, 2013] suggest.

DSR mixes the geometric and symbolic levels and thus it could be the starting point to generate an episodic memory. This idea raises several questions: would DSR be rich enough to conform such a memory or will it need additional elements? Which elements will be stored, and which of them will be forgotten? How will memories in the episodic memory be matched against the current state of the representation? Is the search for patterns in the graph the best approach? We believe that all these questions can be analysed from the perspective of DSR and the CORTEX architecture.

Another interesting discussion is related to the granularity of DSR. If raw data from sensors are directly inserted into the graph, the system will be more homogeneous but at the cost of an enormous requirement of bandwidth among agents. In order to achieve this objective new research has been already started in our group exploring ways of using proxies to substitute real copies while maintaining the same functionality. Strictly speaking, the data will not be stored, but a similar effect is achieved using the plasticity and dynamic properties of our component-oriented framework.

To sum up, if DSR is correctly employed to encode each situation that the robot faces, it is possible to build around it new agents, or use several of them to evolve the cognitive architecture towards the new challenges these situations represent.

Capítulo 8

Trabajo futuro

El trabajo futuro y el horizonte de DSR y de CORTEX, se presentan prometedores. Como hemos mencionado a lo largo de esta Tesis, su propósito general permitirá ese recorrido. En el horizonte más inmediato, sería interesante trasladar el concepto de *legaly task* de las arquitecturas cognitivas a la representación geométrica. Por ejemplo, en DSR se certifican los cambios legales a nivel de la gramática pero, ¿qué sucede con los cambios geométricos? ¿Cómo comprobar que son "legales"? En CORTEX, esta responsabilidad recae actualmente en el agente, pero tal vez haya que incluir una comprobación o validación de esos cambios pero, ¿cómo se comprueba que una modificación geométrica es válida? Todo esto, probablemente, necesitaría extender CORTEX con algún agente más. Tal vez una ruta inspiradora se encuentre en incluir o usar un planificador geométrico. Algunas propuestas altamente interesantes se encuentran en [Alili et al., 2010], o también en [Kaelbling and Lozano-Pérez, 2013].

Otra línea que despierta mucho interés, surge debido a la unión entre la geometría y lo simbólico en DSR, ya que parece un buen punto de partida para generar una memoria episódica. Esto plantea algunos interrogantes en la forma de proceder inicial. Primero, *Deep State Representation* en su estado actual, ¿sería suficiente? Es decir, en el hipotético caso de que cada cambio estructural de DSR, -estos podían ser entendidos como instantáneas, o incluso como trazas de la tarea realizada por el robot-, se almacenase, ¿esta secuencia de estados sería suficiente para una verdadera memoria episódica?

En caso afirmativo, un nuevo interrogante surge: ¿cómo se encontraría, dentro de esa memoria, el recuerdo de lo que está percibiendo el robot en el momento actual?, ¿es la búsqueda de patrones en el grafo, al igual que hace ahora mismo el sistema entero para validar y decidir la siguiente acción, una tentativa viable?

De otra parte, una línea futura trata de responder al tratamiento de los datos en crudo del sensor en DSR. A este respecto, la pregunta -ya planteada en esta Tesis- es: ¿Y si dentro del grafo también se almacenan los datos de los sensores? Tal vez serviría, como intuimos, para atacar con mayor facilidad los nuevos retos a los que se enfrenten nuestros robots.

Este trabajo, más que futuro, es inmediato. La idea es desarrollarlo en un paso previo. En DSR todos los sensores tienen su correspondiente representación asociada en un nodo. Por tanto, es posible guardar en uno de sus atributos su *proxy*. Esto permitiría en tiempo de ejecución, a un agente, crear una conexión directa con los datos. No se almacenan los datos -estrictamente hablando- en DSR, pero se consigue un efecto similar aprovechando la plasticidad de la arquitectura orientada a componentes subyacente.

En un plano tecnológico nos plantea retos, ¿será capaz de soportar la estructura y los meca-

nismos de gestión el crecimiento del grafo?, ¿se necesitará recurrir a una implementación más estandarizada del grafo, por ejemplo, una base de datos orientada a grafos como Neo4j o hypergraphdb? En cualquier caso, lo interesante es que siempre que el grafo, como estructura de datos de almacenamiento y relación, sea capaz de codificar la situación a la que se enfrente el robot, alrededor de él pueden crearse nuevos agentes que permitan evolucionar la arquitectura CORTEX hacia los nuevos desafíos.

En relación al concepto de emulación interna, una de las futuras líneas de investigación abiertas es extender estas ideas al dominio del nivel simbólico, donde una instancia emulada de la arquitectura pudiera ser invocada para razonar acerca de las consecuencias de ciertas acciones. Esto crearía una realidad alternativa -basada en el actual estado real del mundo-. Esa realidad alternativa iniciaría nuestro simulador con el estado actual de DSR. El mundo exterior de los agentes se modifica transparentemente, del real al simulado. Por tanto se podrían determinar las consecuencias del plan, o parte de sus actos, antes de producirse en la realidad. En este caso, las acciones hipotéticas serían resueltas contra la representación emulada, evaluando la ejecución de las acciones físicas en esa realidad emulada -nuestro simulador-.

De momento no tenemos todos los detalles técnicos de como resolverlo por completo. Debe crearse una especie de copia de la arquitectura, que, a partir de un instante, diverge y evoluciona en una rama paralela. Los agentes trabajarían proponiendo cambios contra DSR y percibiendo y actuando contra la realidad emulada, para resolver las hipotéticas acciones del plan. Los cambios deberían ser validados por un ejecutivo emulado el cual, en caso afirmativo, obtendría la hipotética siguiente acción del plan, a enviar de nuevo al resto de agentes. Todo ello para, de alguna forma, volver al punto inicial, con una conclusión acerca del resultado del plan o la acción. Además, el proceso debería ser realizado en un tiempo razonable.

En cualquier caso, es otra de las líneas interesantes que plantean la propuesta. Muestra sus posibilidades, potencial y recorrido. Por tanto, DSR y CORTEX se presentan como una propuesta de futuro para resolver y elaborar procesos más complejos en el área de la robótica cognitiva.

Capítulo 9

Publications by the author

This section lists the publications of the author, and provides a brief description of their contents and relation with this Thesis.

Publications covered in this thesis:

2015:

1. **Calderita**, L., Bustos, P., Mejías, C. S., Fernández, F., Viciana, R., and Bandera, A. (2015). **Socially Interactive Robot for Motor Rehabilitation Therapies with Paediatric Patients**. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 12(1):99–110. **I.F 0.318**. link

   This article is a compendium about the software and the hardware developed in an interactive robot designed for rehabilitation therapies. Several modules developed in this robot, as well as several conceptual ideas have been incorporated in CORTEX and DSR.

2. Romero-Garcés, A., **Calderita**, L. V., Martínez-Gómez, J., Bandera, J. P., Marfil, R., Manso, L. J., Bandera, A., and Bustos, P. (2015). **Testing a fully autonomous robotic salesman in real scenarios.** In *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, pages 124–130. IEEE. link

   The Thesis proposal was inspired in the difficulties and problems encountered in the development and testing of our robot called Gualzru.

3. (**In press**) Romero-Garcés, A., **Calderita**, L. V., Martínez-Gómez, J., Bandera, J. P., Marfil, R., Manso, L. J., Bandera, A., and Bustos, P. (2015). **The cognitive architecture of a robotic salesman.** *Workshop on Social Robotics and Social and Multimodal Interaction* Conference of the Spanish Association for Artificial Intelligence, CEAPIA, October 2015.

   The paper shows a first implementation of the CORTEX architecture and DSR and it is presented as an example of real usage of the proposal.

2014:

1. Manso, L. J., **Calderita**, L. V., Bustos, P., García, J., Martínez, M., Fernández, F., Romero-Garcés, A., and Bandera, A. (2014). **A general-purpose architecture to control mobile robots.** In *XV Workshop of physical agents: book of proceedings, WAF 2014, June 12th and 13th, 2014 León, Spain*, pages 105–116. link

   This article describes the first step towards the CORTEX architecture, already integrating planning, re-planning, monitoring and learning capabilities and, at the same time, managing a semantic graph, which strongly inspired DSR.

2. **Calderita**, L. V., Manso, L. J., Bustos, P., Suárez-Mejías, C., Fernández, F., and Bandera, A. (2014). **Therapist: Towards an autonomous socially interactive robot for motor and neurorehabilitation therapies for children.** *JMIR Rehabil Assist Technol*, 1(1):e1. link

   Our research has followed a parallel path where the software improvements were adapted to solve specific tasks, in this case related to neuro rehabilitation. This article proposes RoboCog a control architecture that integrates a deliberative planner with a set of modules working at situational or sensorimotor levels. This can be understood as the first steps to the proposal of this Thesis.

3. Martínez-Gómez, J., Marfil, R., **Calderita**, L. V., Bandera, J. P., Manso, L. J. and Bandera, A., Romero-Garcés, A., Bustos, P. (2014). **Toward social cognition in robotics: Extracting and internalizing meaning from perception.** In *XV Workshop of physical agents: book of proceedings, WAF 2014, June 12th and 13th, 2014 León, Spain*, pages 93–104. link

   This article is a discussion on cognitive processing in social robots, and on how the information of the external world should be extracted and used.

2013:

1. **Calderita**, L. V., Bandera, J. P., Bustos, P., and Skiadopoulos, A. (2013b). **Model-based reinforcement of kinect depth data for human motion capture applications.** *Sensors*, 13(7):8835–8855. **I.F. 2.048** link

   In the search of a robust method in the human-motion-capture without markers this paper describes a method to filter the results provided by an RGBD device. A kinematic model of the body and its inverse function is used to improve the estimation of the sensor ant to eliminate artefacts caused by occlusions and noise.

2. Bustos, P., Martinez-Gomez, J., Garcia-Varea, I., Rodriguez-Ruiz, L., Bachiller, P., **Calderita**, L., Sanchez, A., Bandera, A., and Bandera, J. P. (2013). **Multimodal interaction with loki.** In *Proceedings, Workshop of Phisical Agents, WAF 2013*. link

   This paper proposes a central representation of the robot's world using a kinematic tree. This is related to the previous article and of course to the geometric representation proposal in this Thesis.

3. **Calderita**, L., Bustos, P., Suárez, C., Fernández, F., and Bandera, A. (2013). **Rehabilitation for children while playing with a robotic assistant in a serious game**. In *Neurotechnix*. link

   The experiments performed in this paper shows the necessity of increasing levels of adaptability in order to cope with children. Also the problems and barriers imposed by the communication among components without a well defined structure are clearly observed.

2011:

1. **Calderita**, L., Bachiller, P., Bandera, J. P., Bustos, P. and Núñez, P. (2011) **MIMIC: A Human motion imitation component for RoboComp.** In *Workshop on Recognition and Action for Scene understanding (REACTS)*, link

   The article shows the first step towards the estimation, internalization and imitation of the human body, from the robot's perspective.

2. Naranjo-Saucedo, A., Suárez, C., Bustos, P., **Calderita**, L., Cintas, R., Bachiller, P., and Parra-Calderón, C. (2011). **Design of a social robot as assistant to training children with motor impairments.** In *Proceedings of the ROBOT 2011 Conference*, Seville, Spain. link

   Following a similar approach than the other related papers, several experiments show the limitations in current architectures to understand children's behaviour during rehabilitation activities. The overall conclusion from the point of view of this Thesis is that a real social robot needs to better understand and recognize the scene ahead.

2010:

1. Manso, L. J., Bachiller, P., Bustos, P., Núñez, P., Cintas, R., and **Calderita**, L. (2010). **RoboComp: a Tool-based Robotics Framework.** In *Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, pages 251–262. link

   This article describes our robotics framework on which DSR and CORTEX have been built.

Publications not covered in this thesis:

2014:

1. **Calderita**, L. V., Bandera, J. P., Manso, L. J., and Vázquez-Martín, R. (2014). **3d laser from rgbd projections in robot local navigation.** In *XV Workshop of physical agents: book of proceedings, WAF 2014, June 12th and 13th, 2014 León, Spain*, pages 18–28. link

2013:

1. **Calderita**, L., Bustos, P., Suárez, C., Fernández, F., and Bandera, A. (2013). RGB-D Database for Affective Multimodal Human-Robot Interaction. In *Proceedings of the XIV Workshop on Physycal Agents (WAF-2013)*, Madrid, September 18-19, 2013 link

2012:

1. **Calderita**, L., Manso, L., and Núñez, P. (2012). **Assessment of primesense rgb-d camera for robotic applications.** In *Proceedings of the XIII Workshop of Physical Agents 2012, WAF 2012*, pages 147–153, Santiago de Compostela, Spain. link
2. Cid, F., Manso, L. J., **Calderita**, L. V., Sánchez, A., and Núñez, P. (2012). **Engaging human-to-robot attention using conversational gestures and lip-synchronization.** In *Journal of Physical Agents*, 6. link

2011:

1. Cid, F., Cintas, R., Manso, L., **Calderita**, L., Sánchez, A., and Núñez, P (2011). **A real-time synchronization algorithm between text-to-speech (tts) system and robot mouth for social robotic applications.** In *Workshop of Physical Agents* Albacete, Spain, pages 81–86. link

2010:

1. **Calderita**, Luis Vicente and Moreno, Vidal and Curto, Belén (2010). **Navegación de Robots en Formación: un Enfoque Reactivo con Restricciones.** In *Avances en Informática y Automática*. link

Bibliografía

[Agarwal and Triggs, 2006] Agarwal, A. and Triggs, B. (2006). Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58.

[Agre and Chapman, 1987] Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *AAAI*, volume 87, pages 286–272.

[Alami et al., 2006] Alami, R., Clodic, A., Montreuil, V., Sisbot, E. A., and Chatila, R. (2006). Toward human-aware robot task planning. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 39–46.

[Alcázar et al., 2010] Alcázar, V., Guzmán, C., Prior, D., Borrajo, D., Castillo, L., and Onaindia, E. (2010). Pelea: Planning, learning and execution architecture. *PlanSIG'10*, pages 17–24.

[Ali and Shapiro, 1993] Ali, S. S. and Shapiro, S. C. (1993). Natural language processing using a propositional semantic network with structured variables. *Minds and machines*, 3(4):421–451.

[Alili et al., 2010] Alili, S., Pandey, A. K., Sisbot, E. A., and Alami, R. (2010). Interleaving symbolic and geometric reasoning for a robotic assistant. In *ICAPS Workshop on Combining Action and Motion Planning*.

[Anderson et al., 2004] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of the mind. *Psychological review*, 111(4):1036.

[Angelopoulou et al., 1992] Angelopoulou, E., Hong, T. H., and Wu, A. Y. (1992). World Model Representations for Mobile Robots. In *Proc. of the Intelligent Vehicles Symposium*, pages 293–297.

[Arkin et al., 1987] Arkin, R. C., Riseman, E. M., and Hanson, A. R. (1987). Aura: An architecture for vision-based robot navigation. In *proceedings of the DARPA Image Understanding Workshop*, pages 417–431.

[Azad et al., 2007] Azad, P., Ude, A., Asfour, T., and Dillmann, R. (2007). Stereo-based markerless human motion capture for humanoid robot systems. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 3951–3956, Roma, Italy.

[Baerlocher, 2001] Baerlocher, P. (2001). *Inverse kinematics techniques for the interactive posture control of articulated figures*. PhD thesis.

[Baerlocher and Boulic, 2004] Baerlocher, P. and Boulic, R. (2004). An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The visual computer*, 20(6):402–417.

[Bandera, 2010] Bandera, J. P. (2010). *Vision-based gesture recognition in a robot learning by imitation framework*. Ph.d. dissertation, Dpto. Tecnología Electrónica, Universidad de Málaga, Spain, Universidad de Málaga, Spain.

[Baron-Cohen, 1995] Baron-Cohen, S. (1995). The eye direction detector (edd) and the shared attention mechanism (sam): Two cases for evolutionary psychology. In *Joint attention: Its origins and role in development*. Lawrence Erlbaum Associates, Inc.

[Beetz et al., 2010] Beetz, M., Jain, D., Mösenlechner, L., and Tenorth, M. (2010). Towards performing everyday manipulation activities. *Robotics and Autonomous Systems*, 58(9):1085–1095.

[Benjamin et al., 2004] Benjamin, D. P., Lyons, D. M., and Lonsdale, D. W. (2004). Adapt: A cognitive architecture for robotics. In *ICCM*, pages 337–338.

[Biswas et al., 2011] Biswas, J., Coltin, B., and Veloso, M. (2011). Corrective gradient refinement for mobile robot localization. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 73–78.

[Blumenthal et al., 2013] Blumenthal, S., Bruyninckx, H., Nowak, W., and Prassler, E. (2013). A scene graph based shared 3d world model for robotic applications. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 453–460. IEEE.

[Bonasso, 1991] Bonasso, R. P. (1991). Integrating reaction plans and layered competences through synchronous control. In *IJCAI*, pages 1225–1233.

[Borgida et al., 1991] Borgida, A., Brachman, R., Coombs, M., and Sowa, J. (1991). Principles of semantic networks: explorations in the representation of knowledge. *Morgan Kaufmann series in representation and reasoning Show all parts in this series*.

[Brock and Khatib, 2002] Brock, O. and Khatib, O. (2002). Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052.

[Brooks, 1986] Brooks, R. (1986). A robust layered control system for a mobile robot. *Journal of Robotics and Automation*, 2(1):14–23.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. *Artificial intelligence*, 47(1):139–159.

[Brooks, 1999] Brooks, R. A. (1999). *Cambrian intelligence: the early history of the new AI*, volume 97. Mit Press Cambridge, MA.

[Brooks et al., 1989] Brooks, R. A., Flynn, A. M., and Fast, C. (1989). Fast, cheap and out of control: A robot invasion of the solar system. *Journal of the British Interplanetary Society*, 42:478–485.

[Bustos et al., 2013] Bustos, P., Martínez-Gómez, J., García-Varea, I., Rodríguez-Ruiz, L., Bachiller, P., Calderita, L., Sánchez, A., Bandera, A., and Bandera, J. P. (2013). Multimodal interaction with loki. In *Proceedings, Workshop of Phisical Agents, WAF 2013*.

[Calderita et al., 2015] Calderita, L., Bustos, P., Mejías, C. S., Fernández, F., Viciana, R., and Bandera, A. (2015). Asistente robótico socialmente interactivo para terapias de rehabilitación motriz con pacientes de pediatría. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 12(1):99–110.

[Calderita et al., 2013a] Calderita, L., Bustos, P., Suárez, C., Fernández, F., and Bandera, A. (2013a). Therapist: towards an autonomous socially interactive robot for motor and neurorehabilitation therapies for children. *REHAB2013*.

[Calderita et al., 2012] Calderita, L., Manso, L., and Núñez, P. (2012). Assessment of primesense rgb-d camera for robotic applications. In *Proceedings of the XIII Workshop of Physical Agents 2012, WAF 2012*, pages 147–153, Santiago de Compostela, Spain.

[Calderita et al., 2013b] Calderita, L. V., Bandera, J. P., Bustos, P., and Skiadopoulos, A. (2013b). Model-based reinforcement of kinect depth data for human motion capture applications. *Sensors*, 13(7):8835–8855.

[Calderita et al., 2014a] Calderita, L. V., Bandera, J. P., Manso, L. J., and Vázquez-Martín, R. (2014a). 3d laser from rgbd projections in robot local navigation. In *XV Workshop of physical agents: book of proceedings, WAF 2014, June 12th and 13th, 2014 León, Spain*, pages 18–28.

[Calderita et al., 2014b] Calderita, V. L., Manso, J. L., Bustos, P., Suárez-Mejías, C., Fernández, F., and Bandera, A. (2014b). Therapist: Towards an autonomous socially interactive robot for motor and neurorehabilitation therapies for children. *JMIR Rehabil Assist Technol*, 1(1):e1.

[Carbonell et al., 1991] Carbonell, J., Etzioni, O., Gil, Y., Joseph, R., Knoblock, C., Minton, S., and Veloso, M. (1991). Prodigy: An integrated architecture for planning and learning. *ACM SIGART Bulletin*, 2(4):51–55.

[Chamorro and Vázquez Martín, 2009] Chamorro, D. and Vázquez Martín, R. (2009). R-ORM: relajación en el método de evitar colisiones basado en restricciones. In *X Workshop de Agentes Físicos, Cáceres, España*.

[Chella et al., 2008] Chella, A., Frixione, M., and Gaglio, S. (2008). A cognitive architecture for robot self-consciousness. *Artificial intelligence in medicine*, 44(2):147–54.

[Choi et al., 2009] Choi, D., Kang, Y., Lim, H., and You, B.-J. (2009). Knowledge-based control of a humanoid robot. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3949–3954. IEEE.

[Choi et al., 2007] Choi, D., Könik, T., Nejati, N., Park, C., and Langley, P. (2007). A believable agent for first-person shooter games. In *AIIDE*, pages 71–73.

[Choi et al., 2011] Choi, J., Choi, M., Nam, S. Y., and Chung, W. K. (2011). Autonomous topological modeling of a home environment and topological localization using a sonar grid map. *Autonomous Robots*, 30(4):351–368.

[Cid et al., 2011] Cid, F., Cintas, R., Manso, L., Calderita, L., Sánchez, A., and Núñez, P. (2011). A real-time synchronization algorithm between text-to-speech (tts) system and robot mouth for social robotic applications. In *Proceedings, Workshop en Agentes Físicos (WAF 2011), Albacete, Spain*, pages 81–86.

[Cid et al., 2014] Cid, F., Moreno, J., Bustos, P., and Nuñez, P. (2014). Muecas: A multi-sensor robotic head for affective human robot interaction and imitation. *Sensors*, 14(5):7711–7737.

[Cid et al., 2013] Cid, F., Prado, J. A., Bustos, P., and Núnez, P. (2013). A real time and robust facial expression recognition and imitation approach for affective human-robot interaction using gabor filtering. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2188–2193. IEEE.

[Connell, 1989] Connell, J. H. (1989). A colony architecture for an artificial creature. Technical report, DTIC Document.

[Connell, 1992] Connell, J. H. (1992). Sss: A hybrid architecture applied to robot navigation. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2719–2724. IEEE.

[Contini, 1972] Contini, R. (1972). Body segment parameters, part ii. *Artificial Limbs*, 16(1):1–19.

[Corchado and López, 2002] Corchado, J. M. and López, J. M. M. (2002). *Introducción a la teoría de agentes y sistemas multiagente*. Catedral.

[Corkill, 1991] Corkill, D. D. (1991). Blackboard systems. *AI expert*, 6(9):40–47.

[Corkum and Moore, 1998] Corkum, V. and Moore, C. (1998). The origins of joint visual attention in infants. *Developmental psychology*, 34(1):28.

[Cowie, 2009] Cowie, R. (2009). Perceiving emotion: towards a realistic understanding of the task. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1535):3515–3525.

[Cowie et al., 2001] Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., and Taylor, J. G. (2001). Emotion recognition in human-computer interaction. *Signal Processing Magazine, IEEE*, 18(1):32–80.

[Demirdjian et al., 2005] Demirdjian, D., Ko, T., and Darrell, T. (2005). Untethered gesture acquisition and recognition for virtual world manipulation. *Virtual Reality*, 8:222–230.

[Deutscher and Reid, 2005] Deutscher, J. and Reid, I. (2005). Articulated Body Motion Capture by Stochastic Search. *International Journal of Computer Vision*, 61(2):185–205.

[Ekman and Friesen, 1978] Ekman, P. and Friesen, W. (1978). Facial action coding system: a technique for the measurement of facial movement. *Consulting Psychologists, San Francisco*.

[Elfes, 1989] Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57.

[Eliasmith, 1999] Eliasmith, C. (1999). The third contender: a critical examination of the dynamicist theory of cognition. In Thagard, P., editor, *Mind Readings: Introductory Selections on Cognitive Science*. MIT Press.

[Erman et al., 1980] Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R. (1980). The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys (CSUR)*, 12(2):213–253.

[Feil-Seifer and Matarić, 2011] Feil-Seifer, D. and Matarić, M. J. (2011). Socially assistive robotics. *Robotics & Automation Magazine, IEEE*, 18(1):24–31.

[Firby, 1987] Firby, R. J. (1987). An investigation into reactive planning in complex domains. In *AAAI*, volume 87, pages 202–206.

[Foote, 2013] Foote, T. (2013). tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–6. IEEE.

[Franklin et al., 1997] Franklin, S., Franklin, S., Graesser, A., and Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. *ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 21–35.

[Ganaphathi et al., 2010] Ganaphathi, V., Plagemann, C., Koller, D., and Thrun, S. (2010). Real time motion capture using a single time-of-flight camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 755–762, San Francisco, USA.

[Gat, 1991] Gat, E. (1991). *Reliable goal-directed reactive control of autonomous mobile robots*. PhD thesis.

[Gat, 1998] Gat, E. (1998). On Three-Layer Architectures. In *Artificial Intelligence and Mobile Robots*, pages 195–210. MIT Press.

[Grauman et al., 2003] Grauman, K., Shakhnarovich, G., and Darrell, T. (2003). Inferring 3d structure with a statistical image-based shape model. In *Proceedings of the International Conference on Computer Vision*, pages 641–647, Nice, France.

[Haazebroek et al., 2011] Haazebroek, P., van Dantzig, S., and Hommel, B. (2011). A computational model of perception and action for cognitive robotics. *Cognitive processing*, 12(4):355–65.

[Haigh and Veloso, 1998] Haigh, K. Z. and Veloso, M. M. (1998). Interleaving planning and robot execution for asynchronous user requests. In *Autonomous agents*, pages 79–95. Springer.

[Hanford, 2011] Hanford, S. D. (2011). *A Cognitive Robotic System based on the SOAR cognitive architecture for mobile robot navigation, search and mapping missions*. PhD thesis.

[Harrison et al., 2003] Harrison, A. M., Schunn, C. D., et al. (2003). Act-r/s: Look ma, noçognitive-map. In *International conference on cognitive modeling*, pages 129–134.

[Hawking and Mlodinow, 2010] Hawking, S. W. and Mlodinow, L. (2010). *El gran diseño*. Crítica.

[Hayes-Roth, 1985] Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial intelligence*, 26(3):251–321.

[Heerink et al., 2010] Heerink, M., Kröse, B., Evers, V., and Wielinga, B. (2010). Assessing acceptance of assistive social agent technology by older adults: the almere model. *International journal of social robotics*, 2(4):361–375.

[Helmert, 2006] Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

[Hiebert, 1981] Hiebert, K. (1981). An evaluation of mathematical software that solves nonlinear least squares problems. *ACM Transactions on Mathematical Software (TOMS)*, 7(1):1–16.

[Hofland et al., 2005] Hofland, K., Jørgensen, A. M., Drange, E.-M., and Stenström, A.-B. (2005). Cola: A spanish spoken corpus of youth language. In *Proceedings from the Corpus Linguistics Conference Series. U. o. B. Centre for Corpus Research. Birmingham, Centre for Corpus Research, University of Birmingham*, volume 1.

[Holland, 2003] Holland, O. (2003). Robots with internal models. *Journal of Consciousness Studies*, 10(4-5):77–109.

[Holland, 2004] Holland, O. (2004). The future of embodied artificial intelligence: Machine consciousness? In *Embodied artificial intelligence*, pages 37–53. Springer.

[Huang et al., 2001] Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., and Tanie, K. (2001). Planning walking patterns for a biped robot. *Robotics and Automation, IEEE Transactions on*, 17(3):280–289.

[Jaimes and Sebe, 2007] Jaimes, A. and Sebe, N. (2007). Multimodal human–computer interaction: A survey. *Computer vision and image understanding*, 108(1):116–134.

[Jelinek, 1997] Jelinek, F. (1997). *Statistical methods for speech recognition*. MIT press.

[Jennings, 1993] Jennings, N. R. (1993). Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *International Journal of Intelligent and Cooperative Information Systems*, 2(03):289–318.

[Joosse et al., 2013] Joosse, M., Sardar, A., Lohse, M., and Evers, V. (2013). Behave-ii: The revised set of measures to assess users' attitudinal and behavioral responses to a social robot. *International journal of social robotics*, 5(3):379–388.

[Kaelbling and Lozano-Pérez, 2013] Kaelbling, L. P. and Lozano-Pérez, T. (2013). Integrated task and motion planning in belief space. *The International Journal of Robotics Research*.

[Kessous et al., 2010] Kessous, L., Castellano, G., and Caridakis, G. (2010). Multimodal emotion recognition in speech-based interaction using facial expression, body gesture and acoustic analysis. *Journal on Multimodal User Interfaces*, 3(1-2):33–48.

[King et al., 2007] King, J. R. A. R. et al. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

[Kirsch, 2009] Kirsch, A. (2009). Robot learning language—integrating programming and learning for cognitive systems. *Robotics and Autonomous Systems*, 57(9):943–954.

[Laird et al., 2012] Laird, J., Kinkade, K., Mohan, S., and Xu, J. (2012). Cognitive robotics using the soar cognitive architecture. In *8th International Workshop on Cognitive Robotics*, pages 46–54.

[Laird, 2008] Laird, J. E. (2008). Extending the soar cognitive architecture. *Frontiers in Artificial Intelligence and Applications*, 171:224.

[Laird et al., 1987] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64.

[Langley, 2006] Langley, P. (2006). Cognitive architectures and general intelligent systems. *AI magazine*, 27(2):33.

[Langley and Choi, 2006] Langley, P. and Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1469. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Langley et al., 2009] Langley, P., Laird, J. E., and Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160.

[Langley and Messina, 2004] Langley, P. and Messina, E. (2004). Experimental studies of integrated cognitive systems. Technical report, DTIC Document.

[Lemaignan et al., 2010] Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., and Beetz, M. (2010). Oro, a knowledge management platform for cognitive architectures in robotics. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3548–3553. IEEE.

[Levemberg, 1944] Levemberg, K. (1944). A method for the solution of certain problems nonlinear in least square. *Quarth. Appl. Math*, 2:164–168.

[Levesque and Lakemeyer, 2008] Levesque, H. and Lakemeyer, G. (2008). Cognitive robotics. In *Handbook of Knowledge Representation*.

[Lewis, 1993] Lewis, R. L. (1993). An architecturally-based theory of human sentence comprehension. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pages 108–113.

[Lindström et al., 2000] Lindström, M., Orebäck, A., Christensen, H., et al. (2000). Berra: A research architecture for service robots. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 4, pages 3278–3283. IEEE.

[Liu and Chen, 2003] Liu, X. and Chen, T. (2003). Video-based face recognition using adaptive hidden markov models. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–340. IEEE.

[Lourakis and Argyros, 2009] Lourakis, M. I. and Argyros, A. A. (2009). Sba: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2.

[Maes, 1991] Maes, P. (1991). The agent network architecture (ana). *SIGART Bulletin*, 2(4):115–120.

[Magerko et al., 2004] Magerko, B., Laird, J., Assanie, M., Kerfoot, A., and Stokes, D. (2004). Ai characters and directors for interactive computer games. *Ann Arbor*, 1001(48):109–2110.

[Manso, 2013] Manso, L. J. (2013). *Perception as Stochastic Sampling on Dynamic Graph Spaces*. PhD thesis, University of Extremadura.

[Manso et al., 2010] Manso, L. J., Bachiller, P., Bustos, P., Núñez, P., Cintas, R., and Calderita, L. (2010). RoboComp: a Tool-based Robotics Framework. In *Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, pages 251–262.

[Manso et al., 2014] Manso, L. J., Calderita, L. V., Bustos, P., Garcıa, J., Martınez, M., Fernández, F., Romero-Garcés, A., and Bandera, A. (2014). A general-purpose architecture to control mobile robots. In *XV Workshop of physical agents: book of proceedings, WAF 2014, June 12th and 13th, 2014 León, Spain*, pages 105–116.

[Marquardt, 1963] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441.

[Martz, 2007] Martz, P. (2007). *OpenSceneGraph Quick Start Guide*. Computer Graphics Systems Development Corporation, Mountain View, California.

[Matarić, 1990] Matarić, M. J. (1990). A Distributed Model for Mobile Robot Environment-Learning and Navigation. Technical report, Massachusetts Institute of Technology AI Lab.

[Mateos et al., 2010] Mateos, J., Sánchez, A., Manso, L. J., Bachiller, P., and Bustos, P. (2010). RobEx: an open-hardware robotics platform. In *Workshop of Physical Agents*.

[Mayer et al., 2009] Mayer, C., Wimmer, M., Eggers, M., and Radig, B. (2009). Facial expression recognition with 3d deformable models. In *Advances in Computer-Human Interactions, 2009. ACHI'09. Second International Conferences on*, pages 26–31. IEEE.

[McManus and Bynum, 1996] McManus, J. W. and Bynum, W. L. (1996). Design and analysis techniques for concurrent blackboard systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 26(6):669–680.

[Meijer, 2012] Meijer, R. (2012). Pddl planning problems and groove graph transformations: combining two worlds with a translator. In *17th Twente student conference on IT. University of Twente*.

[Mikić et al., 2003] Mikić, I., Trivedi, M., Hunter, E., and Cosman, P. (2003). Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223.

[Minsky, 1988] Minsky, M. (1988). *The Society of Mind*. Simon & Schuster.

[Moeslund et al., 2006] Moeslund, T., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126.

[Mori and Malik, 2002] Mori, G. and Malik, J. (2002). Estimating human body configurations using shape context matching. *Lecture Notes in Computer Science*, 2352:666–674.

[Naranjo-Saucedo et al., 2011] Naranjo-Saucedo, A., Suárez, C., Bustos, P., Calderita, L., Cintas, R., Bachiller, P., and Parra-Calderón, C. (2011). Design of a social robot as assistant to training children with motor impairments. In *Proceedings of the ROBOT 2011 Conference*, Seville, Spain.

[Nehmzow, 2006] Nehmzow, U. (2006). *Scientific Methods in Mobile Robotics: quantitative analysis of agent behaviour*. Springer Science & Business Media.

[Newell, 1962] Newell, A. (1962). Some problems of basic organization in problem-solving programs. Technical report, DTIC Document.

[Newell, 1994] Newell, A. (1994). *Unified theories of cognition*. Harvard University Press.

[Nii, 1986] Nii, H. P. (1986). The blackboard model of problem solving and the evolution of blackboard architectures. *AI magazine*, 7(2):38.

[Nii et al., 1982] Nii, H. P., Feigenbaum, E. A., and Anton, J. J. (1982). Signal-to-symbol transformation: Hasp/siap case study. *AI magazine*, 3(2):23.

[Noë, 2004] Noë, A. (2004). *Action in perception*. The MIT Press.

[Norman, 2013] Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition*. Basic books.

[Nuxoll and Laird, 2007] Nuxoll, A. M. and Laird, J. E. (2007). Extending cognitive architecture with episodic memory. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1560. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Olson, 2011] Olson, E. (2011). Apriltag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3400–3407. IEEE.

[Orebäck and Christensen, 2003] Orebäck, A. and Christensen, H. I. (2003). Evaluation of architectures for mobile robotics. *Autonomous robots*, 14(1):33–49.

[Payton, 1986] Payton, D. W. (1986). An architecture for reflexive autonomous vehicle control. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1838–1845. IEEE.

[Perez and Carbonell, 1994] Perez, A. and Carbonell, J. G. (1994). Control knowledge to improve plan quality. AAAI.

[Peter Bonasso et al., 1997] Peter Bonasso, R., James Firby, R., Gat, E., Kortenkamp, D., Miller, D. P., and Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):237–256.

[Poole and Mackworth, 2010] Poole, D. L. and Mackworth, A. K. (2010). *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

[Poppe, 2007] Poppe, R. (2007). Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108:4–18.

[Prado et al., 2012] Prado, J. A., Simplício, C., Lori, N. F., and Dias, J. (2012). Visuo-auditory multimodal emotional structure to improve human-robot-interaction. *International Journal of Social Robotics*, 4(1):29–51.

[Puigbo et al., 2013] Puigbo, J.-y., Pumarola, A., and Tellez, R. (2013). Controlling a general purpose service robot by means of a cognitive architecture. In *Artificial Intelligence and Cognition Workshop*.

[Pulido et al., 2014] Pulido, J. C., González, J. C., González-Ferrer, A., García, J., Fernández, F., Bandera, A., et al. (2014). Goal-directed generation of exercise sets for upper-limb rehabilitation assisted by humanoid robots. In *Knowledge Engineering for Planning and Scheduling Workshop*, pages 21–26.

[Quigley et al., 2009] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). ROS: an open-source Robot Operating System. In *Proc. of ICRA Workshop on Open Source Software*.

[Quinlan and Khatib, 1993] Quinlan, S. and Khatib, O. (1993). Elastic bands: connecting path planning and control. *[1993] Proceedings IEEE International Conference on Robotics and Automation*.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). Bdi agents: From theory to practice. *Icmas*, pages 312–319.

[Rego et al., 2010] Rego, P., Moreira, P. M., and Reis, L. P. (2010). Serious games for rehabilitation: A survey and a classification towards a taxonomy. In *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on*, pages 1–6. IEEE.

[Romero-Garcés et al., 2015] Romero-Garcés, A., Calderita, L. V., Martínez-Gómez, J., Bandera, J. P., Marfil, R., Manso, L. J., Bandera, A., and Bustos, P. (2015). Testing a fully autonomous robotic salesman in real scenarios. In *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, pages 124–130. IEEE.

[Russell and Norvig, 2009] Russell, S. J. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition.

[Scheutz, 2013] Scheutz, M. (2013). Computational mechanisms for mental models in human-robot interaction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8021 LNCS(PART 1):304–312.

[Selfridge, 1958] Selfridge, O. G. (1958). Pandemonium: a paradigm for learning in mechanisation of thought processes.

[Shotton et al., 2013] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116 – 124.

[Sloman et al., 2006a] Sloman, A., Wyatt, J., Hawes, N., Chappell, J., and Kruijff, G.-J. (2006a). Long term requirements for cognitive robotics. In *Cognitive Robotics Papers from the 2006 AAAI Workshop Technical Report WS0603*, number McCarthy, page 143–150.

[Sloman et al., 2006b] Sloman, A., Wyatt, J., Hawes, N., Chappell, J., and Kruijff, G.-J. M. (2006b). Long term requirements for cognitive robotics. In *Cognitive Robotics: Papers from the 2006 AAAI Workshop: Technical Report WS-06-03, http://www. aaai. org/Library/Workshops/ws06-03. php*, pages 143–150.

[Sminchisescu and Triggs, 2003] Sminchisescu, C. and Triggs, B. (2003). Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22(6):371–391.

[Starck and Hilton, 2008] Starck, J. and Hilton, A. (2008). Model-based human shape reconstruction from multiple views. *Computer Vision and Image Understanding*, 111(2):179 – 194.

[Stenger et al., 2001] Stenger, B., Mendonca, P. R. S., and Cipolla, R. (2001). Model-based hand tracking using an unscented kalman filter. In *Proceedings of the British Machine Vision Conference*, pages 63–72, Manchester,United Kingdom.

[Tambe et al., 1995] Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., and Schwamb, K. (1995). Intelligent agents for interactive simulation environments. *AI magazine*, 16(1):15.

[Thrun, 2002] Thrun, S. (2002). Robotic Mapping: A Survey. Technical report, School of Computer Science, Carnegie Mellon University.

[Trafton et al., 2013] Trafton, G., Hiatt, L., Harrison, A., Tamborello, F., Khemlani, S., and Schultz, A. (2013). Act-r/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, 2(1):30–55.

[Trafton et al., 2009] Trafton, J. G., Fransen, B., Harrison, A. M., and Bugajska, M. (2009). An embodied model of infant gaze-following. Technical report, DTIC Document.

[Veloso and Carbonell, 1993] Veloso, M. M. and Carbonell, J. G. (1993). Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. In *Case-Based Learning*, pages 55–84. Springer.

[Walker and Walker, 2001] Walker, C. and Walker, E. (2001). *Game Modeling Using Low Polygon Techniques (Charles River Media Graphics)*. Charles River Media, 1st edition.

[Wallach, 2006] Wallach, H. M. (2006). Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM.

[Wang, 1995] Wang, X. (1995). Learning by observation and practice: An incremental approach for planning operator acquisition. In *ICML*, pages 549–557. Citeseer.

[Williamson, 1980] Williamson, F. (1980). Richard courant and the finite element method: a further look. *Historia Mathematica*, 7(4):369–378.

[Wintermute, 2012] Wintermute, S. (2012). Imagery in cognitive architecture: Representation and control at multiple levels of abstraction. *Cognitive Systems Research*, 19-20:1–29.

[Yamamoto and Yagishita, 2000] Yamamoto, M. and Yagishita, K. (2000). Scene constraintsaided tracking of human body. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 151–156, Hilton Head Island, USA.

[Yang and Brock, 2009] Yang, Y. and Brock, O. (2009). Elastic roadmaps - motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130.

[Younes et al., 2005] Younes, H. L. S., Littman, M. L., Weissman, D., and Asmuth, J. (2005). The first probabilistic track of the international planning competition. *J. Artif. Int. Res.*, 24(1):851–887.

[Zeng et al., 2009] Zeng, Z., Pantic, M., Roisman, G. I., and Huang, T. S. (2009). A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(1):39–58.

[Zhang and Patel, 2006] Zhang, J. and Patel, V. L. (2006). Distributed cognition, representation, and affordance. *Pragmatics & Cognition*, 14(2):333–341.

[Zhang et al., 2012] Zhang, L., Sturm, J., Cremers, D., and Lee, D. (2012). Real-time human motion tracking using multiple depth cameras. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pages 2389–2395, Vilamoura, Portugal.