The CORTEX Cognitive Robotics Architecture: use cases

P. Bustos[a], L.J. Manso[d], A.J. Bandera[b], J.P. Bandera[b], I. García-Varea[c], J. Martínez-Gómez[c]

[a] *University of Extremadura, Spain*
[b] *University of Málaga, Spain*
[c] *University of Castilla-La Mancha, Spain*
[d] *Aston University, United Kingdom*

Abstract

CORTEX is a cognitive robotics architecture inspired by three key ideas: modularity, internal modelling and graph representations. CORTEX is also a computational framework designed to support early forms of intelligence in real world, human interacting robots, by selecting an a priori functional decomposition of the capabilities of the robot. This set of abilities was then translated to computational modules or agents, each one built as a network of software interconnected components. The nature of these agents can range from pure reactive modules connected to sensors and/or actuators, to pure deliberative ones, but they can only communicate with each other through a graph structure called Deep State Representation (DSR). DSR is a short-term dynamic representation of the space surrounding the robot, the objects and the humans in it, and the robot itself. All these entities are perceived and transformed into different levels of abstraction, ranging from geometric data to high-level symbolic relations such as "the person is talking and gazing at me". The combination of symbolic and geometric information endows the architecture with the potential to simulate and anticipate the outcome of the actions executed by the robot. In this paper we present recent advances in the CORTEX architecture and several real-world human-robot interaction scenarios in which they have been tested. We describe our interpretation of the ideas inspiring the architecture and the reasons why this specific computational framework is a promising architecture for the social robots of tomorrow.

Keywords: Cognitive Robotics, Robot Control Architectures

1. Introduction

Service robotics in professional and personal/domestic applications is experiencing strong global growth. Future product visions point to service robots of higher sophistication, capability and value, such as assistive robots for supporting the elderly. Cognitive robotics is concerned with endowing robots with the capacity to plan solutions for complex goals and to enact those plans while being reactive to unexpected changes in their environment. To pursue this goal, cognitive architectures for robotics attempt to provide a reasonable structure where all the functionalities of a working cognitive robot can be fitted. If the final goal is to implement these architectures within the upcoming, new service robots, the structure designed will be forced to provide an adequate response to the demanding requirements imposed by the human-robot interaction scenario.

Currently, robots working in real scenarios are not usually driven by a cognitive architecture. In the exhaustive review by Kostseruba and Tsotsos [19], the authors that only a few architectures implement multiple skills for complex scenarios. One of the cited architectures is CORTEX [31, 4]. The main feature of the CORTEX architecture is the existence of a unified, dynamic working memory that can represent sensory data and high-level symbols. This Deep State Representation (DSR) is a short-term dynamic representation of the space surrounding the robot, the objects and humans in it, and the robot itself. All these entities are perceived and internalized in the DSR by transforming them into different levels of abstraction, ranging from the raw data provided by the sensors to high-level symbolic relationships. To be deployed in real scenarios, the DSR has been designed not to be too demanding on the precision or updating requirements.

Over the last few years, CORTEX has been successfully implemented within platforms equipped with different actuators and/or sensors, and has been in charge of achieving different use cases such as attracting potential consumers to a stand [31] or conducting geriatric tests on elderly people [35]. Although it has been updated over time, the overall structure remains the same in all cases. This paper describes the global idea driving the successful design of the CORTEX architecture, which can be summarized in the use of predictive imagery in a robotics cognitive architecture. With memory being represented as a shared repository of goals, problems and partial results, and modifiable by all agents, our paradigm can resemble the blackboard model. However, in CORTEX, the objective of this central representation is not only to be a means for moving/sharing information, but also to give a response to the three different meta-problems identified by S. Wintermute [36]: Physical symbol grounding, perceptual abstraction and irreducibility. To achieve these goals, the DSR is also used to estimate how the current state changes when something perturbs it. Thus, it can serve as an efficient predictive mechanism [16], which allow the architecture to quickly look for alternative courses of actions.

The rest of the paper is organised as follows: Section 2 describes the structure of the CORTEX architecture and of the DSR. The aim is to provide a clear snapshot of CORTEX that allows the comparison with other blackboard or Global Workspace Theory (GWT)-based approaches in Section 3. As mentioned above, the core of CORTEX is the existence of the DSR. This graph representation is not only a blackboard, but a dynamic structure able to integrate the multiple perceptions and facets of the same entity, and also able to *move* this entity towards the close future. The previously mentioned meta-problems that this inner world must address are also described in Section 2. Section 4 introduces some of the most frequently used agents present in the instantiations of CORTEX, which are briefly described in Section 5. This Section describes our experimental validation of CORTEX and collects results obtained in four scenarios of application. Based on these experiences, Section 6 discusses the main properties of CORTEX. Conclusions and our current on-going efforts to improve the architecture are detailed in Section 7.

2. The CORTEX Architecture

Briefly, we can describe CORTEX as a collection of agents that cooperate to achieve a goal. Thus, the deployment of a CORTEX architecture within a specific use case and robot starts by selecting an a priori functional decomposition of the capabilities of the robot. They are then translated to computational modules or *agents*. These agents can be anywhere in the reactive-deliberative spectrum, but each of them is in charge of a basic robotic functionality affecting a specific domain. Some agents transform sensor information into internal objects. Other agents generate actions activating the robot effectors and compute plans to satisfy the human requests or to react to a goal. All of them interact through the DSR. This graph represents the current state of the robot and its environment, and is created and updated by the agents according partly to each agent's internal dynamics and partly to the current plan in execution. Figure 1 depicts the overall organization of CORTEX. The domain-dependent modules might not necessarily be the same as those shown in the figure. Moreover, the complexity of these modules typically means that they will be internally organised as networks of software components. In the following sections, we provide further details about the DSR structure, the degree of closeness that the DSR provides between abstraction and imagery, and how the DSR evolves to reach a goal. Section 4 will provide a description of the most common agents employed in CORTEX.

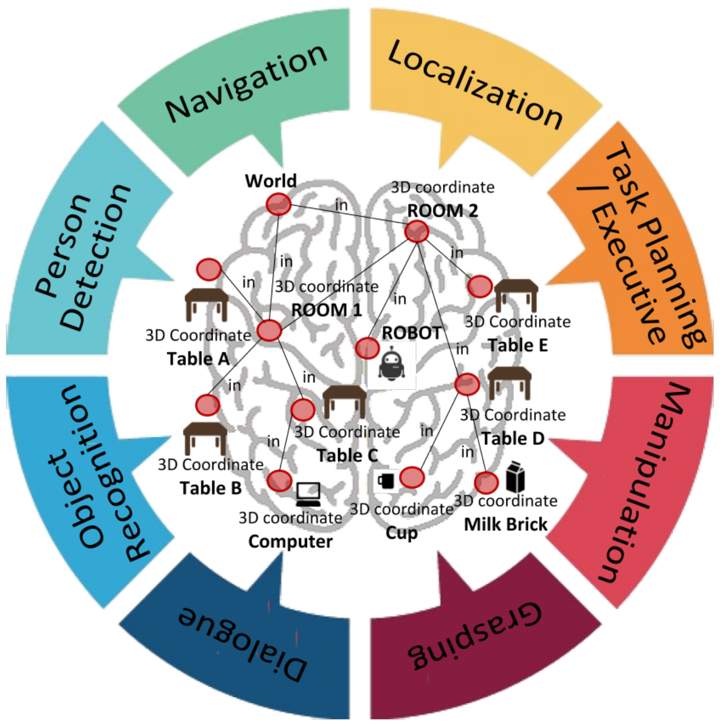


Figure 1: Overall organization of CORTEX. The architecture can be depicted as the confluence of a set of agents that communicate through a shared representation of the robot's internal state and its environment. Each agent can contribute by adding, modifying or removing elements in the DSR. Missions are processed into plans that are incorporated as intentions in the robot's node. Once in the DSR, all agents start working to make the represented world match the pending intention.

2.1. Deep State Representation

Deep State Representation (DSR) is a multi-labelled directed graph which holds symbolic and geometric information within the same structure. Symbolic tokens are stated as logic attributes related by predicates that, within the graph, are stored in nodes and edges respectively. Geometric information is stored in special edges as 4×4 homogeneous matrices and in symbols as properties. Figure 2 shows a simple example. The **person** and **robot** nodes are geometrical entities, both linked to the **room** (a specific anchor providing the origin of coordinates) by a rigid transformation. However, at the same time as we can compute the geometrical relationship between both nodes ($RT^{-1} \times RT'$), the **person** can be located (is_with) close to the **robot**. Furthermore, an agent can annotate that currently the **robot** is_not **speaking**, so it is also used to convey symbolic information. There is no a rigid format for the data that can be stored on the nodes of the DSR. For instance, they can include pointers to images, point clouds, or complete sentences (text files). As mentioned, it stores all the information that different agents need to provide a coordinated response.

As a hybrid representation that stores information at both geometric and symbolic levels, the nodes of the DSR store concepts that can be symbolic, geometric or a mix of both. Metric elements describe numerical quantities of objects in the world that can be structures, such as a three-dimensional mesh, scalars, such as the mass of a link, or lists like revision dates, for example. Edges represent relationships between symbols. Two symbols may have several kinds of relationships but only one of them can be geometric. The geometric relationship is expressed with a fixed label called RT. This label stores the transformation matrix (expressed as a Rotation-Translation) between the two. Therefore, the DSR can be described as the union of two *quivers*: the one associated with the symbolic part of the representation, $\Gamma_s = (V, E_s, s_s, r_s)$, and the other related to the geometric part, $\Gamma_g = (V_g, E_g, s_g, r_g)$. A quiver is a quadruple, consisting of a set V of nodes, a set E of edges, and two maps $s, r : E \rightarrow V$. These maps associate with each edge $e \in E$ its starting node $\mathbf{u} = s(e)$ and ending node $\mathbf{v} = r(e)$. Sometimes we denote by $e = \mathbf{uv} : \mathbf{u} \rightarrow \mathbf{v}$ an edge with $\mathbf{u} = s(e)$ and $\mathbf{v} = r(e)$. Within the DSR, both quivers will be finite, as both sets of nodes and edges are finite sets. A *path* of length m is a finite sequence $\{e_1, ...e_m\}$ of edges such that $r(e_k) = s(e_{k+1})$ for $k = 1...m-1$. A path of length $m \geq 1$ is called a *cycle* if $s(e_1)$ and $r(e_m)$ coincide. According to its nature, the properties of the symbolic quiver Γ_s are:

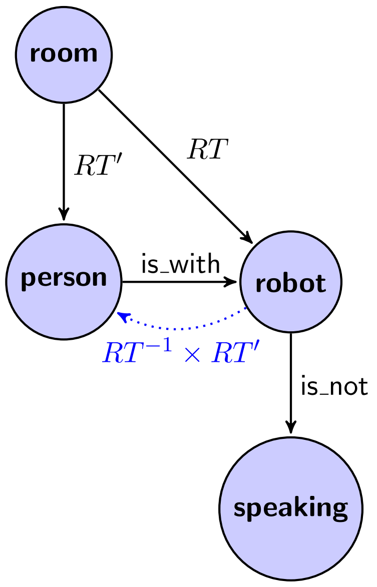


Figure 2: Unified representation as a multi-labelled directed graph. Edges labelled as is_with and is_not denote logic predicates between nodes. Edges starting at **room** and ending at **person** and **robot** are geometric and they encode a rigid transformation (RT' and RT respectively) between them. Geometric transformations can be chained or inverted to compute changes in coordinate systems.

1. The set of symbolic nodes V contains the geometric set V_g ($i.e.$, $V_g \in V$)
2. Within Γ_s there are no cycles of length one, that is, there are no *loops*
3. Given a symbolic edge $e = \mathbf{uv} \in E_s$, we cannot infer the inverse $e^{-1} = \mathbf{vu}$
4. The symbolic edge $e = \mathbf{uv}$ can store multiple values

According to its geometric nature and the properties of the transformation matrix RT, the characteristics of the geometric quiver Γ_g are:

1. Within Γ_g there are no cycles (acyclic quiver)
2. For each pair of geometric nodes $\mathbf{u}$ and $\mathbf{v}$, the geometric edge $e = \mathbf{uv} \in E_g$ is unique
3. Any two nodes $\mathbf{u}, \mathbf{v} \in V_g$ can be connected by a unique simple path
4. For each geometric edge $e = \mathbf{uv} = RT$, we can define the inverse of e as $e^{-1} = \mathbf{vu} = RT^{-1}$

Thus, quiver Γ_g defines a directed rooted tree or rooted tree quiver [17]. The *kinematic chain* $C(\mathbf{u}, \mathbf{v})$ is defined as the path between nodes $\mathbf{u}$ and $\mathbf{v}$. The equivalent transformation RT of $C(\mathbf{u}, \mathbf{v})$ can be computed by multiplying all RT transformations associated to the edges on the paths from nodes $\mathbf{u}$ and $\mathbf{v}$ to their closest common ancestor $\mathbf{w}$. Note that the values from $\mathbf{u}$ to the common ancestor $\mathbf{w}$ will be obtained by multiplying the inverse transformations. One example of computing a kinematic chain is shown in Figure 2.

2.2. Imagery within the DSR

It would be pretentious to say that the idea of integrating abstract and concrete concepts into a unique representation is a novel contribution in CORTEX. Apart from surveying previous work, Samuel Wintermute provided a meticulous analysis of the problems inherent to building an abstract representation of real world entities [36]. Basically, these problems can be summarised into (i) physical symbol grounding, (ii) perceptual abstraction, and (iii) irreducibility. The first is related to how to ground symbol tokens to real world entities, *i.e.* to percepts that can have a high dimensionality and, unfortunately, can vary under different conditions and with time. The problem is even more complicated, as different tokens could refer to the same real world entity. The perceptual abstraction problem is related to the difficulty of developing a single perception system able to induce appropriate abstract representations in any task the robot can address. Finally, the irreducibility problem is related to the resistance of some percepts being abstracted at all. Figure 3 depicts the scheme proposed by S. Wintermute for dealing with these problems. The architecture maintains two representations of the problem state: the abstract one, denoted by R_a, and the concrete one, R_c. In this architecture, the low-level perception, P_c, is provided to R_c. This R_c is not a strict representation of P_c, as it can be manipulated. This is a central issue in the architecture: the R_c representation can be manipulated through high-level actions A_a and, then, transformed into P_a. In the architecture, this happens regardless of whether the contents of R_c are real (from P_c) or *imagined* (after A_a).

The architecture in Figure 3 introduces a non-symbolic layer of concrete representations that involves the capacity to simulate the outcome of actions as executed by the robot in the real world. This simulation involves the generation of synthetic perceptive information that is fed back into the system, and is referred to as *imagery.* This layer is accessed by the abstraction layer and can provide geometric computations and continuous controllers that would, otherwise, be too detailed in time and space for an abstract symbolic

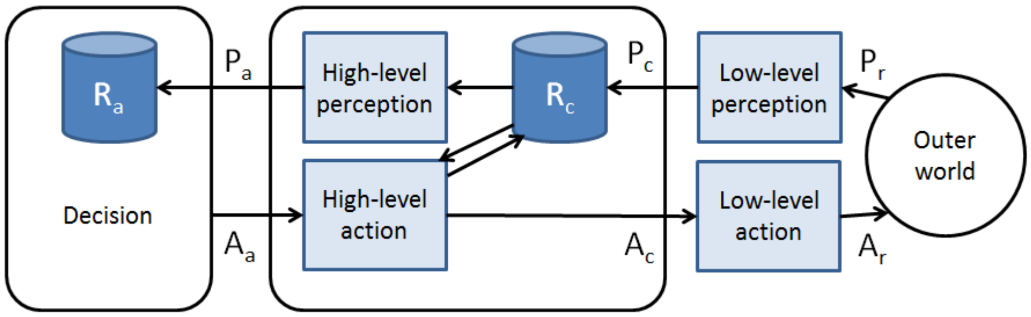


Figure 3: The imagery architecture by Samuel Wintermute [36].

representation. Imagery is used to compute pieces of information that are naturally expressed in continuous, metric domains. It is important to note here that alternate states are already imaged in the symbolic abstract domain using, for instance, a planning algorithm. Conversely, imagery provides simulation in concrete representations and, in doing so, mitigates the perceptual abstraction and irreducibility problems previously stated. Furthermore, it provides the abstract symbolic representations with complementary expressive power by linking task-independent symbols to task-dependent computations. The scheme depicted in the figure illustrates the typical organization of a three-layer architecture: there exists a strict separation between the high and low–level modules, which is interfaced by the *concrete representation*. This requires an a priori definition of these two domains, a delimitation that is not always easy to address. Moreover, this separation prevents a certain module being able to access data that is not available at its *own–level*, *i.e.* complex agents could require to access to both kinds of data.

Alternatively, Figure 1 shows the CORTEX proposal. Here, the DSR includes geometric data that can be accessed, such as kinematics chains or as a complete tree, to obtain a concrete geometric representation of the problem to be solved. This concrete representation can be updated and locally manipulated by the agents in the architecture. For instance, the `Grasping` agent (see Shelly robot at Section 5.4) has to plan safe trajectories toward the objects. While this task could be difficult to address without using a geometric representation of the world, thanks to DSR, the `Grasping` agent can use a local copy of the geometric data to simulate hundreds of possible courses of actions in super real-time. The best trajectory is then selected and executed using a generalised inverse kinematics algorithm. The `Navigation` agent is another good example of the use of these geometric data. If the robot has to move to a desired location, a path is required to safely drive to that target location. The `Navigation` agent, again, uses a local copy of the DSR to plan the motion of the robot, testing several candidate paths before selecting the best one. Furthermore, the use of a DSR including data at different abstraction levels makes it possible to execute more complex predictions about the world state. A planner working with a pure symbolic representation may be aware of the connections between states and symbols to infer the expected results of an action or to trace a plan. For instance, in the situation shown in Figure 4, the **robot** wants to ask the **patient** a series of questions. It can plan to move towards the **patient** and ask her to sit down on **chair_2**. However, as the DSR also includes lower levels of abstraction (all these nodes store, among other attributes, the position of the corresponding real entities on a bidimensional map), it can introduce physical constraints and relations in these computations. When moving towards the **patient**, it will navigate **very_close**[1] to **person_1** and **person_2**. These entities can be accessed and considered as 'obstacles' by the `Navigation` agent without employing additional interfaces or simulation components. However, the representation also informs that they are people. The response to an unexpected event such as the sudden movement of a person is reactively provided by the agents involved (*e.g.* the `Navigation` or the `Person` agents, see Section 4.5), which maintain the direct contact with the outer world via sensors and actuators. Simultaneously, this action is reflected in the DSR, generating, if needed, the posterior response of the slower deliberative agents (*e.g.* by changing the trajectory to avoid the person, or to politely ask her to move out of its way). If the decision is to change the path, this response can now be simulated before it is performed, as the imagery architecture by S. Wintermute proposes.

The management of the DSR is performed using a software library which provides the basic operations needed for all agents: adding or removing a node, adding or removing an edge, and modifying the attributes on a node or an edge. When an agent introduces or removes an item from the representation, the DSR is considered to have suffered a structural change. If this is not the case, the structure of the DSR remains unaltered and it has only been updated. The predicates that can be represented on the DSR are

[1]This *qualitative* token could be added to the DSR by the `Navigation` agent, using geometric information captured on-line

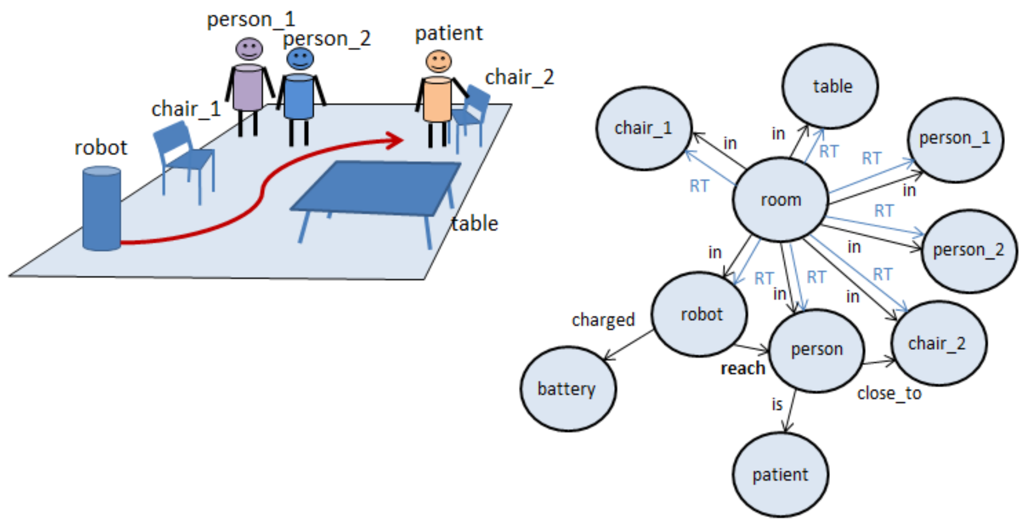


Figure 4: A simplistic representation of a specific situation for the DSR. Only some relationships are considered (see text for details).

not restricted to a specific vocabulary. Depending on the use case, we have chosen one or another database. For instance, when the robot must manipulate objects, the physical aspects are more relevant and the descriptions gain in details. Obviously, when the same agent is used within two different scenarios, the terms and substructures employed to encode the information are reused.

2.3. Reaching a goal within CORTEX

The execution of a task in CORTEX can be understood as a perturbation of an otherwise resting situation. A rough description of the dynamics would start when a new request enters the system such as, for example, a human utterance. The `Dialogue` agent processes the raw data and builds an internal object that is injected in the DSR as the robot's new desire. The new state of the DSR is propagated back to all agents but it is the `Executive`, a key agent, that is in charge of the execution monitoring, the one that processes the change as a new planning request. This request is usually interpreted as the need to change the world from its current state to a new one (see Figure 5 for a simple example showing the basic dynamics of goal pursuing). To do this, the `Executive` uses the available domain theory and a planning algorithm to explore possible sequences of actions and select the one that ends with the world (and the robot) in the desired final state. The plan is injected as a property of the symbol corresponding to the robot in the DSR graph, and is propagated back to the agents. Each agent selects from the updated graph the information it can interpret as a local goal, and starts its activity. When the final state is reached, the task is accomplished and the whole system goes back to its restful state. In pursuing their goals, perceptual and behavioural agents can inject their own sub-goals in the DSR for the `Executive` to compute a plan for them. The interleaving dynamics facilitate for a reactive-deliberative behaviour, in which top-down plans and bottom-up reactions to unforeseen situations can cooperate.

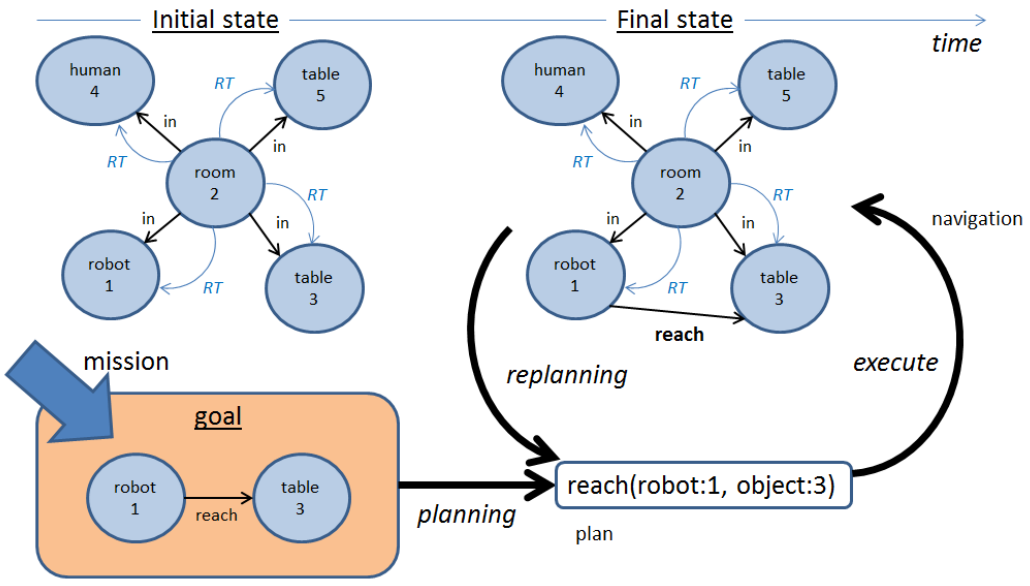


Figure 5: Example of the transition from an initial state triggered by a mission request. The mission is transformed into a goal which is formally represented as a sub-graph that shows the changes that have to be applied to the current state. In this case the required change is the inclusion of the *reach* predicate between nodes 1 and 3, meaning that robot 1 should be in reaching distance of table 3. The planning agent transfors the goal into an executable plan that is injected into the DSR. All agents are notified and the execution starts. If the new state (or states in case of a longer, more realistic plan) cannot be reached, the planner activates and introduces a new plan overcoming the current limitations. Finally, when the required state is reached, the mission is fulfilled and the current state becomes the initial state again.

All agents are connected to the DSR through a specific component, which includes the code required for sending modifications or updates, or for receiving changes in the global state. The internal execution of these components can be summarised by Algorithm 1.

```
Input: action from the Executive core
while (1) do
    subscribe to DSR updates;
    process { action };
    if DSR_changes then
        publish new DSR;
    end
end
```

Algorithm 1: High-level execution loop of an agent within CORTEX

CORTEX has agents for navigation, manipulation, person perception, conversation, planning, and monitoring, among others. The most relevant ones are briefly presented in Section 4. These agents interact with the DSR at different levels of abstraction, adding, removing or updating symbolic concepts or low-level geometric details. One of the main benefits of using cognitive architectures to program robots is the reuse of proven functionalities and a well-defined method to express each new experimental situation. In CORTEX, these functionalities are encoded in agents and they build on many years of continuous effort in many of the different AI fields needed to build a robot. We provide further details about this issue in Section 5.

3. Related work

To provide control and monitoring within a framework where several agents interact to satisfy certain goals, the blackboard model proposes an architecture consisting of:

- Independent agents, which work on their *portion* of the problem, dealing with their data, and writing their results in the blackboard.
- A control module, in charge of choosing the agents whose responses must be executed.
- The blackboard, the common solution space shared by the agents.

In this scheme, the blackboard represents a shared repository of goals, problems and partial results which can be accessed and modified by the agents running in parallel. The paradigm has been modified for working well in situations where many sources of information must be combined to solve complex real-time tasks. In the AIS architecture [14], two levels are differentiated: a cognitive and a physical level. The blackboard scheme is used as an implementation of the cognitive layer with a central working memory, where all behaviours (*methods*) write (perceptual inputs or previously executed behaviours) and to which all behaviours are connected to be triggered when certain conditions are fulfilled. The CERA-CRANIUM model [2] was inspired by the Global Workspace Theory (GWT). CERA is an architecture structured in layers, which uses the blackboard-based model provided by CRANIUM. Within CRANIUM, global workspace dynamics are modelled as an information processing system, whose input is the raw sensory data and whose output is a stream of artificial *qualia* (integrated multimodal representation). The filtering and integration processes endowed within this system are implemented as a service-oriented scheme. The processors in CERA use the workspace for sharing messages. Each piece of content (*percepts*) sent to the workspace is temporarily stored and accessible to any processor. Like CERA-CRANIUM, the ARCADIA architecture [3] is internally organised to drive attention (*i.e.*, computational resources) to a specific focus of attention. It is also organised following the guidelines of the GWT. Low-level processes are encapsulated in modules called *components*, which can be designed without theoretical restrictions. Each component communicates its accessible content and the focus of attention through *interlingua*, a repository of structured elements organised into topics (*worlds*). These elements are short pieces of information, consisting of a unique identifier, argument lists, and a symbolic name for the collection of arguments. Within the GWT-based architecture CHARISMA [9], the relatively small chunk of information that is currently deemed most important is broadcast to a host of processes, which work together to extract information, make associations, decompose problems, etc. The knowledge shared by these processes is organised as a semantic hyper network (SHYNE). SHYNE is composed of nodes (basic and complex) and links, characterised by a unique identifier. These identifiers are passed between processes. When a process updates a node and a certain threshold is exceeded, a new process can be automatically triggered. Using a different scheme, the CoSy proposal [33] divides the architecture into *sub-architectures*, which have specific local, working memories at their disposal.

Each sub-architecture contains a number of processing components which share information via their internal working memory. These local working memories are also writeable by an external single global process (the so-called *goal manager*).

Although the deep structure of the representation in the DSR may resemble the map models generated on the CoSy architecture [33], there are significant differences. On the one hand, CORTEX maintains a unified representation for all agents. Agents' internal memories are only used locally but all agents have access to the wider context representation created collaboratively among them. This point is important because if agents were to be designed as informationally encapsulated modules, as is partially developed in CoSy, then only the *goal manager* would have access to their outputs and would be the only one able to generate all the behaviours that need information coming from two (or more) different agents. CORTEX offers a more flexible scheme, allowing a sort of *porous modularity* in which agents have access to the representation maintained by all other agents working together. On the other hand, the DSR does not present the strict separation between layers (*e.g.*, metric map, navigation map, topological map, conceptual map) which characterises the CoSy architecture, and is also present in SHYNE. In the DSR, geometric and symbolic links are established between any nodes on the graphical model. In other words, there are not restrictions on linking geometric or symbolic concepts related to different *topics*. For instance, a **person** node can be linked to a **location** node, which is also linked to the robot pose, being able to determine the geometric relative position, but it is also possible to represent that the **robot is_not** currently **speaking** (see Figure 2).

4. CORTEX agents

The outline of the current structure of CORTEX is illustrated in Figure 1. To facilitate the implementation of CORTEX instances, coding is implemented on top of the RoboComp framework [24] which now includes a powerful code generator based on DSL descriptions, including communication parameters. The generated components are ready to run, and the generator can create agents with all the auxiliary functionalities, such as communication with the DSR, logging, debugging, user interface, binary building and documentation. The generated code is ready to be completed with the actual agent-specific features.

This Section provides a brief description of several functionalities that have been successfully implemented using CORTEX as agents. Other agents, such as those involved in the recognition of human emotions [7] or those more recently developed in charge of socially acceptable navigation policies in presence of groups of people [34], are only mentioned. As we show in Section 5, most of these agents have been reused in the different use cases.

4.1. Localization

Localization and mapping is the basic functionality that allows the robot to keep itself situated with respect to a known representation (map) of the environment. This agent combines two type of maps, an occupancy grid created by the SLAM algorithm *gmapping* [12], and a continuous map holding the geometric entities that exist in the environment. This second map is contained in the DSR representation and updated every time an agent changes it. The entities in this second map may belong to two different categories. The first category include fixed entities such as walls or doors. They are manually inserted by the users in an off-line process, defining a map with the distribution of rooms. This map is put in correspondence with the occupancy grid using a graphic tool created for this purpose. The second category of entities in the map stored in the DSR includes common objects, robots, and people. These entities are inserted into the specific deployment of CORTEX by the different agents. For instance, people are inserted by the `Person` agent (see Section 4.5). The ability to provide an intuitive framework for fusing geometrical information coming from different sources is one of the additional advantages of the DSR.

4.2. Navigation

Navigation provides the capability of global and local path planning and the robot displacement control. Whenever a step of the plan requires the robot to move to some location in the world, this agent takes control. It computes an obstacle-free path to the target and drives the robot through it, handling unforeseen obstacles. To do this, the agent uses three internal processes (components) that run concurrently. The first is a Probabilistic Road Map (PRM) planner [18] that uses a learnt graph of the free space to search for a path free of obstacles from the robot location to the target (see Figure 6). If the graph has more than one connected region or does not have a direct line of sight from the robot (or the target) to the graph, this component uses a Rapidly-exploring Random Tree (RRT) planner [20] to close the path. Once this raw path is found, it is shared with the other two components. The second component is connected to a laser sensor and takes the path to project it inside the laser field (reactive navigation). The aim is to check that the path remains free as perceived in real-time by the range sensor. It applies a variation of the *elastic band* algorithm [29] as explained in [13]. The current path (elastic band) is deformed according to a set of dynamic equations that generate a virtual force exerted by the objects.

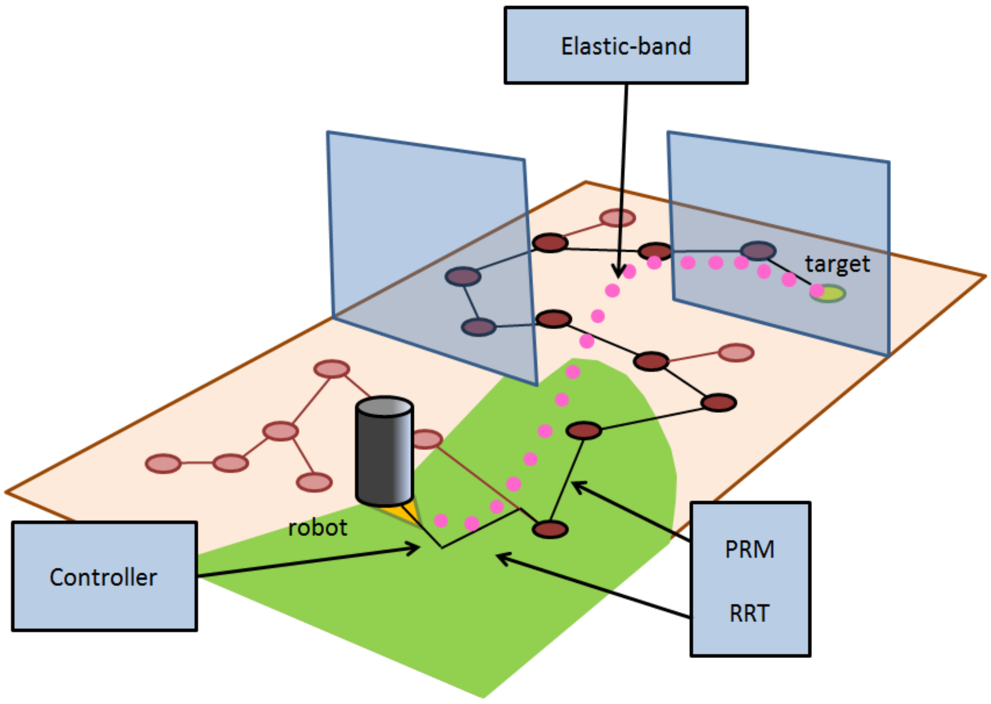


Figure 6: Graphical representation of the Navigation agent showing the PRM, the selected path and the projection of the path on the laser field.

Deviations from the map and dynamic obstacles can be easily avoided using this technique. To guarantee the feasibility of the path, the robot is virtually moved along the path checking whether its shape falls inside the free laser area. The third component implements a controller that drives the robot along the path. To achieve a smooth response it integrates a set of coupled equations combining variables such as the perpendicular distance to the path or the angle that the robot forms with the tangent to the path, curvature or current speed (see [13] for details). This scheme resembles the global CORTEX architecture, with the path (elastic band) taking the role of the DSR. The interaction between local and global path planning relies on a mental construction, the path, which is dynamically adapted to upcoming circumstances in the world.

4.3. Dialogue

Dialogue refers to the capability of the robot to maintain conversations with humans. Most of these conversations will be commonly limited to a list of human commands the robot must understand and internalize. The agent in charge of internalizing the dialogues can propose changes in the internal representation by means of the DSR. The robot must play an active role in the conversation guided by symbolic edges concerning its interlocutor. We can identify two stages involved in the speech recognition process: transcription and comprehension. Transcription can be carried out using third party services, like the Google speech recognition system, which allow input speeches to be successfully transcribed in different languages. In order to internalize the information retrieved from these transcriptions, they are parsed in order to extract elements such as entities, locations, or relevant semantic terms, using Natural Language Processing (NLP) techniques, like semantic role labelling, using toolkits like SENNA [8] or NLTK [21]. The versatility of CORTEX allows the comprehension stages to be implemented using different techniques. For instance, we may find a Bayesian classifier trained from the relevant terms following a Bag-of-Words approach. Other rule-based approaches have also been used to obtain a semantic representation of the input phrase, usually known as CFR (Command Frame Representation).

4.4. Manipulation

Manipulation is one of the most difficult skills in current social robots. For a mobile manipulator, *i.e.*, a mobile base with arms, the best approach seems to be that of using whole body inverse kinematics and dynamics approach [10]. Although some initial work is being done in this direction, the CORTEX's `Manipulation` agent is separated from the previously described `Navigation` agent. The coupling between the control of the robot's base and its arm(s) is done through the DSR structure and in some situations this can be a source of problems. For example, in precise manoeuvres where the robot has to grasp an object on a table, it might need to move the body sideways in order to achieve a safe grasping orientation of the hand. With a whole body, unified approach, inverse kinematics can compute a minimum displacement of both the body and the arm, which solves the problem. With separated controllers, it is easy to oscillate between small corrections coming from the body and the arm, each triggered by a limiting situation of the other. However, in the experimental situations we describe below, and in many others of similar complexity, the current separation into two agents is a good enough approach. Manipulation in CORTEX follows a similar schema to Navigation. A trajectory planner computes a path free of collisions from the current pose to a target position close to the object. This movement is to be executed blindly, like a fast saccadic movement of the eye.

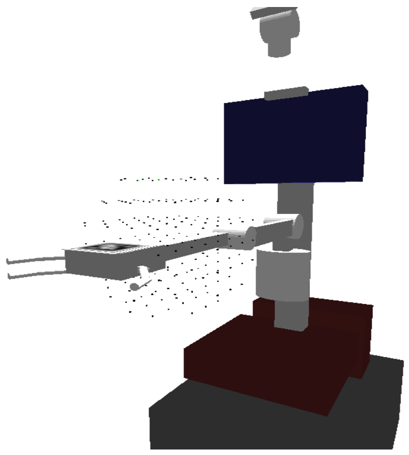


Figure 7: Graphical representation of the robot with the inverse kinematics grid. Each point in the grid represents a Cartesian pose, translation and orientation, where the angle values that correspond to this pose are stored.

The goal is to place the hand near the object and prepare it for a second phase that is driven by visual feedback. To compute this first trajectory the agent relies on an explicit, discrete representation of its arm's inverse kinematics. This function is stored as a grid in 3D space occupying the reachable volume of the arm (see Figure 7). Each cell in the grid holds a set of preferred orientations for the hand at that location, providing the inverse kinematic map $C_j = IK(c_i, o_k)$ where i runs through all the cells and k through the orientations in each cell i. To guarantee an initial free path, the agent uses its access to the robot's RGBD camera to compute the intersection between the grid and the cloud of 3D points provided by the sensor. Cells in the grid intersecting with points in the cloud are assumed to be in collision with objects in the world (a table, for instance) and are marked as non-navigable. The agent now searches the grid for a free path and shares it with a second module that drives the arm through the *way points* using a generalised inverse kinematics algorithm based on the Levenberg-Marquardt non-linear optimization method. Once the arm reaches the target position, the agent visually localizes the hand and the target in the same image frame. From this point, the agent starts a visual-servo approximation where arm calibration and object estimation errors are cancelled out as both elements get closer to each other [13]. When a final grabbing position is reached, the fingers are closed around the object and the transfer concludes.

4.5. Person detection and tracking

Person detection and tracking is a basic functionality for social robots. This task may rely on RGB-D sensors, omnidirectional cameras, or even external agents. In our case, we use Kinect or Xtion cameras placed in the robots. The standard libraries used to detect human skeletons are not sufficiently constrained and may produce some erroneous interpretations, as Figure 8 shows. The `Person` agent uses a model-based approach that filters the raw skeleton data through the kinematic model of a human torso. This post processing step increases precision and removes false positives [6].

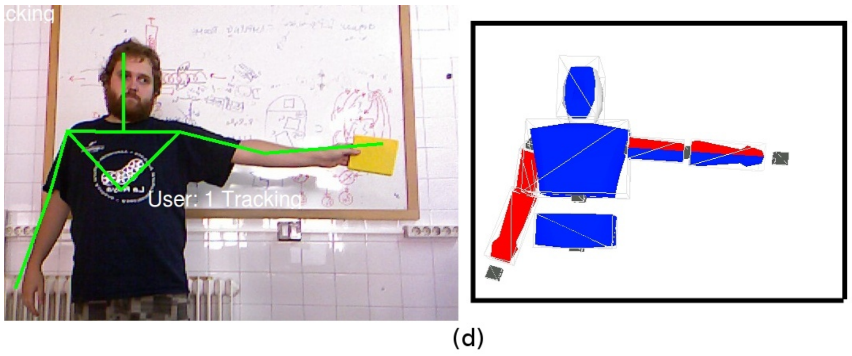


Figure 8: Person detection using a model-based approach. The skeleton obtained by the OpenNI library fails to estimate the real length of the arm, which is corrected by the fitted human model that is forced to respect kinematic constraints.

Once a person has been detected, the DSR is updated by a modification proposal to the `Executive` agent. These structural modifications introduce new elements in the graph, namely a node of type *person* for each human detected and additional nodes and RT edges to complete each person's parts. Meanwhile the person remains within the field of view of the camera, the

Person agent tracks his/her movements and a unique identifier is maintained in the DSR. If the person is suddenly lost and redetected, a new identifier will be assigned in the DSR. However, once a human is in the DSR, it is visible by all the agents, and the Person agent can update the DSR with an image of the face of the person. We have tested agents that use this image for face emotion recognition, face-based gender and age estimation [27] or face recognition [11]. When the Face_recognition agent is present, the identification of the human is based on the results that this agent inserts in the DSR –a specific attribute associated to the node *person*. When the Face_emotion agent is present, the *person* node can be attributed with the recognised emotion [7].

A case of special interest occurs in the relatively new area of social navigation, where the Navigation agent is required to interpret obstacles in a different manner depending on wheter they are humans or inanimate objects[34]. In the first case, it must obey some additional rules that involve, for example, stopping and asking for any commands, or performing an avoidance manoeuvre with extra clearance space. The human detected by this agent is visible and accessible to the Navigation agent at a very low coding cost. This is one of the main advantages of using a software architecture that provides, among other things, a versatile communication mechanism.

4.6. Executive

Robots are often expected to be able to achieve more than a small set of different missions. Generating and supervising the corresponding plans using automated planning techniques is generally faster to develop and more robust than embedding them in state machines. In CORTEX, these tasks are handled by an Executive module in charge of invoking an automated planner and monitoring the execution of the plans, ensuring that the rest of the agents have access to the current plan. The current implementation uses a graphical planning domain definition language named Active Graph Grammar Language (AGGL) which can later be transformed to PDDL (Planning Domain Definition Language) if desired. These domains are used to describe the transformations that can be implemented in the world model (see Figure 9). The language and the planner used were proposed in [25].

Some of these transformations are associated to behavioural or perceptual actions, but they can also be *deliberative actions* in which no physical action is involved (*e.g.*, the manipulation agent is in charge of marking the appropriate table to leave the objects in the robot's gripper, an *action* that has no

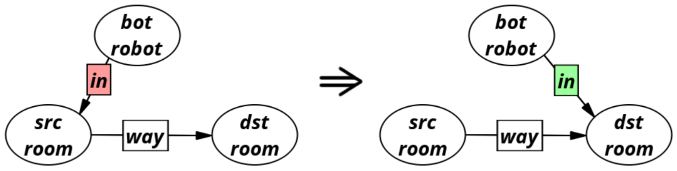


Figure 9: Transformations have left-hand and right-hand sides. The left-hand side describes the preconditions for the transformation to be executed, whereas the right-hand side describes how the pattern in the left-hand side would change. The action at hand describes how the world model is modified when the robot moves from one room to another. Note that variable identifiers are used in rule definitions so that they can be used with any set of symbols satisfying the conditions.

direct physical consequence). Such a domain is used by the planner to find a sequence of actions that would take the world model from the current state to another one in which the goal is satisfied. To avoid wasting computational time, instead of finding a plan with every modification made to the world model, before querying the planner for a new plan, the `Executive` checks first wheter the current plan (or a version with the first actions removed) is still valid.

5. Use cases with CORTEX

The success of a robotic architecture depends on its capability for providing robots with skills for performing complex and heterogeneous tasks. In the following, we introduce some robotic developments relying on CORTEX. One of the most promising features of CORTEX is its adaptation to different robots and objectives. Indeed, three different research groups belonging to different Universities (UMA, UNEX, and UCLM[2]) opted for CORTEX as their architecture. These groups have actively collaborated in different research projects, but they differ in their skills, and more significantly in their physical platforms. Specifically, UMA and UNEX have traditionally designed and assembled their own robots [7], while UCLM has historically worked with standard platforms [26].

[2]Universities of Málaga (UMA), Extremadura (UEX) and Castilla La-Mancha (UCLM)

5.1. The welcome home situation

The objective of this use case [11] is to have a social robot capable of welcoming visitors in an indoor environment. This use case involves a standard robotic platform (a PeopleBot), and an external Velodyne sensor. As soon as a new visitor is identified by the Velodyne sensor, the robot approaches and welcomes him or her. Then, the robots offers itself to accompany the visitor to a staff employee or to find an object in the environment. After a short dialogue, the robot performs the desired action and is ready to welcome a new visitor. The use case is shown in Figure 10. It must be noted that it illustrates the 'good' execution of the plan. Unexpected events (*e.g.*, losing the person, low-battery detection) can occur during the execution of any trial, but the robot adapts the plan to the new situation. Figure 12.a shows the robot while performing this use case.

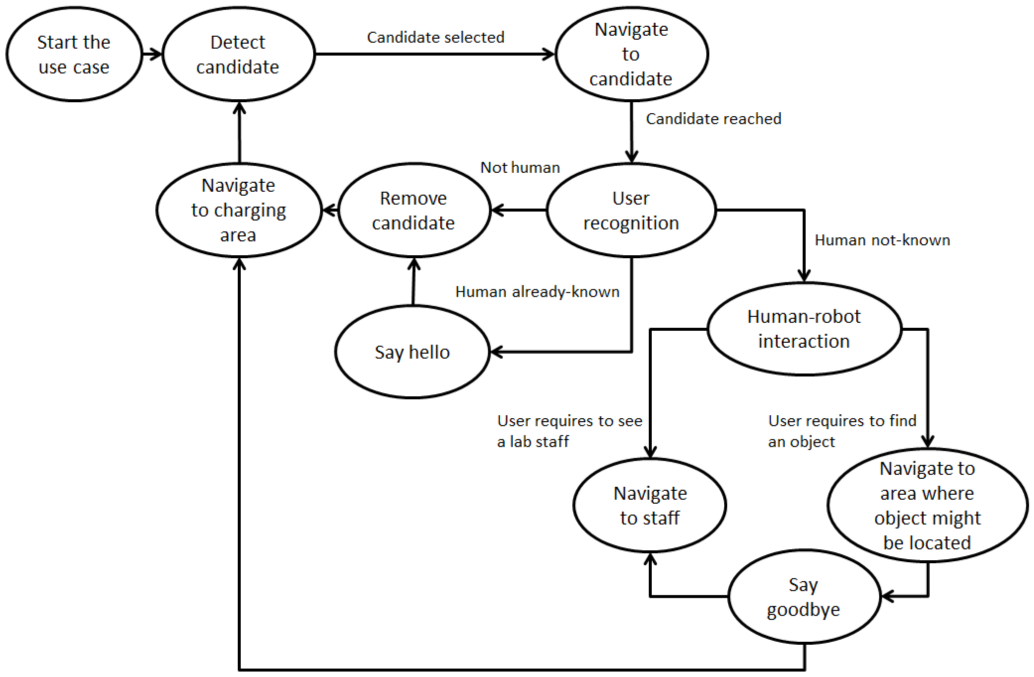


Figure 10: Overall scheme of the use case for the *welcome home* scenario.

The use of CORTEX was essential for the development of the welcome home situation. The DSR was successfully used to integrate the data sensed by the robot with those obtained from the Velodyne sensor. In addition to a problem domain definition, the use case requested the inclusion of agents for people detection, user recognition, localization and mapping, navigation, and speech recognition and generation. Some of these were briefly presented in Section 4.

5.2. The advertisement robot scenario

This experiment involves an advertisement robot, Gualzru, built to approach people, and try to convince them to visit a panel where new products are exhibited for marketing purposes [31]. The conviction stage involves recognising the age and gender of the interlocutor. This information is exploited to generate user-oriented dialogues, which would increase the probability of convincing the human to go to the panel.

The robot Gualzru (Figure 13.a) was specifically designed and built for advertising purposes. In this case, some of the requested tasks were solved using third-party libraries or SDKs. For instance, the speech recognition relies on the Microsoft Speech SDK, which allows the use of grammars trained from a specific corpus. These corpora can incorporate domain specific words (*e.g.*, the name of products to be advertised), which may be mandatory to properly understand the human speech. The Microsoft Speech SDK was integrated in Gualzru by adding a CORTEX agent running on a Windows computer. This speech recognition agent, in conjunction with other agents for skeleton detection and face identification, is directly connected to a Kinect sensor, and helps to reduce the workload of the main computer thanks to a distributed processing scheme. In any event, the `Dialogue` and `Person detection and tracking` agents are those presented in Section 4. This example shows that the adoption of the CORTEX architecture simplified the overall integration of all the different modules, as well as the problem definition thanks to the use of the DSR.

The execution of the use case in real scenarios was evaluated through questionnaires administered to a representative group of people after interacting with the robot. The questionnaire uses Likert scale items, although it uses six levels, from 0 to 5, to remove the neutral option (middle point) [30]. Table 1 shows the results for the questions related to the performance of the cognitive architecture: Specifically it includes a collection of questions arranged in three blocks (conversation, interaction and general sensations). They consider the participation of Gualzru in two fairs in Madrid or Málaga (Spain) in December 2014 and March 2015, respectively. Contrary to the

Table 1: Questionnaire results (76 tests) for a subjective evaluation of a human-robot interaction experiment

Question	$\bar{x}$	σ
Did you understand what the robot said?	4.31	1.17
Do you think the robot understood you?	3.76	1.23
Did the robot get blocked?	1.39	1.23
Was the interaction natural?	3.04	1.27
Was the conversation fluent?	3.11	1.21
Did the robot seem to be tele-operated?	1.23	1.49
Was the touch screen useful for the interaction?	4.17	1.28
Did you enjoy the experiment?	4.61	0.63
Would you like to repeat?	4.52	0.72
Would you recommend it to other people?	4.73	0.61

results reported in [30], these results were obtained using the CORTEX architecture. As depicted in the Table 1, the subjective experience of people interacting with the robot was satisfactory. The robot proved to be a good *added value* to attract people to the advertising panel. In general, the overall system usage was quite reliable and robust. The interaction was evaluated as natural and fluent, despite there being more than 20 software components running within CORTEX.

5.3. The CLARC scenario

Comprehensive Geriatric Assessment (CGA) procedures evaluate the degree of autonomy of elderly people. CGA procedures involve subjective interviews, but also include physical and cognitive tests, that could be autonomously performed by a robot. The CLARC proposal[3] was selected within a challenge proposed by the ECHORD++ EU project to automatize the execution of these tests. The software architecture of the platform is currently an implementation of CORTEX. This allows to encode the whole CGA session using Automated Planning, which works over symbolic tokens in the

[3]*http://echord.eu/essential_grid/clarc/*

DSR to autonomously plan, drive, monitor and evaluate the session. Additionally, in parallel, the geometric information allows the robot to navigate, detect people and track their movements, and to dialogue with the users. The CLARC platform (Figure 13.b) was developed by Metralabs GmbH. For localization and navigation, it uses the CogniDrive software running over the MIRA middleware[4]. This is not a problem for CORTEX, and a specific bridge has been coded to connect these *external agents* to the DSR. Figure 11 shows the instantiation of CORTEX on the CLARC robot. Surrounding the DSR, we can find seven agents and the `WinKinectComp` agent, the same module presented in the Gualzru robot. Apart from the aforementioned `CogniDrive` agent, the high-level planning is provided in CLARC by a second external agent: the `PELEA` one [1]. Significantly, there is no `Executive` agent in CLARC; being the deliberative response coordinated by the `PELEA` agent through annotations in the DSR (see [35] for further details).

The CLARC robot is able to navigate to a specific room from the charging room when a doctor calls it on a dedicated web application. Once in the room, the robot is able to introduce itself to the patient and their caregiver and conduct the desired sequence of tests. Currently, CLARC is able to autonomously perform three CGA tests: Barthel, Minimental, and GetUp-and-Go [35]. The first ones are questionnaire-based tests, where the major challenge is to design an interaction scheme that allows the engagement between the robot and the elderly person. The verbal channel has been augmented with a touch-screen and an external tablet-based device. Responses acquired from all these channels are easily fused using the DSR, as all of them are attributes of the same conceptual entity. The GetUp-and-Go test requires the robot to track the movement of the patient while performing specific physical exercises. The sequence of tests and their parameters can be adjusted on the fly for each user. Carefully designed robot-clinician interfaces allow the expert to check sessions, analyse results or modify data using a web-based application. Although the project is not yet finished, the current evaluations are showing the robustness of the architecture [35].

5.4. The Bring me X scenario

This final use case is performed by the robot Shelly, who occupies an apartment adapted for people with limited personal autonomy. The robot

[4]http://www.mira-project.org/joomla-mira/

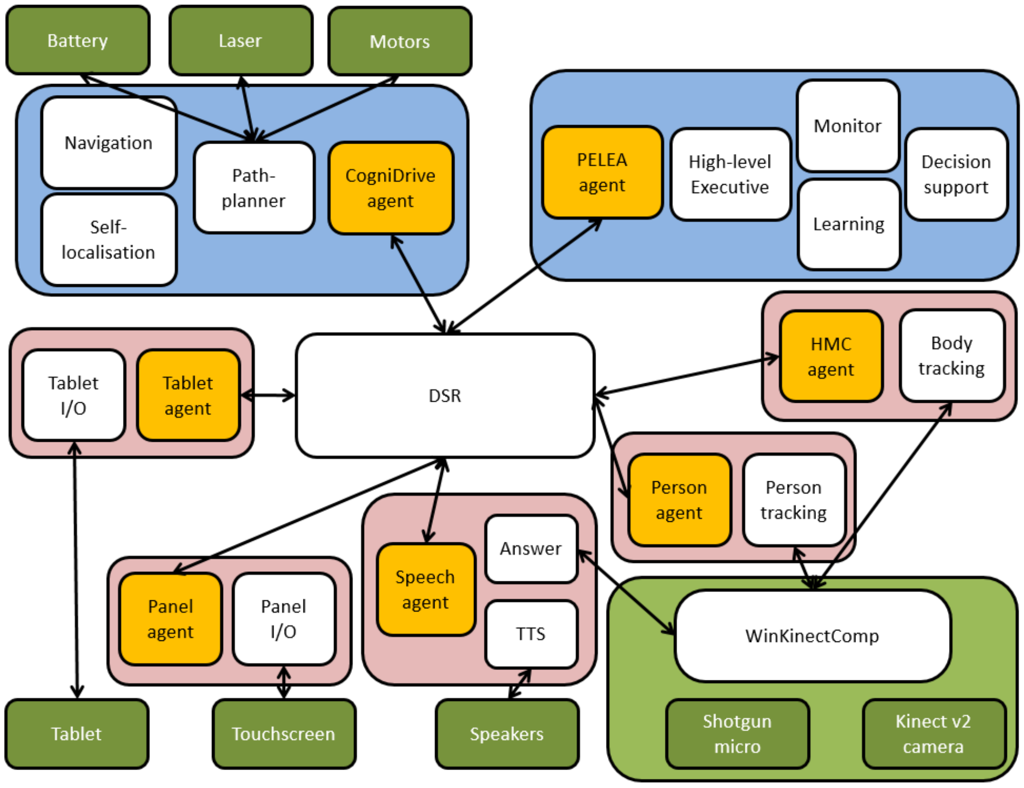
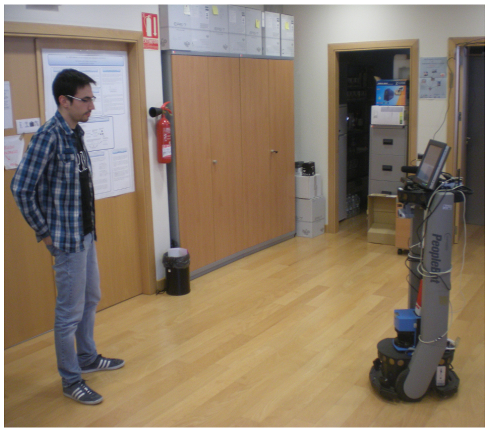


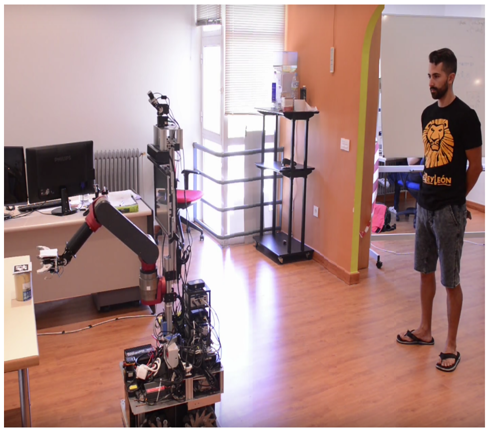
Figure 11: Instantiation of CORTEX on the CLARC robot [35]

attends requests from humans, who ask the robot to bring them objects located on tables in the apartment (Figure 12.b).

The execution of this use case requires coordination across all agents in the architecture: the `Localization` agent is required for the robot to know the relative position of the objects from its point of view; the `Navigation` agent makes the robot change rooms and approach the objects and the persons with whom the robot needs to interact; the `Dialogue` agent is used to interpret the commands from the humans; the `Person detection` agent is required to detect the persons and obtain their pose; the `Manipulation` agent is used to grasp the objects and transfer them to the humans. In a public demonstration, on September 2016, Shelly was able to successfully deliver the requested object to different persons 40 times in a row, proving the robustness and maturity of the underlying hardware and software architecture.

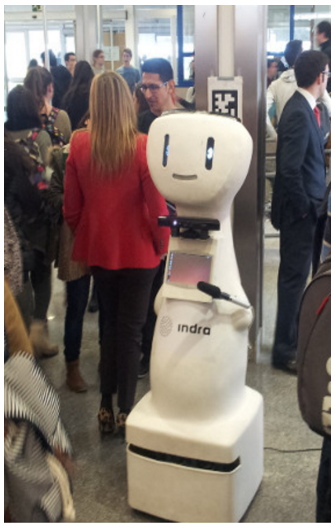


(a) Welcoming

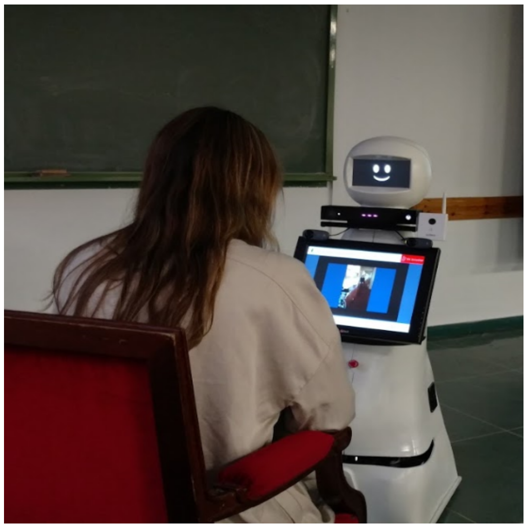


(b) Bringing an object

Figure 12: Robots working in the welcoming and Bring_Me(x) scenarios.



(a)



(b)

Figure 13: (a) Gualzru and (b) CLARC working in their corresponding scenarios

Table 2: Usage of the same agents in different scenarios

Agent	Welcome	Advertisement	CGA-tests	Bring_me
Localization	yes	yes	no	yes
Navigation	yes	yes	no	yes
Dialogue	yes	yes	yes	yes
Manipulation	no	no	no	yes
Person detection & tracking	yes	yes	yes	yes
Executive	yes	yes	no	yes

6. Discussion

In the previous section, four use cases involving different robots and scenarios were described. The success of all four experiences was possible, despite a limited amount of human and material resources, because of the reusability provided by the CORTEX architecture. For instance, Table 2 provides a snapshot of the usage of the agents described in Section 4 in the use cases presented in Section 5. As mentioned in the description of these use cases, other agents that solve specific tasks that are not listed in this table. Despite the use cases and hardware platforms (including sensors or actuators) being very different, the algorithms built on the agents have been reused. Obviously, this usage is conditioned by the platform itself and the final needs of the robotic application. In this sense, it can be noted that the `Manipulation` agent is only integrated in the *Bring_me* scenario.

However, Table 2 shows that most of the code was reused. This allowed the efforts to be directed to extend and modify the domain knowledge in order to capture the specifics of each scenario. In this section we argue in favour of using a cognitive architecture in real-world robotics, despite most current robots being controlled by state machines. The reasons for avoiding cognitive architectures might be related to the complexity and effort of developing a new architecture or even of using an existing one. Although new architectures tend to be built on top of well-proven and documented frameworks (*e.g.*, RoboComp, SmartSoft, Genom, OROCOS, etc.), the unavoidable complexity derived from very large code sets, maintenance effort, diversity of involved research topics, variety of available theories and potential algorithms, hinders the development process and quickly exhausts the groups' resources. Another possible reason for roboticists to avoid using advanced architectures is that they often focus on optimizing a single task instead of developing a robot able to perform multiple missions that have not been explicitly preprogrammed. Nevertheless, we believe that using, sharing and comparing cognitive architectures is the way to advance in the development of more complex autonomous robots that can operate with humans.

We will consider two reasons here to support our argument: reusability and planning capability. Reusability concerns the structural aspects of the architecture that are completely independent of the problem domain, namely:

- the libraries handling the DSR, which allows a hybrid, metric and symbolic representation of the world model held by the robot;
- the `Executive` agent, which can be configured to use any planning domain definition file, is completely domain-independent;

and also the elements that are domain-dependent but can be used by different robots with different goals:

- the domains: to a large extent, the four robots presented here reused the same domain (39 out of 53 actions), with the exception of some robot-specific actions (*i.e.*, only Shelly is able to grasp and deliver objects);
- the agents, which have been written and tested in many experimental scenarios. In most cases agents depend on lower-level modules that are dependent on the robot, but the agents themselves are often independent.

Reusability is considered to be of great importance because it reduces the time spent on developing the software and improves its robustness. A good architecture should provide a set of agents implementing robotic functionalities that are robust, efficient and, eventually, replaceable by improved versions of themselves. CORTEX is based on a technology of distributed software components that provides a reliable solution to the problem of connecting decoupled elements in a system using public interfaces [15]. Using component-oriented technologies usually entails the additional effort of maintaining the middleware-related code. As mentioned at the beginning of Section 4, the current implementation of CORTEX is based on the RoboComp framework, which offers tools to automatically generate such code, reducing the development time and the probability of programming errors [32].

RoboComp is a distributed, component-based programming framework [24], which was developed entirely by the groups involved in this paper and it is now in a mature, usable state.

The planning capability concerns the mandatory cognitive feature of using an explicit knowledge representation of the problem domain, along with a planning algorithm to create, in run-time, programs that, when executed, drive the robot to complete the mission assigned by the human. Task planning is a more general solution than state-machines and a practical one if there exists a cognitive architecture providing the underlying mechanisms for it to work. As explained in Section 2, the execution flow in CORTEX is distributed among the intervening agents and has top-down and bottom-up components. In designing the architecture there is a tension between centralized and distributed control. The mission or human request is handled by the `Dialogue` agent and passed in a normalized format to the `Executive/Planning` agent. This agent computes a plan that solves the requested task and distributes it to the other agents. The complete plan is propagated to the agents and it is their responsibility to choose the order in which it is executed. Agents must attend the restrictions imposed by the set of preconditions that precede each rule but if they perceive an opportunity to jump to the final, desired state, they will do it. The classic example here is the one in which the robot is commanded to fetch the butter inside the fridge in the kitchen but it happens to be on a table by the entrance door. Supposing it sees the butter when passing by, should it not pick the butter up and deliver it to the human, prematurely ending the task? A reactive agent monitoring all critical conditions in the plan does that. So, plan order is not strictly controlled by the `Executive`, it is only suggested.

Another interesting consequence of having a formalized domain knowledge is that the process of learning the external world can be filtered by what can be interpreted by this knowledge. All agents with perceptive attributions can inject information into the shared graph, opening the possibility of degrading it. When an agent decides to introduce a structural change in the graph, there should be a mechanism to guarantee that the new modified graph is valid within the limits imposed by the grammar of transformation rules[5]. Currently, the `Executive` validates all structural changes through

[5]No *elephants in the kitchen* could be hallucinated because there are no worlds derived from the grammar that contain elephants.

a model checking procedure. This procedure is actually a planning operation of the sequence of valid transformations between the current state of the world and the new state obtained after applying the structural change. If there exists such a valid sequence of transformations then the change is accepted and the new graph is propagated back to the agents. The alternative choice would be to translate the model checking responsibility to the agents, eliminating the processing burden imposed by the model checking in the `Executive` agent. This solution, tested in the CLARC scenario, is not without its problems since now the agents would need access to the whole domain knowledge and to a planning algorithm. Currently, we are actively investigating this problem in order to endow CORTEX with a reasonable option or set of options.

Another crucial feature of CORTEX being redesigned and extended is its emulation capability. The central, hybrid representation that holds symbolic and concrete information is used to emulate future courses of action at different levels. At the agent level, the contents of the graph can be extracted as a local copy and used to emulate how a situation might evolve when a series of actions are taken. This is the case of trajectory planning for navigation paths, arm movements or objects grasping, where the representation of both the external and internal world is used as a replacement of the real measurements, and the forward and inverse algorithmic models inside the agents are used to manipulate those representations and predict realistic outcomes. Another level of difficulty arises if an emulation is requested by the task-level deliberative agent that requires the intervening of several agents. In fact, planning with symbolic tokens provides a real form of emulation, but it is always restricted to the abstract domain defined by the transformation rules. This kind of emulation does not alters the working memory (DSR). However, what is needed and yet remains elusive is a global emulation where all agents are involved over a temporally detached copy of the working memory. This copy would evolve towards the future while the original DSR remains synchronized with the external world. At some point in the near future and after some insight has been extracted from the copy, it is destroyed liberating the valuable resources. This functionality is part of our current efforts to make CORTEX a fully predictive architecture.

7. Conclusions and future work

This paper introduces the theoretical foundations of CORTEX: the need for a shared representation that can provide the functionalities of standard blackboard models but which, at the same time, can integrate symbolic and geometric representations. Furthermore, the paper evaluates the appropriateness of this unified representation for solving real use cases. This evaluation was carried out based on the adoption of the CORTEX architecture in heterogeneous robotic projects. We consider that the generation of an architecture suitable for use by different research groups is a highly important contribution.

Apart from the previously mentioned emulation of complex tasks by a set of agents, another interesting possibility that we are exploring is the idea of giving more deliberative resources to traditional perceptive-action agents. We have observed that for these agents to correctly interpret the actions and state transitions that are stated in the plan, they have to reason about the requested actions along with their local state and the global context provided by the DSR. We have initiated research to provide all agents with domain specific symbolic knowledge and a planning algorithm. With these resources, agents will be able to reason about what the planned task level wants and about how to achieve it, maintaining the necessary level of reactivity to unforeseen events.

One of the most interesting problems in an agent-based architecture that uses a common representation is how they communicate and get access to the shared structure. The central, shared representation has two important benefits: it decouples the interactions so agents do not need to directly communicate with each other (nor even know of their existence); and, given a common naming system, the shared structure provides an extended context for each agent to make better, more global, choices. However, global sharing comes at a price and involves certain potential synchronization and efficiency issues. DSR is currently implemented as a client-server architecture in which one agent (usually the Executive, but a dedicated one could also do the job) hosts the graph and receives requests from the agents to modify the nodes' attributes or to make structural changes. Nevertheless, we are already working on other technologies borrowed from the field of computer games and specially their online implementation as distributed networking environments [22, 23, 28, 5]. There are many available technologies ranging from networked-server to peer-to-peer architectures that can be used to improve our current implementation. This kind of solution would take the form of a distributed graph without a central, server-based, existence that would exist as (partially) synchronized copies in all participating agents.

Finally, our view of CORTEX for the near future includes all the mentioned features under current research to provide a truly predictive architecture, capable of computing future courses of action at different levels of abstraction and of using the outcome of those computations to improve the choice of the next action.

Acknowledgments

This work was partially funded by FEDER funds and the Spanish Government (MICINN) through projects TIN2015-65686-C5; and by the "Red de Agentes Físicos" Excelence Network TIN2015-71693-REDT and Group Consolidation Grants from the Regional Goverment of Extremadura, 2016. This work was partially funded by the European Union ECHORD++ project (FP7-ICT-601116). The authors warmly thank the members of the "Amis du Living Lab" community for their participation in the CLARC research.

8. References

[1] Vidal Alcázar, César Guzmán, David Prior, Daniel Borrajo, Luis Castillo, and Eva Onaindia. PELEA: Planning, learning and execution architecture. In *Proceedings of the 28th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG'10)*, pages 17–24, Brescia (Italia), December 2010.

[2] R. Arrabales, A. Ledezma, and A. Sanchis. Simulating visual qualia in the cera-cranium cognitive architecture. In *From Brains to Systems: Brain-Inspired Cognitive Systems 2010*, Advances in Experimental Medicine and Biology, pages 223–238. Springer, 2011.

[3] P. Bello, W. Bridewell, and C. Wasylyshyn. Attentive and pre-attentive processes in multiple object tracking: A computational investigation. In *Proc. 38th Annual Meeting of the Cognitive Science Society*, pages 1517–1522, Philadelphia, PA, USA, 2016.

[4] P. Bustos, L.J. Manso, J.P. Bandera, A. Romero-Garcés, L.V. Calderita, R. Marfil, and A. Bandera. A unified internal representation of the outer

world for social robotics. In *ROBOT (2)*, volume 418 of *Advances in Intelligent Systems and Computing*, pages 733–744. Springer, 2015.

[5] E. Buyukkaya, M. Abdallah, and G. Simon. A survey of peer-to-peer overlay approaches for networked virtual environments. *Peer-to-Peer Networking and Applications*, 8(2):276–300, 2015.

[6] L.V. Calderita, J.P. Bandera, P. Bustos, and A. Skiadopoulos. Model-based reinforcement of Kinect depth data for human motion capture applications. *Sensors*, 13(7):8835–8855, 2013.

[7] F. Cid, J. Moreno, P. Bustos, and P. Núnez. Muecas: a multi-sensor robotic head for affective human robot interaction and imitation. *Sensors*, 14(5):7711–7737, 2014.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, 2011.

[9] M. Conforth and Y. Meng. Embodied intelligent agents with cognitive conscious and unconscious reasoning. In *Proc. of the Int. Conf. on Brain-Mind*, pages 15–20, Brain-Mind Institute, 2012.

[10] J. J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley Longman Publishing Co., 1989.

[11] D. González-Medina, A. Villena, C. Romero-Gonzlez, J. Martínez-Gómez, L. Rodríguez-Ruiz, and I. García-Varea. The welcoming visitors task in the apedros project. In *WAF 2016 - Proceedings of the XVII Workshop de Agentes Físicos*, pages 17–25, 2016.

[12] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *Informatica*, pages 1–12, 2007.

[13] M. Haut, L.J. Manso, D. Gallego, M. Paoletti, P. Bustos, A. Bandera, and A. Romero-Garcés. A navigation agent for mobile manipulators. In *Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 2*, pages 745–756, Cham, 2016. Springer International Publishing.

[14] B. Hayes-Roth. A domain-specific software architecture for a class of intelligent patient monitoring agents. *J. Exp. Theor. Artif. Intell.*, 8(2), 1996.

[15] M Henning, M Spruiell, et al. Distributed programming with ice; zeroc. *Inc.: Jupiter, FL, USA*, 2013.

[16] O. Holland and Goodman. R.B. Robots with internal models: A route to machine consciousness? *Journal of Consciousness Studies*, 10(4):77–109, 2003.

[17] V. Katter and N. Mahrt. Reduced representations of rooted trees. *Journal of Algebra*, 413:41 – 49, 2014.

[18] L. E. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[19] I. Kotseruba, O.J. Avella Gonzalez, and J.K. Tsotsos. A review of 40 years of cognitive architecture research: Focus on perception, attention, learning and applications. *CoRR*, abs/1610.08602, 2016.

[20] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[21] E. Loper and S. Bird. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70. Association for Computational Linguistics, 2002.

[22] M. Lozano, P. Morillo, J. M. Orduna, V. Cavero, and G. Vigueras. A new system architecture for crowd simulation. *Journal of Network and Computer Applications*, 32(2):474–482, 2009.

[23] M. R. Macedonia and M. J. Zyda. A taxonomy for networked virtual environments. *IEEE Multimedia*, 4(1):48–56, 1997.

[24] L. J. Manso, P. Bachiller, P. Bustos, and L.V. Calderita. RoboComp: a Tool-based Robotics Framework. In *Proceeding of the 2nd International Conference on Simulation, Modelling and Programming for Autonomous Robots (SIMPAR)*, volume 6472, pages 251–262, Darmstadt, Germany, 2010.

[25] L.J. Manso, P. Bustos, P. Bachiller, and P. Núñez. A perception-aware architecture for autonomous robots. *International Journal of Advanced Robotic Systems*, 12(174):13, 2015.

[26] J. Martínez-Gómez, J. A. Gámez, I. García-Varea, and V. Matellán. Using genetic algorithms for real-time object detection. In *Robot Soccer World Cup*, pages 215–227. Springer, 2009.

[27] J. Martínez-Gómez, R. Marfil, L.V. Calderita, J.P. Bandera, L.J. Manso, A. Bandera, A. Romero-Garcés, and P. Bustos. Toward Social Cognition in Robotics : Extracting and Internalizing Meaning from Perception. In *Workshop of Physical Agents*, pages 1–12, Leon, Spain, 2014.

[28] M. Naef, E. Lamboray, O. Staadt, and M. Gross. The blue-c distributed scene graph. In *Proceedings - Virtual Reality Annual International Symposium*, pages 275–276, 2003.

[29] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 802–807. IEEE, 1993.

[30] A. Romero-Garcés, L. V. Calderita, J. Martínez-Gómez, J. P. Bandera, R. Marfil, L. Manso, A. Bandera, and P. Bustos. Testing a fully autonomous robotic salesman in real scenarios. In *Procs. of the 9th IEEE International Conference on Autonomous Robots Systems and Competitions (ICARSC 2015)*, pages 124–130. IEEE, 2015.

[31] A. Romero-Garcés, L.V. Calderita, J. Martínez-Gómez, J.P. Bandera, R. Marfil, L. J. Manso, A. Bandera, and P. Bustos. The cognitive architecture of a robotic salesman. In *Procs. of the XVI Conferencia de la Asociación Espaola para la Inteligencia Artificial (CAEPIA 2015)*, Albacete, Spain, November 2015.

[32] A. Romero-Garcés, L.J. Manso, M.A. Gutiérrez, R. Cintas, and P. Bustos. Improving the life cycle of robotics components using domain specific languages. In *Proc. of Int. Workshop on Domain-Specific Languages and models for ROBotic systems (DSLRob'2011)*, pages 1–9, 2011.

[33] K. Sjöö, H. Zender, P. Jensfelt, G. M. Kruijff, A. Pronobis, N. Hawes, and M. Brenner. The Explorer system. In Henrik I. Christensen, Geert-

Jan M. Kruijff, and Jeremy L. Wyatt, editors, *Cognitive Systems*, volume 8 of *Cognitive Systems Monographs*, pages 395–421. Springer Berlin Heidelberg, 2010.

[34] A. Vega, L.J. Manso, P. Bustos, P. Núñez, and D.G. Macharet. Socially acceptable robot navigation over groups of people. In *IEEE Conference on Robot and Human Interactive Communication, RO-MAN2017*, Portugal, 2017.

[35] D. Voilmy, C. Suarez, A. Romero-Garcés, C. Reuther, J.C. Pulido, R. Marfil, L.J. Manso, K. Lan Hing Ting, A. Iglesias, J.C. González, J. García, A. García Olaya, R. Fuentetaja, F. Fernández, A. Dueñas, L.V. Calderita, P. Bustos, T. Barile, J.P. Bandera, and A. Bandera. CLARC: A cognitive robot for helping geriatric doctors in real scenarios. In *ROBOT (1)*, volume 693 of *Advances in Intelligent Systems and Computing*, pages 403–414. Springer, 2017.

[36] S. Wintermute. Imagery in cognitive architecture: Representation and control at multiple levels of abstraction. *Cognitive Systems Research*, 19:1–29, 2012.