The CORTEX Cognitive Robotics Architecture: use cases

P. Bustos[a], L.J. Manso[a], 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*

Abstract

CORTEX is a cognitive robotics architecture inspired by three key ideas: modularity, internal modelling and graph representations. CORTEX is a computational framework designed to support early forms of intelligence in real world, human interacting robots. The deployment of a CORTEX architecture starts by selecting an a priori functional decomposition of the capabilities of the robot. This set of abilities are 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 to pure deliberative ones which only operate within the internal model, but they can only communicate among them 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 humans in it, and the robot itself. All these entities are perceived and transformed into different levels of abstraction, ranging from the raw data provided by the sensors to high-level symbolic relations such as "the person is talking and staring at me" or "person A and person B are holding hands". In this paper we present recent advances in the CORTEX architecture and several real-world human-robot interaction scenarios in which it has been deployed and tested. We describe our interpretation of the ideas inspiring the architecture and the reasons why this specific computational framework that integrates all them is a promising architecture for the robotics intelligence that is needed in the social robots of tomorrow.

Keywords: Cognitive Robotics, Robot Control Architectures

1. Introduction

Cognitive robotics is concerned with endowing a robot with the capacity to plan solutions for complex goals and to enact those plans while being reactive to unexpected changes in its 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 fit. Social robots, in addition, are also required to interact with humans in order to be of help to them. The design space of these architectures lies somewhere among cognitive architectures that study human cognition, the sensory-motor processes that connect and anchors the robot to the world, the more recently appeared human-robot interaction (HRI) field, and the possibilities offered by current hardware and software technologies working under real-time, or close to it, conditions. Each of these dimensions spawns one or more disciplines in its own, so we won't be reviewing them here. Instead, we will select what we think is a promising region of that space that is currently being explored by several research groups and it is also where our architecture CORTEX is placed. We delimit this region with a short list of constraints:

- The global behaviour of the robot is divided into a set of functional blocks or agents, each one specialized in aspects such as navigation, manipulation, grasping, localization, object detection and recognition, space knowledge acquisition, dialoguing, planning or reasoning. This division provides a broad encapsulation level to reduce the overall complexity of the system but, at the same time, this division immediately creates the need to transfer information among them.

- At the most abstract level, the robot operates within the limits of a symbolic domain theory which is defined in a formal language, similar to PDDL [1]. Objects are the elements that make up the represented world and rules are possible actions that, when applied, change the world in predictable ways. One of this agents will implement a procedure to search for sequences of these rules creating plans to achieve human requests.

- Agents build and maintain a working representation of the environment and of the robot. These representation is used to perform internal computations, metric and symbolic, so agents can perform their tasks and human requests are attended.

- Agents will try to follow a global plan by executing specialized behaviours. In doing so, they might use themselves a domain specific planner and they also might introduce sub-goals needed to proceed with their actions. Those subgoals will be planned at a global level -planning agent- and the result propagated so the others become aware of it. This dual dynamics provide a general support for an interleaved deliberative-reactive execution of plans.

- The hardware and software implementation must exploit distribution at most design levels, so the robot can be autonomous and achieve real-time performance in human-robot interactions.

CORTEX is a cognitive architecture that fits in this region of design space. In addition to these criteria, it is inspired by three key ideas: modularity [2], internal modelling [3] and dynamic graph representations. Fodor's modularity theory [2] proposes the existence of a set of encapsulated input/output modules pervaded by a central system where beliefs and plans are computed. Modules are domain specific and perform local computations, while the central system has potential access to the relevant representations construed by the modules through proper interfaces. Holland and Goodman [3] have vigorously defended the need for internal (direct) models as a crucial element in intelligent, conscious machines. These internal models have to capture the current and past state of the robot itself and of its environment and the ways that state changes when something perturbs it. They have to serve as efficient predictive mechanisms that can provide alternate course of actions but, at the same time, they cannot be too demanding on the precision or updating requirements. As Holland and Goodman [3] suggests, it would be when an internal model of the robot is put in relation to the world, when a sort of self-awareness would emerge. Finally, graphs are mathematical objects with extensive development as computational structures that can be used to hold, using the same support, representations of different aspects of the world. In CORTEX we are interested on representing objects with symbolic and metric properties using the same structure.

In this paper we present recent advances in the CORTEX architecture and several real-world human-robot interaction scenarios in which it has been deployed and tested. CORTEX is conceived as a computational framework designed to support early forms of intelligence in real world, human interacting robots. The deployment of a CORTEX architecture starts by selecting an

a priori functional decomposition of the capabilities of the robot. They are then translated to computational modules or agents that can be anywhere in the reactive-deliberative spectrum but with the restriction that they can only communicate among them through a graph structure, that we call 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 transformed into different levels of abstraction, ranging from the raw data provided by the sensors to high-level symbolic relations.

2. Related work

Over the years, several cognitive architectures have been proposed. In the following the most used are briefly described. ACT-R (Adaptive Components of Thought- Rational)[4, 5] is concerned primarily with modelling human behaviour, and has been developed since the late 1970s. It is a hybrid cognitive architecture and theoretical framework for emulating and understanding human cognition. The architecture is organized into a set of modules processing different type of information. SOAR (State, Operator And Result) [6] is a cognitive architecture designed to model general intelligence [6]. It stores its knowledge in form of production rules, arranged in terms of operators that act in the problem space which describe simple, primitive actions that modify the agent's internal state or generate primitive external actions, whereas others describe more abstract activities. EPIC (Executive Process Interactive Control) is a cognitive architecture that aims at capturing human perceptual, cognitive and motor activities through several interconnected processors working in parallel, while trying to make models of human-computer interaction. CLARION (The Connectionist Learning Adaptive Rule Induction ON-line) is a hybrid architecture that incorporates a distinction between symbolic and sub-symbolic processes and captures the interactions between the two [7]. The architecture contains four memory modules: action-centred system, non-action centred subsystem, motivational subsystem and metacognitive subsystem. Also, there have been some attempts at using these cognitive architectures in robotic systems. For example, ADAPT [8], which uses SOAR, is an attempt at developing a robot with a full range of cognitive abilities. CiceRobot [9] makes use of expectations in the perception loop as a method to understand and react to the environment. An embodied version of ACT-R, namely ACT-R/E [10], has been used on robots to perform

coordinated activities with humans. Also, a recent version of SOAR[6] has been used to learn representations of objects in the world. SAL (Synthesis of ACT-R and Leabra) [11], is an attempt to connect ACT-R with a neural implementation of the visual system, which has been used to recognize and clasify objects. Recently, Mala [12] a cognitive architecture intended to bridge the gap between a robots sensori-motor and cognitive components. Mala is a multi-entity architecture which comprises several essential capabilities and characteristics of architectures needed for robots to develop high levels of autonomy such as modular and asynchronous processing, specialised representations, relational reasoning, mechanisms to translate between representations, and multiple types of integrated learning.

All these architectures have a common denominator, the use of finite-state based behavioural mechanisms for specific robotic (even very complex ones) tasks, that have to be accordingly and manually adapted to new upcoming tasks. As an attempt to avoid this adaptation, in this paper we describe a robotics cognitive architecture for social robots named CORTEX. This architecture integrates different levels of abstraction (from basic geometry to high-level predicates) into a unique Deep State Representation (DSR) that different agents interface.

3. The CORTEX Architecture

CORTEX is organized as a collection of agents that cooperate to achieve a goal. Each agent is in charge of a basic robotic functionality affecting a specific domain. Some agents are input systems that transform sensor information into internal objects. Other agents generate actions activating the robot effectors. Still other agents compute plans to satisfy the human requests. All of them interact through a graph structure named Deep State Representation (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 internal dynamics and partly to the current plan on execution.

The execution of a task in CORTEX, in its current state, can be understood as a perturbation to an otherwise resting situation. A very rough description of this dynamics would start when a new request enters the system as, for example, a human utterance, the Dialog agent processes the raw data and builds an internal object that is injected in the DSR as the new robot's desire. The new state of the DSR is propagated back to all agents

but is the Executive, a key agent in charge of the execution monitoring, who picks the change and processes it as a new request. This request is interpreted as, usually, the need to change the world from its current state to a new one. To do that 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 in the DSR graph as special edges, and is propagated back to the agents. Each agent picks from the updated graph the information that it can interpret as a local goal, and starts its activity. When the final state is reached the task is accomplished and the whole systems goes back to its restful state. In pursuing their goals, agents, perceptive and action, can inject their own sub-goals in the DSR for the Executive to compute a plan for them. This interleaving dynamics allows for a reactive-deliberative behaviour, in which top-down plans and bottom-up reactions to unforeseen situations can cooperate. In Figure 1 an overall organization of CORTEX is depicted.

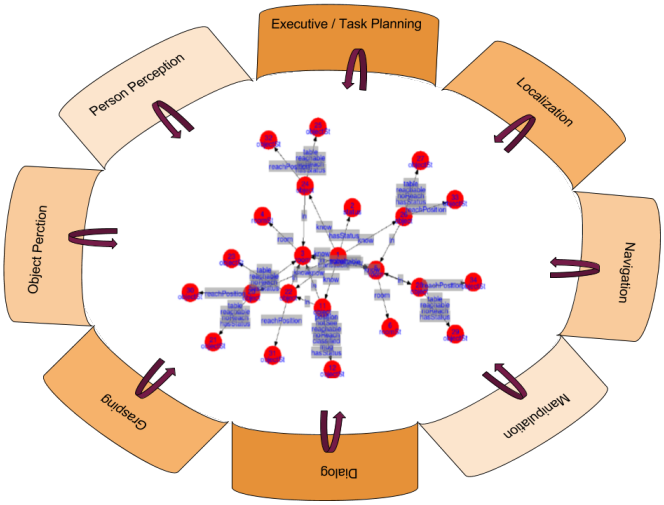


Figure 1: Overall organization of CORTEX. The architecture can be depicted as the confluence of a set of agents that communicate through a shared representation of the robot's internal state and of the environment. Each agent can contribute by adding structure or by changing it. Human transmitted missions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

3.1. Deep State Representation

The Deep State Representation (DSR) is a multi-labeled 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 as predefined object types linked by 4×4 homogeneous matrices. Again, they are respectively stored as nodes and edges of the graph. Figure 2 shows one 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. But, at the same time that 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**.

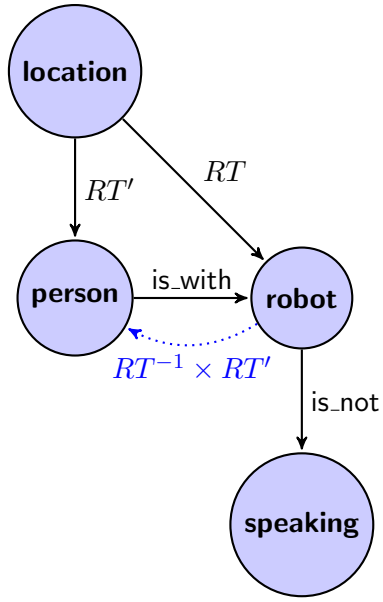


Figure 2: Unified representation as a multi-labeled directed graph. Edges labeled as is_with and is_not denote logic predicates between nodes and they belong to Γ_s. Edges starting at **room** and ending at **person** and **robot** are geometric and they encodes a rigid transformation (RT' and RT respectively)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

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 them. Metric concepts describe numeric quantities of

objects in the world that can be structures like a three-dimensional mesh, scalars like the mass of a link, or lists like revision dates. Edges represent relationships among 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 them. Then, the DSR can be described as the union of two *quivers*: the one associated to the symbolic part of the representation, $\Gamma_s = (V, E_s, s_s, r_s)$, and the one 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 $s, r : E \to V$. 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 $e = \mathbf{uv} : \mathbf{u} \to \mathbf{v}$ 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:

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

On the other hand, 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, the quiver Γ_g defines a directed rooted tree or rooted tree quiver [13]. The *kinematic chain* $C(\mathbf{u}, \mathbf{v})$ is defined as the path between the 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 **u** and **v** to their closest common ancestor **w**. Note that the values from **u** to the common ancestor **w** will be obtained multiplying the inverse transformations. One example of computing a kinematic chain is shown in Figure 2.

One of the main benefits in 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 collect many years of continuous effort in many of the different AI fields needed to build a robot. The most relevant agents are described bellow:

3.2. 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* [14], and a continuous map holding the geometric objects that exist in the environment. This object map is contained in the common DSR representation and updated every time an agent changes it. Objects may belong to three different categories: common objects, robots, and people. Currently, this map is manually created by the user. Both maps are put in correspondence using a graphic tool created for this purpose.

3.3. Navigation

Navigation provides the capability of global and local path planning and 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 will compute a clear path to the target and drive the robot through it handling unforeseen obstacles. To do this, the agent initiates three internal processes (components) that run concurrently. The first one is a Probabilistic Road Map (PRM) planner[15] 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 3. In case that the graph still had more that one connected region or there was not a direct line of sight from the robot (or the target) to the graph, the component uses a RRT planner [16]. Once a path is found it is shared with the other two components. The second one is connected to a laser sensor and takes the path an projects it inside the laser field, to check that it remains clear from the world as perceived in real-time by the range sensor. It applies

a variation of the *elastic band* algorithm [17] as explained in [18]. The current path (elastic band) is deformed according to a set of dynamic equations that code a virtual force exerted by the objects.

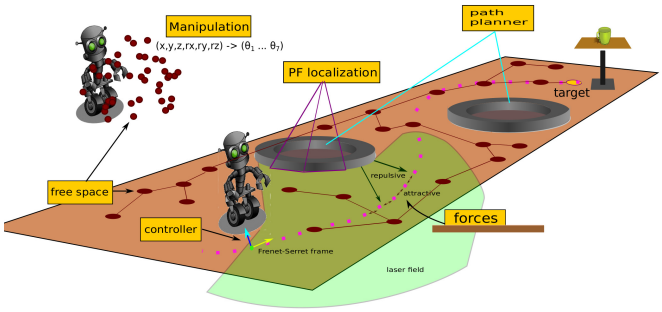


Figure 3: Graphical representation of the Navigation agent showing the PRM, the selected path and the projection of the path on the laser field. See text for details.

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 if 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, the angle that the robot forms with the tangent to the path, curvature or current speed [18]. 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, that is dynamically adapted to upcoming circumstances in the world.

3.4. 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 to successfully transcribe input speeches in different languages. In order to

internalize the information retrieved from the transcriptions, we may parse input phrases to extract elements like entities, locations, or relevant terms, using toolkits like SENNA [19]. The versatility of CORTEX allows to implement the comprehension stages 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 been also used to obtain a semantic representation of the input phrase, usually known as CFR (Command Frame Representation).

3.5. Manipulation

Manipulation is one of the most difficult skills in today social robots. For a mobile manipulator, i.e. a mobile base with arms, the best approach seems to be the classical use of a whole body inverse kinematics and dynamics approach [20]. Although some work is being done in this direction, 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 is a source of problems. However, in the experimental situations that we describe below, and in many others of similar complexity, the current separation in two agents is a good enough approach. Manipulation in CORTEX follows a similar schema that 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.

Its 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 4. Each cell in the grid holds a set of preferred orientations for the hand at that location, providing the inverse 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 RGBD camera of the robot 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 not navigable. The agent now searches the grid for a free path and shares it with a second modules that drives the arm through the *way points* using a

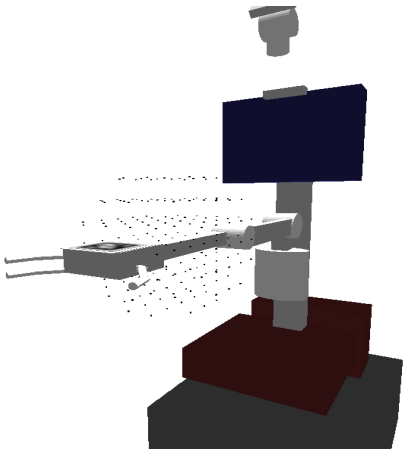


Figure 4: Graphical representation of the robot with the inverse kinematics grid. Each point in the grid represents a cartesian pose, translation and orientation, where the angle values that correspond to that pose are stored. See text for details.

generalized 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 [18]. When a final grabbing position is reached, the fingers are closed around the object and the transfer concludes.

3.6. Person detection

Person detection 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 the human skeletons are not constrained enough and may produce some erroneous interpretations, as Figure 5 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 [21].

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.

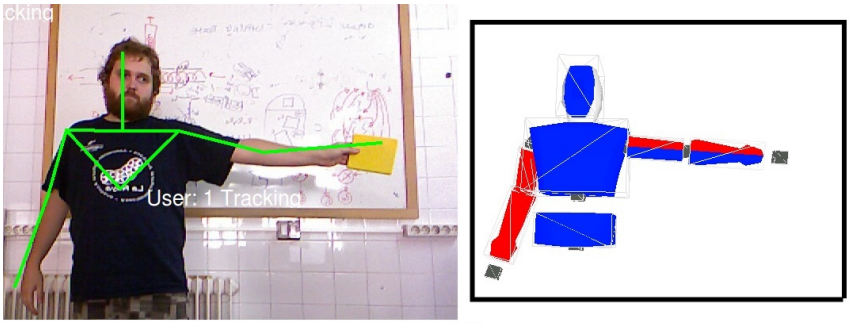


Figure 5: Person detection using a model-based approach. The skeleton obtained by the OpenNI library fails to estimate the real length of the arm, that is corrected by the fitted human model that is forced to respect kinematic constraints.

Once a human is in the DSR, it is visible by all the agents. 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 them being a human or an inanimate thing. In the first case it must obey some additional rules that involve for example, to stop and ask for any commands, or to perform 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.

3.7. Executive

Robots are often expected to be able to achieve more than a handful of different missions. Generating and supervising the corresponding plans using automated planning techniques is generally faster 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, making sure that the rest of the agents have access to the current plan. The current implementation uses the planning domain definition language, the planner, and a modification of the world models proposed in [22]. The planning domain definition language is used to describe the transformations that can be made to the world model (see figure 6).

Some of these transformations are associated to physical or perceptual

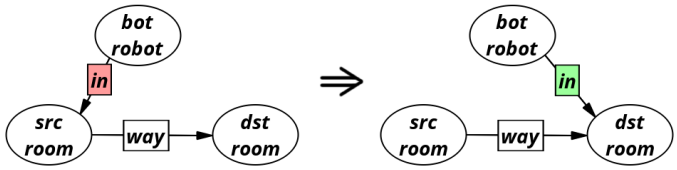


Figure 6: transformations have left and right-hand sides. The left-hand side describes the preconditions for the transformation to be executed, whereas the right hand side describes how the pattern in the left-hand side would change. The rule at hand describes how the world model is modified when the robot moves from a 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..

actions, but there can also be *deliberative actions* in which no 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 direct physical consequence). Such 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 if the current plan (or a version with the first actions removed) is still valid.

4. Use cases with CORTEX

The success of a robotic architecture relies on its capability for providing a robot with capabilities of 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) 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 the physical platforms they have. Concretely, UMA and UNEX have traditionally designed and assembled their own robots [23, 24], while UCLM has historically worked with standard platforms [25].

4.1. The welcome home situation

The objective of this use case [26] 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 new visitor is identified by the Velodyne sensor, the robot welcomes him/her by previously moving to their position. Then, the robots offers to accompany the visitor to some of the staff employees or a location in the environment. After a short dialogue, the robot performs the desired action and is ready for welcoming a new visitor. Figure 7.a shows the robot while performing this use case.

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 development of several agents like people detection, user recognition, localization, mapping, navigation or speech recognition and generation.

4.2. The advertisement robot scenario

This experiment involves an advertisement robot, Gualzru, built to approach people, and then try to convince them of going back to a panel where some new product is exposed for marketing purposes [27]. The conviction stage involves recognizing 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 for going to the panel.

Gualzru robot (Figure 8.a) was specifically designed and built for advertisement purposes, and it consists of different units distributed along its structure. The challenging of some requested tasks forced the use of third-party libraries or SDKs. For instance, the speech recognition relies on the Microsoft Speech SDK, which allows to use non generalist grammars trained from corpus. These corpus 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 they help reduce the workload of the main computer thanks to a distributed processing scheme. The adoption of CORTEX as architecture made it easier

the overall integration of all the different modules, as well as the problem definition thanks to the use of the DSR.

4.3. The CLARC scenario

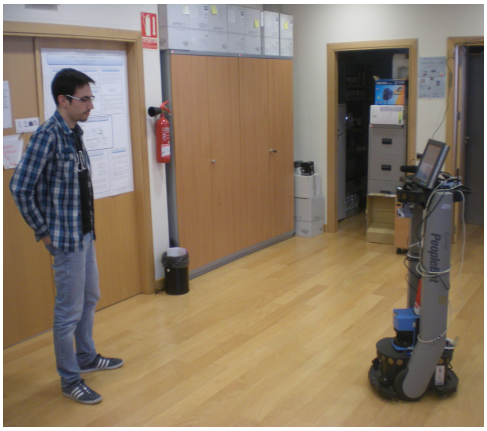
Comprehensive Geriatric Assessment (CGA) procedures evaluate the degree of autonomy of elderly people. CGA procedures imply subjective interviews, but they also include physical and cognitive tests, that could be autonomously performed by a robot. The CLARC proposal [1] was selected in a recently started ECHORD++ challenge to perform these automated tests. The proposal focuses on the development of CLARC, a mobile robot able to receive the patient and his family, accompany them to the medical consulting room and, once there, help the physician to capture and manage their data during CGA procedures. The hardware of the robot is provided by MetraLabs GmbH (Figure 8.b depicts a first prototype already performing tests with elderly people), and it is currently being designed following a human-centered approach. The software architecture of the platform is an implementation of CORTEX. The whole CGA session is encoded using Automated Planning, being able to autonomously plan, drive, monitor and evaluate the session. The sequence of tests and their parameters will be adjusted on-the-fly to each user. Carefully designed robot-clinician interfaces allow the expert to check sessions, analyse results or modify data using a web-based application. This will significantly reduce total times for CGA sessions increasing the quality and quantity of the data collected while maintaining safety and personalised care.

4.4. 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 attends requests from humans, who ask the robot to bring them objects located on the tables of the apartment (Figure 7.b).

The execution of this use case requires coordination among 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

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



(a) Welcoming

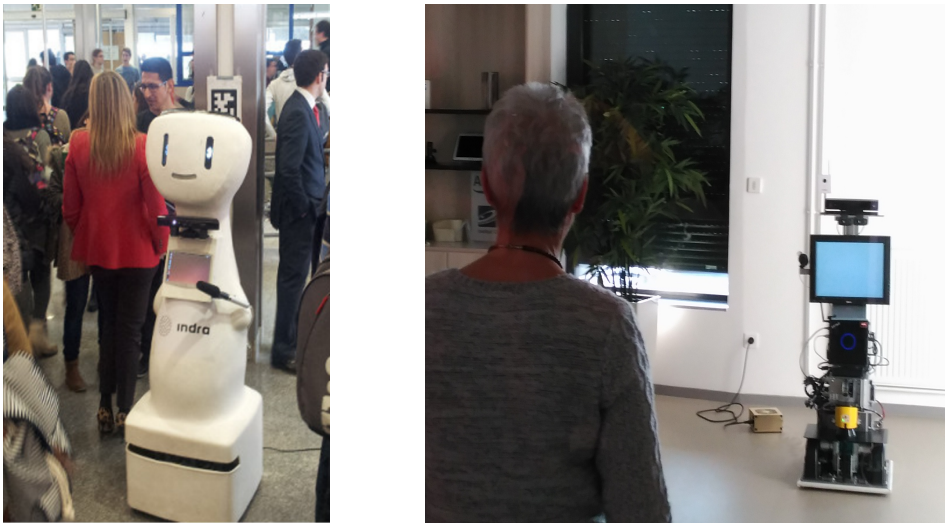
(b) Bringing a mug

Figure 7: Robots working in the welcoming and BringMe(x) scenarios.

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 recent public demonstration, on September 2016, Shelly was able to successfully deliver the requested object to different persons 40 times in a row, proving the maturity of the underlying hardware and software.

5. Discussion

In the previous section, four use cases involving three different robots and scenarios have been described. The three of them used the CORTEX architecture. The success in all four experiences was possible, with a limited amount of human and material resources, thanks to the reusability provided by the CORTEX architecture. Most of the code was reused and the efforts were 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 are controlled by state machines. The reasons to avoid cognitive architectures might be related to the complexity and effort of developing a new architecture or even of using and integrating an existing one. Although new architectures tend to be construed on top of well proven and documented frameworks, i.e. RoboComp, ROS, OROCOS, etc. the intrinsic complexity, code size



(a) Advertising (b) Performing a CGA test

Figure 8: Robots working in the advertisement and CGA scenarios.

and maintenance, diversity of topics, theories and algorithms that it encompasses, hinders the portability and integration by other research groups, let it alone the quantitative comparison among them. 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.

There are two reasons that we will consider 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, that allows a hybrid, metric and symbolic, representation of the world model held by the robot
- the Executive agent, that can be configured to use any planning domain definition file, is completely domain-independent

and also those 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, with the exception of some robot-specific actions (*e.g.*, only Shelly is able to grasp and deliver objects)

- the agents that 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 in 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 [28]. Using component-oriented technologies usually entails the additional effort of maintaining the middleware-related code. 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 [29].

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 that state-machines and a practical one if there exits a cognitive architecture providing the underlying mechanisms for it to work.

6. Conclusions

In this article, we have presented the main functionalities of CORTEX by a) describing this architecture and all the different components it consists of, and b) presenting a list of scenarios where it has been exploited to develop use cases involving real social robots.

One of the main advantages of CORTEX is the management of the symbolic/metric bridge, which is carried out by means of the DSR (Deep State Representation). In order to evaluate the appropriateness of this representation, CORTEX architecture must be adopted in heterogeneous robotic

projects. This evaluation has been carried out and depicted in the article, where we have exposed how CORTEX has been selected by different research groups from different Universities. We consider a very important contribution the generation of an architecture suitable for being adopted by different Universities.

Acknowledgments

This work has been partially funded by FEDER funds and the Spanish Government (MICINN) through projects TIN2015-65686-C5; and by the Red de Excelencia "Red de Agentes Físicos" TIN2015-71693-REDT and Ayudas de Consolidación de Grupos de la Junta de Extremadura, 2016. This work has been 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 this research.

7. References

[1] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL - The Planning Domain Definition Language, Technical Report TR-98-003, Yale Center for Computational Vision and Control,, 1998.

[2] J. A. Fodor, The modularity of mind : an essay on faculty psychology, A Bradford book, MIT Press, Cambridge (Mass.), London, 1983.

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

[4] J. R. Anderson, C. Lebiere, The Newell Test for a theory of cognition, Behavioral and Brain Sciences 26 (2003).

[5] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, Y. Qin, An integrated theory of the mind, PSYCHOLOGICAL REVIEW 111 (2004) 1036–1060.

[6] J. E. Laird, The Soar Cognitive Architecture, The MIT Press, 2012.

[7] R. Sun, The importance of cognitive architectures: an analysis based on clarion, Journal of Experimental & Theoretical Artificial Intelligence 19 (2007) 159–193.

[8] D. P. Benjamin, D. Lyons, D. Lonsdale, Adapt: A cognitive architecture for robotics, in: Proceedings of the International Conference of Cognitive Modeling.

[9] A. Chella, I. Macaluso, The perception loop in cicerobot, a museum guide robot, Neurocomput. 72 (2009) 760–766.

[10] J. G. Trafton, B. R. Fransen, An embodied model of infant gaze-following, Manchester, United Kingdom.

[11] Y. Vinokurov, C. Lebiere, D. Wyatte, S. Herd, R. O'Reilly, Unsupervised learning in hybrid cognitive architectures.

[12] A. Haber, C. Sammut, A cognitive architecture for autonomous robots, Advances in Cognitive Systems 2 (2013) 257–275.

[13] V. Katter, N. Mahrt, Reduced representations or rooted trees, Jornal of Algrebra 413 (2014) 41–49.

[14] G. Grisetti, C. Stachniss, W. Burgard, C. Science, L. Sapienza, V. Salaria, Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters, Informatica (2007) 1–12.

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

[16] S. M. LaValle, Planning Algorithms, volume 2006, Cambridge University Press, 2006.

[17] S. Quinlan, O. Khatib, Elastic bands: Connecting path planning and control, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 802–807.

[18] M. Haut, L. Manso, D. Gallego, M. Paoletti, P. Bustos, A. Bandera, A. Romero-Garcés, A Navigation Agent for Mobile Manipulators, Springer International Publishing, Cham, pp. 745–756.

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

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

[21] L. V. Calderita, J. P. Bandera, P. Bustos, A. Skiadopoulos, Model-based reinforcement of Kinect depth data for human motion capture applications., Sensors (Basel, Switzerland) 13 (2013) 8835–55.

[22] L. Manso, P. Bustos, P. Bachiller, P. Nez, A perception-aware architecture for autonomous robots, International Journal of Advanced Robotic Systems 12 (2015) 13.

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

[24] J. P. Bandera Rubio, A. Bandera, S. López-Chamizo, A. Cumpián, P. J. Sánchez, Roboarch: An autonomous robot for analysis and documentation of historical architectures (2013).

[25] J. Martínez-Gómez, J. A. Gámez, I. García-Varea, V. Matellán, Using genetic algorithms for real-time object detection, in: Robot Soccer World Cup, Springer, pp. 215–227.

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

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

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

[29] A. Romero, L. Manso, M. Gutirrez, R. Cintas, 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), pp. 1–9.