Navigation among People with CORTEX

Pablo Bustos, Luis J. Manso, Pilar Bachiller, and Pedro Núñez

RoboLab, Escuela Politécnica, Universidad de Extremadura
Cáceres, Spain
http://robolab.unex.es

Abstract. Robot navigation is one of fist functionalities that attracted the attention of roboticists and nowadays there problem is considered a mature discipline in many aspects. However, when considered as part of a cognitive architecture and when the usual boundary conditions are removed, navigation becomes again a hard problem that needs to be approached as a gear forming part of a bigger system. In this paper we discuss a navigation sub-architecture designed to be a part of a the robotics cognitive architecture CORTEX, and how it can be evolved from classical navigation tasks to the more challenging social navigation scenarios required in social robots.

Keywords: Robotics, Social Robots, Cognitive Architectures, Social Navigation

1 Introduction

Cognitive architectures have been studied as a part of research in Artificial Intelligence since the 1950s with the ultimate goal of achieving human level intelligence. Cognitive architectures for robotics additionally address the existence of a physical robot that has to evolve in the real world. However and as stated in the extensive review of Kotseruba et al [1], there is not a clear definition of general intelligence, neither a comprehensive set of competencies and behaviours required for intelligence that can be used to compare the many architectures currently under development. To approach the problem of cognitive robotics, some authors have extended existing cognitives architectures created to study human intelligence with specific perception-action modules. ADAPT adds declarative memory and a schema language to Soar, ACT-R/E ads manipulative, spatial reasoning and navigation modules to ACT-R and SS-RICS adds subsymbolic modules to ACT-R providing perception and action algorithms [2]. Architectures specifically designed for robotics give much more weight to perceptive and action components, seeking creative ways to combine task level planning with the effective execution of complex behaviours in real-world environments [3] [4] [5] [6].

In this paper we are interested on architectures designed to create flexible behaviours in autonomous robots, specifically in robots that interact with humans. These architectures have to encompass several critical features such as real-time

operation, interleaving of reactive and deliberative operations, complex perceptive functionalities to understand the dynamic world where the robot operates and large software codebases difficult to manage. To face these challenges, a modular organization is usually adopted making extensive use of distributed computing technology in the form of robotics software development frameworks. These software elements become the building bricks of modern robotics cognitive architectures. Although the goal of these architectures is to be of general application, when applied to real robots in real scenarios some choices have to be made. Usually, the type of scenarios, the physical configuration of the robot and the type of missions to be tackled constrain the set of competences that will be part of the architecture. Sets of generic competences proposed by several authors include perception, learning, reasoning, decision making, planning and acting [7], although it is infrequent to see all of them working together. In practice, more specific functionalities are defined and implemented to support these competences, such as object and human perception, manipulation and grasping, navigation, localization and mapping, under additional constraints imposed by current software and hardware technology.

We are interested on how these architectures can be extended to make existing functionalities operate in more complex situations. This feature might a be a good indicator of the *potential* of an architecture to succeed in achieving robot missions, since most of them are initially designed with a set of necessary constraints that are intended to be removed with time. In this specific case we focus on the *navigation* competence, mandatory in most RCAs. We analyse the problem of extending classical navigation among inanimate obstacles to socially-assitive navigation, where humans enter the scene and new social rules have to be applied. We proceed with a brief introduction to CORTEX and how its navigation agent is modified to account for social navigation requirements.

2 CORTEX

CORTEX is a robotics cognitive architecture organized as a set of cooperating agents that communicate through a shared representation[1]. Agents have been created to provide intelligent behaviour in social robotics scenarios, such advertisement [9], psychiatric evaluation [10] or autonomy enhancement [6] and, thus, to cover competences such as navigation, localization, manipulation, dialoguing, object and human perception, task planning and some sorts of learning. Being a cognitive architecture, CORTEX has a planning and executing agent with access to the domain knowledge of these scenarios. This knowledge is compiled as a collection of typed objects and logical predicates related by transformation -*if-then*- rules [2]. The shared representation used by agents to communicate is a

[1] The concept of *agent* is taken here as a generic software module that implements some functionality of the architecture. These agents can be rational as in [8], reactive or both, but they keep a predefined structure specified by the architecture.

[2] Although CORTEX provides a graphical editor that facilitates the management of this knowledge, it can be automatically translated to PDDL [11]

working memory that holds an updated state of the world around the robot and of itself. It is not size limited as in biological inspired models and is structured as a graph named Deep State Representation (DSR). The DSR itself is maintained by an special agent, DSRa, that provides very efficient concurrent access and publish/subscribe services to the other agents using the robotics *framework* RoboComp[3] [12]. The most salient features that define DSR are:

- Nodes represent objects and edges are predicates relating nodes.
- Objects can have a list of attributes.
- Nodes have types that are defined in a hierarchy of *is-a* relationships. This types capture semantic knowledge about things in the world.
- Predicates have also predefined types that refer to arbitrary relations among objects such as *above*, *inside*, *part-of* or *touching*.
- Node's attributes can be symbolic or numerical. Numerical attributes can represent kinematic relations between nodes allowing for a complete kinematic tree embedding inside the graph. Also, meshes with the geometry of robot's parts or worlds objects can be stored, similarly to the scene-graph of 3D graphic engines. This feature makes DSR a hybrid representation that can be used by the agents to reason about it, solve problems, update it with changes in the world or predict the outcome of actions.

Agents communicate through this graph representation although they can keep a local copy of all or part of it. Changes made by the agents to the DSR can be structural, when nodes or edges are inserted or deleted, or non-structural, when only the attributes of a node are changed. In both cases, all agents receive a notification of the change by a subscription mechanism. When an agent wants to change a node's attribute or add or delete a group of nodes, it sends the proposed change to the DSR agent. This agent, prior to modifying the graph, might check that the final state graph is reachable by application of the domain rules[4].

See [13][14][15][6][10] for a more detailed description of CORTEX and its application to various robots in different scenarios.

With this architecture we want to combine in an efficient way two confronted concepts that come up inevitably in the design of any intelligent system:

- Distribution, is an architectural feature needed to cope with the complexity of the software that makes up the robot's control system and artificial intelligence. Current social robots executing human commanded missions in research facilities might deploy between 20 and 60 coarse grain software components distributed among a local cluster of connected computers. Each

[3] RoboComp is open source and maintained by a community of developers at http://robocomp.org

[4] This function is specially interesting if agents have the freedom to insert new subgraphs in the DSR. It can be the case that the resulting graph cannot be reached by application of the transformation rules in the domain knowledge. If that situation is accepted, the planner might not be able to solve certain missions.

component itself is usually designed as a set of partially decoupled concurrent threads. Engineering needs decoupling to build large complex systems and, in this context, decoupling is implemented as large scale software distributed systems. Each functionality that is isolated from original problem creates a potential need for communication with the remaining parts.

– Context sharing, is a situation that appears in distributed systems composed of several decoupled components that, at some point, require data from other components to make better informed decisions. Cognitive systems rely on declarative knowledge, i.e. *there is a person nearby*, that is created, updated and used in many parts of the system. In general terms, the finer the grain in a distributed architecture, the more difficult is to access the information created by other participants.

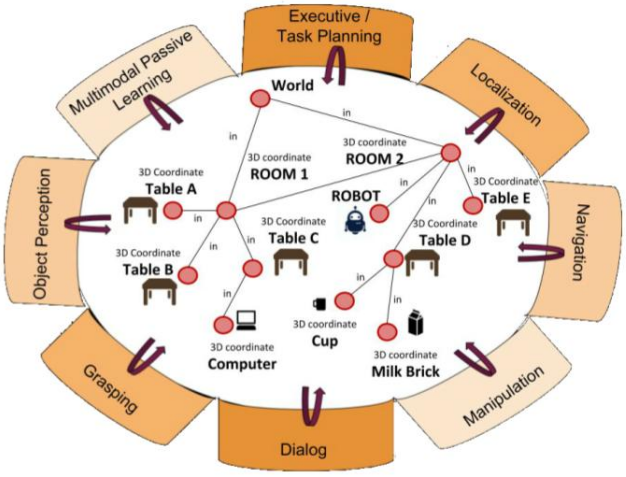


Fig. 1: Scheme of the CORTEX architecture showing a set of agents interacting with a central, shared representation. Nodes represent object in the world (and the robot) and edges represent predicates or relationships among nodes. The result is a hybrid symbolic/numerical representation that can hold data at different levels of abstraction, from sensor reading to high-level symbols.

The approach followed in CORTEX is an attempt to facilitate the creation of large distributed software for robots that concurrently perceives, plans and acts

to take a representational state to a target configuration as defined by a given command. CORTEX global functioning is driven by two forces, a) the creation and maintenance of a coherent representation of the environment and of the robot itself, and b) the transformation of the current represented state into a goal state.

3 Navigation as global-local interactions

The Navigation agent in CORTEX has been in steady evolution for the last few years. It was originally designed to tackle the problem of how to accommodate the local interactions of the robot's path-following controller, with the global inputs provided by the path planner. During typical displacements it is common that a stuck local controller calls for a replanning action, so the path changes and has to be adapted using real time range measurements. This continuous interaction between a fast, real-time, reactive, bottom-up behaviour like the local controller, and a slower, deliberative, top-down reasoning process is paradigmatic in robotics, and occurs in very different domains, time scales and abstraction levels.

From an architectural point of view, these local-global interactions have been approached commonly as function calls among software components running specialized algorithms. In the approach presented here, interactions are handled through a shared structure that represents the current path followed by the robot. This idea was initially inspired by the elastic bands concept presented in [16], subsequently extended in [17][18] and applied in real world scenarios [19][19][20], where the path was represented as an elastic, deformable string exposed to physical forces computed from the range measures of the robot's sensors.

We have extended this idea in several ways and built a navigation sub-system whose structure is self-similar to the overall CORTEX's organization. An initial version was presented in [21]. In our agent, several components interact over a shared representation of the path, creating a particular dynamics that leads the robot to its goal. Components read and modify the path in several ways, reacting to changes introduced by others. This organization has, in our opinion, several interesting advantages:

- The loose coupling among components facilitates the addition of new interactions that modify the robot's behaviour.
- The explicit existence of the *virtual* path simplifies the interpretation and debugging of the whole system.
- The overall behaviour is better understood as a dynamical system driven by several identifiable forces.

Initially, or when there is no goal assigned to the agent, the virtual path does not exist and, consequently, the components are in an idle state. When a target arrives to the agent, the path planner creates a new feasible path using, whether the local context, i.e. exits a clear straight path to the goal, or the

geometric model of the environment that is part of the DSR. Once an initial path is created, the other two components activate. *Projector* will transform the real-time laser measurements into a force field that brings the initial mental construction to the reality. In parallel, *Path Follower* will drive the robot along the path, deleting the part travelled as it advances. Eventually, the path will be completely removed, the robot will be correctly positioned at its destination and all three components will return to a rest.

The three components that access and modify the path are described in the next subsections.

3.1 The path

The path is constructed as a software object with a well defined interface. The core data structure of the path is a list of inflatable 2D points or bubbles that behave as an elastic band under various internal and external forces and processes. The path is defined as the ordered set, $P = p_i : p \in \mathbb{R}^2 x \mathbb{N}, i \in 0..N$, with two real coordinates in a global refererence system maintained by the robot, and an integer one for the bubble's radius. Each radius is computed as the minimum distance to the surrounding objects in the environment, using some robot's measurement device. The function $\rho(p)$ that computes this radius is defined as $\mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \{R^+ \cup 0\}$ and is implemented as a search iterating over the laser array and the list of visible points.

The set of forces and processes that affect the path are:

- The *Path Planner* creates a new path given the robot's position and a target, based on the internal representation of space, DSR, that is maintained by the collection of agents. The creation of the path initiates the internal dynamics that will eventually take the robot to the target.
- The first internal force, f_s models the tension in a physical elastic band. It preserves the virtual band from excessive bending, stretching it out if left alone. The force can be computed from nearby points as,

$$f_s = k_s\left(\frac{p_{i-1} - p_i}{\|p_{i+1} - p_i\|} - \frac{p_{i+1} - p_i}{\|p_{i+1} - p_i\|}\right) \tag{1}$$

- The main external force is created by interaction of the path with the *Projection* component and is described with more detail in Subsection 3.3. This force pushes the band away from nearby obstacles and interact with the previous one creating a dynamic response of the band to the environment, that tends to maintain a smooth, safe path among the obstacles in the world.
- There are two *cleaning* processes that maintain a maximum and a minimum distance between the points in the band. To do that they create or remove exiting points when two simple distance thresholds are exceeded. See Figure 2 for an example.
- There is also a process that removes the points of the path that have been already traversed by the robot. This action causes the path to eventually disappear signalling the successful conclusion of the navigation task.

3.2 Path Planner

The Path Planner activates either because a new command has arrived from the task planner, or because the robot is stalled in the path. In both situations, the Planner will search for a safe path in the DSR representation of the space. The planning functionality is provided by a combination of two classical path planners, PRM [22] and RRT [23]. A PRM is used initially to create and maintain a local, fast access, graph of the free space from the existing model of the environment as represented in the DSR. Most paths are computed on this graph using standard minimum cost algorithms, but when the target is not reachable from the graph because there is not a direct line of sight, an RRT-Connect planner [24] used to search for a safe path. This new path is afterwards added to graph so the next time a similar situation appears, the target will be reachable. See Algorithm 1. This component improves its performance with time since isolated components that might remain in the graph after creation are progressively eliminated through additional, low-priority sampling, and the addition of paths covered by successful missions. With time, path planning is replaced by what could be seen as a very efficient memory recall of past experiences. Notice that if the robot is translated to a new room and given a rough map, grid or objectified, of the occupied space, it will quickly build its own graph of free space and improve it over time.

```
Result: path
while True do
    if new-target then
        r = robot-position() ;
        t = target-position() ;
        pi = r ;
        pr = closest-point-in-free-space-graph(r) ;
        if not atSight( pr ) then
            pi += RRT-plan(r, pr) ;
            update-PRM-with-new-samples(pi) ;
        end
        pt = closest-point-in-free-space-graph(t) ;
        if not atSight( pt ) then
            pf += RRT-plan(pt, t) ;
            update-PRM-with-new-samples(pf) ;
        end
        path = search-minimum-path-in-graph(pr, pt) ;
        path = pi + path + pf ;
        return path ;
    end
end
```

Algorithm 1: High-level view of the hybrid path planning algorithm that integrates PRM and RRT.

3.3 Projector

Once a path is made available by the planner, the Projector component adapts the trajectory to the external world by providing an external force that interacts with the path and is computed using the robot's range sensors. This process corrects errors in the path that are originated by an imprecise model, a miss-localization of the robot in the model or the appearing of unforeseen obstacles like people. The Projector agent computes a force field in which obstacles repel the path proportionally to the distance that separates them. This field relocates the path at a safe distance from the obstacles increasing the robot's clearance, and provides the reactive component necessary for real time control. For each point in the path p_i, the direction of maximum variation of the bubble's radius with respect to (x, y) variation in the bubble's position is computed with a discrete Jacobian:

$$\frac{\partial \rho}{\partial p} = \frac{1}{2\delta} \begin{bmatrix} \rho(p - \delta x) - \rho(p + \delta x) \\ \rho(p - \delta y) - \rho(p + \delta y) \end{bmatrix} \tag{2}$$

where ρ is the minimum distance function defined above, and p is the point in the path and x and y are the point's coordinates. $\delta x, y$ are discrete displacements in the point's position. The Jacobian is multiplied by the difference between a maximum distance threshold and the current circle radius, and by a scaling factor to obtain the repulsion force.

$$f_r = \begin{Bmatrix} k_r(\rho_0 - \rho(p)) \frac{\partial \rho}{\partial p} & p < \rho_0 \\ 0 & p \geq \rho_0 \end{Bmatrix}, \tag{3}$$

Each point in the path is modified according to the sum of the repulsion force, f_r, and the stretching force, f_s.

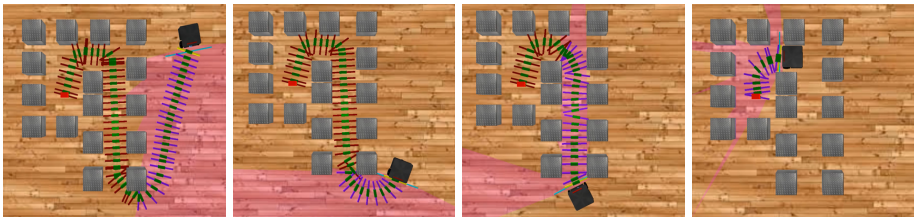
$$p_i^{t+1} = p_i^t + f_r + f_s \tag{4}$$

and, with time, the whole path evolves towards an equilibrium point following a form of downhill search [16].

As an improvement in our implementation, at each iteration of the algorithm the robot is virtually moved along the path to check for collisions using the complete geometry of its base projected on the floor plane. In case that the robot gets too close to some obstacle in this simulation, a situation that can occur if the robot's shape is not circular, the α gain multiplying the f_r force is increased by a small amount. If the situation persists, the robot eventually stops and the path planner searches for an alternative trajectory to the target. Figure 3 shows this situation in a sequence where an obstacle not include in the model steps into the path forcing the robot to stop and replan a new path to the goal.

3.4 Path follower

This component also activates when a new path is created and drives the robot along it using a local controller that combines several measurements relating the



(a) (b) (c) (d)

Fig. 2: Navigation sequence, (a) to (d), showing the interaction of the *Projector* and *Controller* components. Note how the visible part of the path is submitted to repulsion and stretching forces resulting in a smooth trajectory. The pink shadow represents the laser field.

robot to the path. First the rotational velocity v_r is computed as a quantity proportional to the angle that the robot's nose makes with the tangent to the path at the closest point. We set a high gain so most of the turn required to align with the road is made at the beginning and the laser field becomes effective,

$$v_r = 0.7 * \theta \tag{5}$$

with a limiting condition:

$$v_r = \begin{cases} v_{max} & v_r > v_{max} \\ -v_{max} & v_r < -v_{max} \end{cases} \tag{6}$$

Assuming an omnidirectional robot we need to obtain speed values for the x and y directions as an v_t vector. Initially, we want to align this vector with the tangent to the road at the closest point to the path, and pointing slightly inwards if the robot is displaced laterally from the trajectory. Being l_r the unitary line tangent to the path at the closest point to the robot, we rotate it towards the path by an amount proportional to the signed distance d from robot to the point,

$$l_r^* = R(\alpha * sgn(d)) * l_r \tag{7}$$

with,

$$R(\alpha) = \begin{bmatrix} \cos(e^{-\frac{1}{|d|}\lambda}) & \sin(e^{-\frac{1}{|d|}\lambda}) \\ -\sin(e^{-\frac{1}{|d|}\lambda}) & \cos(e^{-\frac{1}{|d|}\lambda}) \end{bmatrix} \tag{8}$$

with λ serving as a scaling parameter.

This expressions give the direction of the translation speed vector v_t. Now we scale it according to the conditions of the path and the location of the target. To do this, different inhibiting factors are computed and multiplied by the maximum speed that the robot can achieve. The robot, we could say, is refrained from jumping ahead by conditions related to the path-robot interaction. Defining the

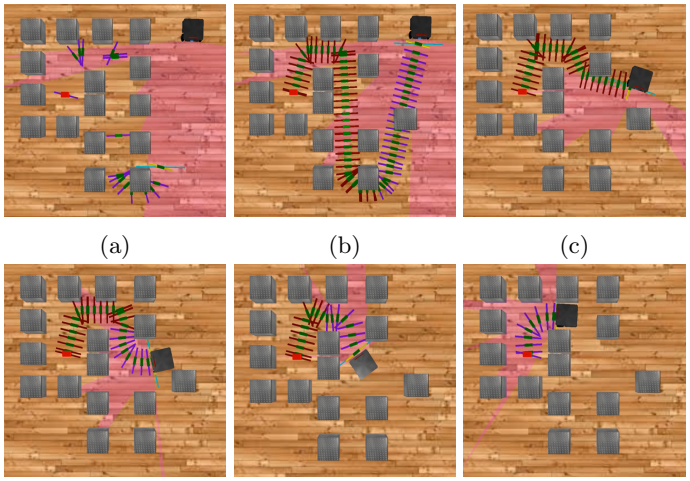


Fig. 3: Navigation sequence showing an initial plan to the target (a) that is interrupted by the appearance of an unexpected obstacle (b). The blocking is detected as a threshold in the amount of free path ahead. The path planner detects the situation and computes a new path (c) that is again projected and followed by the robot (d,e) until the goal (e).

robot's maximum speed as t_{max}, the first factor is a Gaussian function of the local curvature c. For in straight segments with $c = 0$ the factor C takes the value 1 and has no effect on the initial t_max speed,

$$C = e^{-\frac{absc}{\lambda}} \tag{9}$$

The second factor is also a Gaussian function of the inverse of the distance to the target d_t. When the robot is far from it, a high value of d_t produces a result close to 1. Conversely, when the robot is approaching the target, the factor quickly lowers to 0,

$$T = e^{-\frac{1}{d_t}\lambda} \tag{10}$$

The third factor couples non-linearly the rotation speed v_r with the translation speed v_t,

$$R = e^{-\frac{v_r}{\lambda}} \tag{11}$$

The final translation vector l_r^* is assigned a module that is the product of the three factors times the maximum robot's velocity.

$$l_r^* = t_{max} * C * T * R \tag{12}$$

Finally, the forward and sideways translational speeds sent to the robot low level motor controller are obtained as the l_r^* components,

$$v_{tx} = l_r^* \cdot [1, 0]^t$$
$$v_{ty} = l_r^* \cdot [0, 1]^t$$

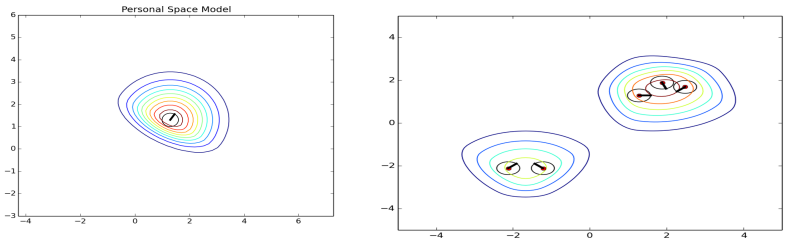
One interesting thing in the Path Controller interaction with the path is that it sees it as a continuous object, as opposed to a collection of ordered points, and functions such as *robot-perpendicular-distance-to-path* or *angle-between-robot-and-path-tangent-at-closest-point* are provided to avoid reasoning about each waypoint conforming the path.

4 Social Navigation

Navigation among inanimate, non-human obstacles can be tackled quite efficiently with the navigation sub-system described above. However, when humans enter the robot's scenario they bring with them social rules denoting how moving things must behave around them. The combination of human social rules with robot navigation among humans have brought to life the *socially-aware robot navigation* new sub-field [25]. As this author claims, "The basic idea is simple: if a mobile robot can understand and follow social conventions then the humans will better understand robot intentions and will find the co-existence with robots more comfortable."

Our goal in this paper is to show how the current navigation agent can be extended to a socially-aware navigation agent using the resources provided by CORTEX, specifically, the architecture's means to dynamically share information among agents. The access to an extended description of the world is the crucial step needed by the navigation agent to become social, specifically, the information about the presence of people nearby, including their orientation and an estimation of their attitude. These information is provided by a *Person* agent that detects people using the data provided by the Microsoft Kinect II SDK. The agent combines this data with a local perceptive state and injects the processed information as a sub-tree in the DSR which, in turn, publishes all structural changes in the graph and also all changes to the attributes of nodes, such as the joint's angles.

Once the information of people nearby is accessible to the navigation agent, the modified algorithm uses Proxemics [36][35] to determine a comfort zone around people and decide what path to follow and how much security distance must be kept. The analysis of the situation of the people around the robot and the computation of their comfort zones has been recently presented in [32][27], where and algorithm has been derived to delimit the comfort areas around people that should not be invaded by the robot. These areas are expressed as a list of 2D polygons, as shown in Figure 4, where three configurations of people give rise to different comfort zones. What interests us in this work is how this new



(a) One individual. (b) Two groups of 2 and 3 people.

Fig. 4: Comfort areas for one and several individuals. The final shape depends on the position and orientation of the individuals. See [32]

interpretation of people wandering around the robot can be translated to the Navigation agent with a small effort, thanks to its decoupled internal architecture that is self-similar to that of CORTEX's. The polygons computed by the algorithm are injected in the navigation sub-system as a hallucination affecting the real laser measurements. In the laser array humans are detected as two small obstacles created by the legs. Now, the laser is modified in real-time to expand those regions into the area covered by the polygons. Figure 5 shows two missions of the robot Shelly in our lab. The first one using the original navigation agent that finds a way through the two people talking, and the second one that decides to detour by an alternative route that avoids the group.

5 Conclusions

We have presented ongoing work dealing with the problem of integrating a new social navigation algorithm inside an existing cognitive architecture. The problem is specially interesting because when moving from standard navigation - i.e. among inanimate objects- to socially-aware navigation, what is needed is access to a new kind of information, the position of nearby people, that is being managed by a different module, probably placed in a distant part of the system. This kind of situations are among the ones we expect to be handled efficiently by the CORTEX architecture. The option of a shared working memory of the nature described in this paper might be controversial, specially in the light of more distributed or dynamicists approaches to intelligence [28], but we believe that it is a valuable tool to study autonomy in social robotics. The outcome of this choices is to be evaluated empirically in forthcoming experiments performed in real-world HRI scenarios. The introduction of social navigation in CORTEX with minimum changes is one step in this direction that underpins the potential of CORTEX to tackle more challenging scenarios.

A next experiment that deepens in this line of reasoning will situate the robot behind a person that blocks its way to the target. In this situation, social rules suggest that the robot stops and alerts the human of its presence, instead of

(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 5: Experiment in our adapted apartment -Autonomy Lab- showing the robot navigating without (a-d) social rules and with (e-h) them among a group of chatting people.

just moving forward around her. Once the robot stops and talks to the person several things can happen. For example, the person may turn around first and step aside afterwards, or she can remain in the same position, without turning, or still she can turn around and ask the robot for some new mission. Social robots must be able to handle this variability but to do that other parts of the architecture must intervene according to the unfolding of events. A conversational module, i.e. *Dialogue* in CORTEX- has to activate and warn the person blocking the way and, possibly, engage in a longer conversation. Also, the *Person* agent has to inject additional symbolic information about the person, so the *Planner* agent can obtain a new plan in accordance with the situation, possibly reasoning about the person's immediate intentions. We expect that these new missions can be addressed by CORTEX with minimum changes affecting only the domain knowledge and some functionality of the existing agents.

Acknowledgments. This work has been partially supported by the MICINN Project TIN2015-65686-C5-5-R, by the Extremaduran Goverment project GR15120, by the Red de Excelencia "Red de Agentes Físicos" TIN2015-71693-REDT, European project 0043-EUROAGE-4-E (Interreg POCTEP Program), MEC project PHBP14/00083 and by Extremadura Government grant EMOROBOTIC IB16090.

References

1. Kotseruba, I., Gonzalez, O.J.A., Tsotsos, J.K.: A Review of 40 Years of Cognitive Architecture Research: Focus on Perception, Attention, Learning and Applications.

Technical report (2016)
2. Kurup, U., Lebiere, C.: What can cognitive architectures do for robotics? Biologically Inspired Cognitive Architectures **2** (2012) 88–99
3. Wei, C., Hindriks, K.V.: An agent-based cognitive robot architecture. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7837 LNAI** (2013) 54–71
4. Beetz, J., van Berlo, L., de Laat, R., van den Helm, P.: bimserver. org–An Open Source IFC Model Server. Proceedings of the CIB W78 2010: 27th International Conference–Cairo, Egypt (Weise 2006) (2010) 16–18
5. Haber, A., Sammut, C.: A Cognitive Architecture for Autonomous Robots. Advances in Cognitive Systems **2** (2013) 257–276
6. Bandera, A., Bandera, J., Bustos, P., García-Varea, I., Manso, L., Martínez-Gómez, J.: CORTEX: a new Cognitive Architecture for Social Robots. In: EUCognition Meeting - Cognitive Robot Architectures, Viena (2016)
7. Metzler, T., Shea, K.: Taxonomy of Cognitive Functions. In: Proceedings of the 18th International Conference on Engineering Design. Volume 1. (2011) 330–341
8. van der Hoek, W., Wooldridge, M.: Chapter 24 Multi-Agent Systems. In F. van Harmelen, Lifschitz, V., Porter, B., eds.: Handbook of Knowledge Representation. Elsevier B.V. (2008) 887–928
9. Romero-Garcés, A., Vicente Calderita, L., Martínez-Gómez, J., Bandera, J.P., Marfil, R., Manso, L.J., Bustos, P., Bandera, A.: The cognitive architecture of a robotic salesman. In: XVI Conferencia de la Asociación Española para la Inteligencia Artificial,, Albacete (2015)
10. Bandera, A., Bandera, J.P., Bustos, P., Calderita, L.V., Fern, F., Fuentetaja, R., Garc, F.J., Iglesias, A., Luis, J., Marfil, R., Pulido, C., Reuther, C.: CLARC : a Robotic Architecture for Comprehensive Geriatric Assessment. In: Workshop on Physical Agents. Volume 1., Málaga (2016) 1–8
11. Manso, L., Bustos, P., Bandera, J., Romero-Garcés, A., Calderita, L., Marfil, R., Bandera, A.: Deep representations for collaborative robotics. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **10087 LNCS** (2016) 1–6
12. Manso, L., Bachiller, P., Bustos, P., Calderita, L.: RoboComp: a Tool-based Robotics Framework. Lecture Notes in Computer Science. Simulation, Modeling and Programming in Autonomous Robots **6472** (2010) 251–262
13. Bustos, P., Manso, L.J., Bandera, J.P., Romero-Garcés, A., Calderita, L.V., Marfil, R., Bandera, A.: A unified internal representation of the outer world for social robotics. In: Advances in Intelligent Systems and Computing. Volume 418. (2016) 733–744
14. Fernández, F., Martínez, M., García-Varea, I., Martínez-Gómez, J., Pérez-Lorenzo, J.M., Viciana, R., Bustos, P., Manso, L.J., Calderita, L., Gutiérrez, M., Núñez, P., Bandera, A., Romero-García, A., Bandera, J.P., Marfil, R.: Gualzru's path to the advertisement world. In: IROS-FineR Workshop and CEUR Proceedings. Volume 1484. (2015) 55–65
15. Romero-Garcés, A., Calderita, L.V., Martínez-Gómez, J., Bandera, J.P., Marfil, R., Manso, L.J., Bandera, A., Bustos, P.: Testing a fully autonomous robotic salesman in real scenarios. In: IEEE International Conference on Autonomous Robots Systems and Competitions, Vilareal, Portugal (2015) 1–7
16. Quinlan, S., Khatib, O.: Elastic bands: connecting path planning and control. [1993] Proceedings IEEE International Conference on Robotics and Automation (1993)

17. Brock, O., Khatib, O.: Elastic Strips: A Framework for Motion Generation in Human Environments. The International Journal of Robotics Research **21**(12) (2002) 1031–1052
18. Yang, Y., Brock, O.: Elastic roadmaps - motion generation for autonomous mobile manipulation. Autonomous Robots **28**(1) (9 2009) 113–130
19. Hirsch, K., Brandt, T.: An elastic beam approach to predictive vehicle motion planning. Pamm **7**(1) (12 2007) 4130025–4130026
20. Keller, M., Hoffmann, F., Bertram, T.: Planning of Optimal Collision Avoidance Trajectories with Timed Elastic Bands. 19th World Congress of ... (2014) 9822–9827
21. Haut, M., Manso, L., Gallego, D., Paoletti, M., Bustos, P., Bandera, A., Romero-Garcés, A.: A navigation agent for mobile manipulators. In: Advances in Intelligent Systems and Computing. ROBOT2015 Conference. Volume 418. (2016) 745–756
22. Kavraki, L.E., Svestka, P., Latombe, J., Overmars, M.: Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. IEEE Transactions on Robotics and Automation **12**(4) (1996) 566–580
23. LaValle, S.: Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical report, TR-98-11. Computer Science Dept. Iowa State University (1088)
24. Kuffner, J., LaValle, S.: RRT-connect: An efficient approach to single-query path planning. Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065) **2**(Icra) (2000) 995–1001
25. Rios-Martinez, J.A.: Socially-Aware Robot Navigation: combining Risk Assessment and Social Conventions. PhD thesis, UNIVERSITY of GRENOBLE (2013)
26. Nuñez, P., Manso, L.J., Bustos, P., Drews-Jr, P., Macharet, D.G.: A Proposal for the Design of a Semantic Social Path Planner using CORTEX ˜. In: Workshop of Physical Agents 2016. Volume 1., Málaga, Spain (2016)
27. Macharet, D.G., Vega-magro, A., Manso, L., Bustos, P.: Socially Acceptable Robot Navigation over Groups of People ˜. In: RO-MAN 27th IEEE International Symposium on Human and Robot Interactive Communication. (2107)
28. Beer, R.: Dynamical approaches to cognitive science. Trends in cognitive sciences **4**(3) (3 2000) 91–99
29. Kirby, R.: Social robot navigation. Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May (2010).
30. Ratsamee, P., MaeKazuto, Y., Horade, M., Kojima, M,. Arai, T.: Social interactive robot navigation based on human intention analysis from face orientation and human path prediction. Robomech Journal, Vol: 2, (2015).
31. Scandolo, L. and Fraichard, T.: An Anthropomorphic Navigation Scheme for Dynamic Scenarios. In proceedings on IEEE International Conference on Robotics and Automation, pp. 809-814, (2011).
32. Núñez, P. and Manso, L. and Bustos, P. and Drews-Jr, P. and Macharet, D.G. A Proposal for the Design of a Semantic Social Path Planner using CORTEX. Workshops on Physical Agent, pp. 31-37, (2016)
33. Kruse, T., Kumar, A., Alami, R. and Kirsch, A.: Human-aware robot navigation: A survey. Robotics and Autonomous Systems, vol. 61, pp. 1726-1743, (2013).
34. J. Mumm and B. Mutlu. Human-robot proxemics: physical and psychological distancing in human-robot interaction. In Proceedings of the 6th international conference on Human-robot interaction (HRI). New York, NY, USA: ACM, pp. 331–338, (2011).

35. M. Walters, M. Oskoei, D. Syrdal, and K. Dautenhahn. A long-term Human-Robot Proxemic study. In IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 137 –142, (2011).
36. R. Mead and M. J. Mataric. A probabilistic framework for autonomous proxemic control in situated and mobile human-robot interaction. In Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, ser. HRI '12. New York, NY, USA: ACM, pp. 193–194, (2012).