Multimodal Interaction with Loki

P. Bustos, J. Martínez-Gómez, I. García-Varea, L. Rodríguez-Ruiz, P. Bachiller, L. Calderita, L.J. Manso, A. Sánchez, A. Bandera, and J.P. Bandera

***Abstract*—Developing a simple multimodal interaction game with a 31 dof's mobile manipulator can become a challenging enterprise. A conceptually simple task quickly unfolds into a rather complex ensemble of driver-oriented, framework-based, software-enabled, state-machine controlled mechatronics. In this paper we propose a multimodal interaction game designed to test the initial steps of a cognitive robotics architecture called RoboCog. In the game, a human shows an object to the robot and asks him to touch it with one of his hands. Loki, the robot, searches, gazes, represents and touches the object, then talks and waits for new events. The game goes on until the human player decides to quit. In this paper we describe the steps taken to achieve this goal, analyzing the decisions made in terms of architectural choices and describing how the sequential control of multimodal resources was built. To conclude, several snapshots of the game are presented and commented along with video material of Loki playing with a volunteer.**

***Index Terms*—Mobile manipulators, Humanoid Robotics, Human-Robot Interaction**

I. Introduction

HU man-robot interaction is becoming a pressing goal in the global robotics agenda. If robots are to accompany people in their daily activities, they need to know how to interact with them using different sensorial modalities. Speech, facial expressions or gestures are among the most basic human ways of interacting and sharing information.

In this work, we present an initial effort towards modeling, building and understanding simple multimodal HR interaction tasks using a quasi-humanoid mobile manipulator called Loki [1]. The experiment consists on a simple game between a human and a robot. The human introduces herself and asks Loki to play the game. Upon acknowledge, she shows a yellow ball to the robot and it starts to track it, continuously fixating its gaze upon the ball with an RGBD sensor placed in the fronthead. After a human verbal indication, the robot reaches the ball with its hand and waits for a new interaction, or moves its arm back to a resting position after some courtesy delay. The robot tracks the object and accepts new speech commands during the whole span of the game. The development of this interaction game involves several problems such as generalized inverse kinematics, RGBD object detection and tracking, speech recognition and synthesis, and sequential task execution. All these problems present a significant degree of complexity, but the most challenging aspect is the integration and coordination of these capabilities in an expandable architecture that, further on, will facilitate the design of new and more complex tasks.

The interaction game is a three-stage process where the following subtasks are performed: scan and track the ball, acquire an internal representation of the position (with respect to the robotic arm) and try to touch the ball.

Fig. 1: The mobile manipulator Loki

The interaction game presented here is an early test of the robotics cognitive architecture, RoboCog, in which several research groups are currently involved. RoboCog has its initial inspiration in the classic three layer architectures described by Gat [2] and in the Simulation Theory of the Mind, see for example [3]. Reactive, planning and executive modules access a complex, multi-faceted representation of the robot itself, its environment and the agents in it. This structure is called *deep representation* and can emulate the result of virtual actions done by the robot in its represented world. RoboCog combines real and virtual actions at different levels of abstraction to generate the best possible behavior given the current task.

During the last few years, the robotics community has developed fully autonomous robots capable of performing potentially interesting tasks in indoor environments such as the one presented in this paper. Despite rather sophisticated planners that support adaptive plan execution are available (e.g., [4]) most of these works are limited to very specific tasks, so the research effort is focused on task execution and the low level algorithms that enable robots to detect the elements of the environment, approach them and pick them up.

The work presented in [5] describes the architecture and several design decisions of a robotic butler. In the paper, task execution is solved using behaviors activated by a hierarchical

P. Bustos, P. Bachiller, L. Calderita, L.J. Manso and A. Sánchez are with the University of Extremadura.

I. García-Varea, J. Martínez-Gómez and L. Rodríguez-Ruiz are with the University of Castilla-La Mancha, Albacete, Spain

A. Bandera and J.P. Bandera are with the University of Málaga, Málaga, Spain

concurrent state machine. Besides few additional features such as the one presented in [6], that enables robots to learn these state machines, most task-specific advanced robots follow the same approach.

The rest of the paper shows the steps taken to set up the HRI game based on RoboCog and is organized as follows. In section II a brief introduction to Multimodal HRI is provided. Section III provides a brief description of the robot Loki. Section IV introduces the basic building blocks of RoboCog. Section V describes how Loki and RoboCog are instantiated in a multimodal interaction game called "touch the ball" Finally, in section VI the results are analyzed along with the ongoing research in this topic.

II. Multimodal interaction

In the scientific literature, multiple definitions of Human-Robot Interaction (HRI) can be found. HRI is basically devoted to the understanding, design and evaluation of robotic systems for their use by or with humans [7]. Moreover, Multimodal HRI can be considered as the study of the interactions between humans and robots using multiple sources of information or channels to provide a natural way of communication.

The interaction between robots and humans is inherently influenced by the proximity between them. According to that, HRI is classified as remote and proximate interaction. Remote interaction is considered when the human and the robot are separated temporally or spatially. This type of interaction focuses on teleoperation, supervisory control and telemanipulation. Proximate interaction is considered when human and robot are both placed in the same scene, and is mainly focused on the so-called social robots, which includes social, emotional and cognitive aspects of interaction [7]. The types of communication that exist in this interaction can be grouped into: oral, visual and gestural.

In this work we are focused in the proximate multimodal interaction approach, where a social mobile robot communicates with a human, with the final goal of playing an interactive game named "touch the ball", where the three types of communication are addressed.

III. Loki the robot

As the human counterpart in the game we use the robot Loki [1] (in Fig. 1 its current aspect is shown), which is an autonomous mobile robot built as a collaboration among several entities: the University of Castilla-La Mancha, the University of Extremadura, Robotnik S.L.L and IADex S.L. It is composed of a mobile base, a rigid back spine, a torso with two arms and hands and an expressive head.

The base of the robot has been designed to support a load of 200 Kg and can accommodate two 36Ah/24V batteries, power supplies, a battery charger, a DC/AC 2 KW inverter, two lasers or four Asus Xtion RGBD cameras attached to each side of the base,and a dual-socket, 12-cores, liquid-cooled Xeon board holding a NVIDA GTX650Ti GPU. This configuration provides enough autonomy and processing power to host our complete cognitive architecture on board.

Each arm is composed of four Schunk servo-drives in a human-like upper arm configuration (3 degrees of freedom for the shoulder and 1 for the elbow) and a forearm built in RoboLab with 3 additional dofs (a rotation along the forearm and two orthogonal rotations in the wrist). This two last dofs are built using a 3R Stewart platform that provides a great holding torque for the wrist.

Attached to the top of the torso, Loki holds the expressive head *Muecas* (see Fig. 2). This head has been developed in parallel to Loki and has a 4 dofs neck that uses the same kinematic construction as the forearms. The head holds a binocular visual system composed of two PointGrey Flea2 1Mp cameras with 6mm focal lenses and a RGBD sensor placed in the fronthead. The cameras are housed inside two hollow spheres made in Teflon. These eye-balls can pan independently and have a common tilt. The eyes are moved by means of three linear motors from Faulhaber that provide enough force to avoid the need of gear trains and to reach maximum angular speeds close to 600 deg/sec. *Muecas* also has an articulated yaw driven by a micro-servo and 2 DOF eyebrows, controlled by 4 servos as well.

(a) Eyes detail (b) Complete head

Fig. 2: Muecas head

IV. RoboCog: a cognitive architecture based on internal simulation

RoboCog is a cognitive architecture being developed as a common effort among several research groups at different universities including UCLM, UMA, UC3M, UJ and UEX. During the last five decades, many architectures have been proposed to model general intelligence in artificial agents and humans. Starting with some of the paradigmatic projects from the golden age of classic AI such as CyC [8], Prodigy [9], SOAR [10] or ACT [11], which many are still alive and in active development, passing through the skimming filter of the Behavior-Based AI in the late eighties [12] and arriving to the integrative look of three-layer architectures compiled by Gnat [2] in the early nineties, most of these control schemes center the debate on the integration of the deliberative/reactive spectrum into a working computational design. Several reviews have been recently published [13], [14] showing the history, evolution and current state of the topic.

A less common approach in the realm of cognitive architectures for robots is the so called Simulation Theory of Cognition [3], [15], [16], [17]. This theory advocates the use of an internal model that is sophisticated enough to anticipate and simulate the outcome of virtual actions in the form of virtual perceptions. Quoting Holland [3]:

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

In cognitive neuroscience, the idea of the brain continuously generating predictions that anticipate the relevant future is widely accepted and important research is being conducted to prove it as one of the basic building blocks of intelligence [18], [19], [20]. This idea is so powerful because it can be applied hierarchically at many abstraction levels, from the short timestamp of a spinal cord reflex, to the anticipatory perception of a distant approaching car, using a priori information recovered from episodic memory in combination with low-frequency visual and auditive indicators.

To translate the core of these ideas to robotic computational architectures, a notion of internal or inner model is often used. This model represents actively the agent's environment and holds enough structure to act as a virtual simulator. Although most architectures use some kind of internal model, i.e. symbolic domain knowledge in task planners or grid representations of space in path planners, a more integrated, flexible and coordinated representation is needed if we want to perform complex simulations at different abstraction levels. We use the term, *deep representation* (DR), to refer to this computational modules that can be used to represent, update and simulate the current state of the robot itself, the proximal environment around it and of the agents in it.

The DR represents the geometry of the scene and a set of symbolic attributes and predicates that are relevant for the rest of the architecture. Symbols and predicates are computed from the model at any time and their description is available to the rest of the architecture through a shared ontology. The DR of a robot is a central repository of elements and events in its ego-space. It can be queried for geometric data such as the coordinates of some object nearby, and also to evaluate symbolic predicates such as if its body will get through a narrow passage. It also can export its entire state as a PDDL expression to be used by task planners. The DR is implemented using the RCIS simulator of the RoboComp ecosystem [21] working as an internal updatable model.

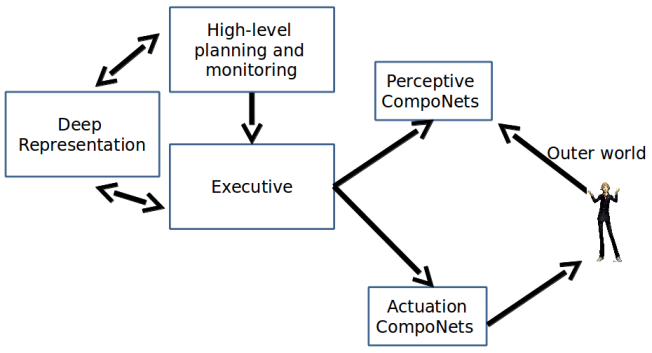

Fig. 3: Outline of the RoboCog architecture.

In the design of RoboCog we have extended the basic three-layer idea with a DR module, that provides the necessary simulation support. As Figure 3 shows, the extended set, deliberative-executive-behavior-DR (DEBD) defines the overall external view of the architecture. Behaviors are implemented as networks of software components called *CompoNets*. Typical behavior modules proposed for current mobile manipulators usually include complex activities, such as planning the movements of the body in cluttered space (including standard navigation), object detection recognition and manipulation, emotion recognition and generation, speech recognition, synthesis and dialog generation, that need a rich underlying software support.

A vaguely related idea to the *deep representation*, widely spread in the eighties, is the blackboard architecture where a common space is used by several experts to gradually solve a problem [22]. RoboCog however, proposes a much more specific and structured content, representing the robot itself and its surrounding space. DR is much closer to virtual worlds than to static memory structures and are designed for real-time functioning.

In the next section, an early implementation of RoboCog for a basic multimodal interaction game is described in detail. With the exception of the planner, that has been replaced in this experiment by a hierarchical state machine, all the others elements of the architecture are in use.

V. Example of use: "touch the ball" game

As explained in Section I, the "touch the ball" game consists on a human showing the robot a ball at different places and asking it to touch it with its hand, while maintaining a simple conversation. Our initial RoboCog implementation allows the robot to engage in the game using and reacting to different sensorial modalities. The architecture has been designed as a distributed system whose basic elements are software components that can be recursively nested into *CompoNets*. The framework used for the C++ implementation is RoboComp [23], which uses a DSL-based target code generation system. This system allows the user to select an specific communication middleware, such as Ice or Nerve [24] in generation time.

The Executive, see Figure 3, drives a hierarchical state machine that provides the necessary sequential order. This machine activates several *CompoNets* implementing the low level behaviors. This behaviors publish the changes in the world that they detect and also actuate on the drivable parts of the robot. The DR is implemented as an instance of RoboComp's RCIS simulator [23]. Typical changes of state detected by the behaviors, such as a ball position change, are directly transmitted to the DR via its management interface. As a consequence, the globally accesible internal representation is updated and the rest of the behaviors become aware of it. For structural changes, such as the appearence of a new object, the Executive is notified first and some comprobations are made before introducing the new object, to preserve the integrity and internal coherence of the DR.

The rules of the game are simple: a) the robot is expected to determine, in real time, a feasible arm configuration to touch

a ball; b) the ball is moved by the human player who controls the game; c) the human determines when the game starts and finishes, and when the robot has to touch the ball; d) the human is also in charge of indicating that the robot has properly touched the ball so that the robot can move its arm back to its home position. Concurrently with all the searching, tracking and touching activity, the robot must interact with the human player understanding asynchronous commands and generating utterances that signal some internal transition or anomalous situation.

A. RoboCog implementation

To describe the architecture, we will first identify all the components and their relationships. The main component is the *Executive*. This component manages the set of *compoNets* and determines the sequential behaviour of the system by implementing a concurrent, hierarchical state machine. This machine runs on the engine provided by the Qt library.

- BallTracker: this component uses the RGBD camera of the robot to continuously track the position of the ball. Once the ball is localized, its 3D coordinates and shape are introduced into the DR and made visible to the other behaviors. Posterior changes in the estimated position of the ball are made available for the rest of components by publishing a topic [1].
- BodyInverseKinematics: computes a generalized inverse kinematics solution to the different parts of the robot's body. To do so, this modules uses the internal representation of the robot hosted in DR to compute collison-free trajectories. Currently, the solution is provided for the neck-head subsystem and for the left arm. Using these solutions, this component is able to drive these body parts to a target position (the ball in that specific case) and publish a topic when the goal has been reached.
- Dialogs: processes the transcription (and the confidence value) obtained from the speech recognition component. Accurate transcriptions are translated into commands for the robot, whereas low confidence results require some kind of confirmation from the user.
- SpeechGenerator: allows the robot to speak to the human player using the Verbio speech synthesis algorithm.
- CommandGenerator: provides the human player with a visual interface to generate commands that can be used by human players with speech disorders. This component can be removed for standard games.
- SpeechRecognition: this component uses the Microsoft Kinect Sensor and SDK software. It records the speech signal using the robot's microphones and then transcribes the speech uttered by the human player.
- DR: provides a dynamic *deep representation* of the robot and the environment. The perception modules update the representation and everone can access it or request a copy of it to simulate the outcome of possible actions, as in the case of the BodyInverseKinematics module.

[1]The term *topic* refers to a data structure that can be broadcasted to modules that are *subscribed* to it.

In addition to these behaviors, there are other components running in the system that provide access to the hardware. The motors of the arm and the head, and the Kinect sensor are controlled by specific components that wrap the manufacturer libraries and provide concurrent access to the rest of the system. These components are *JointMotor* and *RGBD*.

In the current implementation of RoboCog, we use the Ice middleware that the ZeroC company offers with open source licence. Ice provides both a native client/server communication system, and a a publish/subscribe event distribution service named IceStorm[2]. This service allows to decouple component connections: clients are now considered publishers and servers subscribers. A single publisher can generate and send data to any number of subscribers. Icestorm is used in this work to deal with three events:

- **The ball has been touched**: BodyInverseKinematics plays the publisher role and the Executive subscribes to its events. This signal represents when the robot internally detects that it has achieved the desired joint configuration.
- **The position of the ball has been detected**: BallTracker takes the publisher role while the Executive subscribes the events. This signal is sent whenever the ball changes its position. When the ball is out of the field a different signal es emitted.
- **A new command has been detected**: Dialogs and/or CommandGenerator play the publisher role while the Executive subscribes to the events. It is generated after an explicit command is uttered by the human player.

The rest of the communications are summarized below:

- **Executive** $\rightarrow$ **BodyInverseKinematics**: Based on the information received from the ball tracker, the executive sends to BodyInverseKinematics the new goal position to be achieved.
- **BodyInverseKinematics** $\rightarrow$ **JointMotor**: Values are sent by BodyInverseKinematics to the arm joints and obtains the final values from the robot.
- **BallTracker** $\rightarrow$ **RGBD**: BallTracker obtains the stream of RGBD data from the camera installed on the robot.

B. Ball detection and tracking

Ball detection and tracking is solved by a unique component using the RGBD information obtained by the robot camera. The component represents the possible situations using three states: waiting, detecting and tracking. The waiting state corresponds to the initial situation when the component is waiting for a description of the target. Such description is given by a color value (hue and saturation), assuming untextured targets, and the target size. When the component receives the target description, it changes its state to detecting. Detection is carried out by selecting homogeneous regions of the image whose color values differ less than a given threshold from the target color. From the selected regions, the component chooses the one which better approach the target size. If no region coherent with the target description is found for a while, the component returns to the waiting state in order to wait for

[2]http://www.zeroc.com/icestorm/

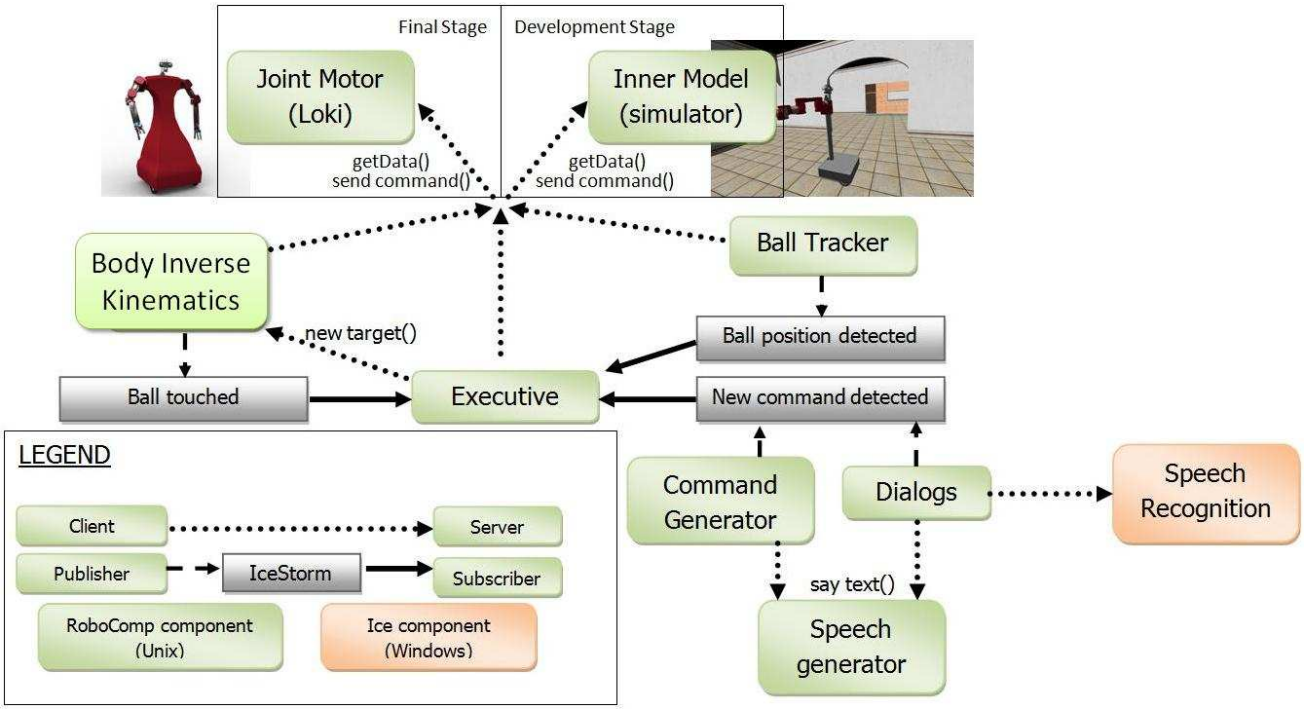

Fig. 4: Global system architecture. The drawing shows the main components of the architecture and the type of communications that occur among them.

another target description.
Once the target is detected, the component enters the tracking state. To track the target, it creates a model HSD (Hue, Saturation, Depth) histogram from the target image window and proceeds as follows:

1) Create a probability image from the current image and the model histogram by copying, for each pixel in the image, the corresponding bin value of the model histogram.
2) Run *CamShift*: find the object center using *MeanShift* on the probability image and adjust the target window size according to it.
3) If *CamShift* reaches the convergence:
   a) Compute the HSD mean and standard deviation of the new target window.
   b) Adjust the *Hue*, *Saturation* and *Depth* ranges of the model histogram according to the mean and the standard deviation.
   c) Compute the model histogram of the new target window.

Once the previous process has finished, the component makes an additional verification over the new target position. It estimates the target size using the depth values of the new window and rejects the tracking result if it does not match with the real size of the target. If the tracking process does not provide a good result during a certain number of iterations, the component returns to the detecting state and restarts the detection process.

C. Arm reaching and head gazing

The problem of positioning an open articulated chain of joints in pose space, $SE(3)$, is generally solved by computing the inverse of the non-linear forward kinematics function of the chain. For a mobile manipulator endowed with an expressive head, like Loki, touching an object might involve most of the joints in the body, specially if natural movements are sought. The head must gaze the object before the arm starts to move and the base might turn slightly in search of a more comfortable HRI posture. Of course, a movement of the base implies a change in the position of the head and the arm, so there must be a final consensus among the body parts. Although this body-parts coordination is still ongoing research, a simple requisite is to conceive the body as a set of parts that whose configuration can change dynamically. For example, an arm can include the first four joints for a simple ”touch the ball” task, or seven joints if the wrist is included for a more complex ”prepare to grab” task, or eleven joints if the hand is included for a ”grab this mug” task, or even 13 joints if the mobile base is also included in the kinematic chain and the task extends to ”grab that mug on the table”. To achieve this flexibility we use a generalized inverse kinematics algorithm for all and any open chains in the robot.

In Loki, both arms and the neck stand as three kinematic chains departing from a central point in the torso. Each arm has eleven dofs, including the hands. The neck is built as a simple Stewart platform with three prismatic links for wrist rotation plus an additional rotation motor aligned with the forearm. As we want to use the same algorithm for all kinematic chains, the closed structure is virtually converted to an open chain composed of pitch, roll and yaw, through its trivial inverse

kinematics equations that map angles to prismatic lengths. These equations allows us to insert three 4x4 matrices in the open chain representing the neck and accepting angles as input parameters.

The generalized inverse kinematics algorithm used in the implementation of this module is the Levenberg-Macquard least-squares minimization procedure [25], [26]. This algorithm regularizes the solution by continuously switching between Newton's method and gradient descent, to avoid singularities. Given ϕ the set of angles of the chain, J the Jacobian of the forward kinematics and e the error vector, the LM method computes the increments in the joints as:

$$\delta\phi = J^T(JJ^T + \lambda I)^{-1}e \tag{1}$$

Note that the expression is valid for any open chain in the kinematic tree of the robot. To compute the gaze of the head, a virtual stick is placed at the optical center of the internally represented - DR in RoboCog- RGBD sensor, that extends outwards to the current target. The residual is computed as the distance between the tip of the virtual stick and the target.

D. Speech Recognition

Speech constitutes one of the most natural communication modalities for human beings. Owing to this fact, the use of speech can be quite adequate for the supervisor to convey information to the robot. Here, the supervisor can use his voice to control the robot actions while he moves the ball around the environment.

Although speech recognition is a challenging task for several reasons (speaker variability, noisy environments, etc.) the use of statistical approaches has proven to achieve accurate results in several scenarios or environments. From this point of view, speech recognition can be stated as the search for the optimal sequence of words $\hat{w}$ given an input utterance x:

$$\hat{w} = argmax_w Pr(w|x) \tag{2}$$

The maximization in Eq. (2) entails the estimation of a single conditional probability ($Pr(w|x)$ which is usually not feasible due to the scarcity of training data which does not allow for accurately estimating such probability distribution.

Alternatively, we can apply Bayes' rule to Eq. (2) to write:

$$\hat{w} = argmax_w Pr(x|w)Pr(w) \tag{3}$$

As can be seen in Eq. (3) now we have two probabilistic models in the maximization. The first one $Pr(x|w)$ is called *Acoustic model* and deals with the probability that the sequence of words w produce the input speech signal x. The second one, $Pr(w)$, is the *Language model* and it is used to score a transcription hypotheses w so that likely sequence of words in the language are given a high probability and unusual ones are given a low probability. The use of this additional term (language model) can, to some extent, mitigate the errors in the estimation of the conditional probability that relates the input signal x and the transcription and, due to this fact, the approach described by Eq. (3) is preferable to directly modelling the conditional probability in Eq. (2).

Regarding the acoustic models, the input speech signal is firstly segmented into short frames where the signal is assumed to be quasi-stationary. Next, different signal processing techniques are applied until each frame in the signal is represented by a feature vector. Hidden Markov Models (HMMs) are widely used to approach acoustic models. For each acoustic unit (a phoneme or a short sequence of phonemes) an HMM is usually defined as a 3-state model, where, in each state, two transitions are defined. One transition to the same state and another one to the next state. This way, the model is able to cope with the different durations of the acoustic units due to different speakers or other factors. The feature vectors in the signal constitute the observations at each state and this observation are modelled by using a mixture of Gaussian. The estimation of both the transition probabilities and the Gaussian mixtures that generate the feature vectors can be efficiently performed by using the Baum-Welsch algorithm [27].

Language models, on the other hand, deal with the joint probability of a sequence of words. Directly estimating this joint probability is not usually feasible in most of the tasks and therefore, a factorization where each word in the sequence is conditioned to the $n-1$ words is followed. This approach is known as n-gram model [28] where n is the model order that defines the number of previous words considered in the conditional probability. Usually 3-grams or 4-grams are employed where the conditional probabilities are estimated by maximum likelihood.

Finally, once both acoustic and language models have been estimated, the sequence of words $\hat{w}$ corresponding to a specific utterance x is obtained according to Eq. (2) where the maximization is solved through a dynamic programming algorithm called Viterbi search algorithm.

In this work, the automatic recognition system is used along with a dialog component. This way, the speech recognition component copes with the process of transcribing the input voice signal. This transcription is then sent to the Dialogs component, along with a score that indicates the accuracy (or the confidence level) of the transcription. In case of a low score, the SpeechGenerator component is used to require a confirmation from the supervisor.

The SpeechRecognition component has been developed in Ice and runs on a Windows machine. It uses the Kinect Sensor and the Software Developers Kit for Speech, provided by Microsoft[3]. In this work, we have used the Spanish acoustics model provided by the SDK. We programmatically built all the constraints for the speech recognition language model or grammar, according to the constrained set of commands that can be recognized, along with two additional confirmation and rejection utterances.

The Dialogs component processes the confidence score computed by the speech recognizer. Accurate transcriptions are directly published through the IceStorm service but low confidence transcriptions result in an additional step where the robot interacts with the supervisor. This interaction mainly consists in the robot uttering the transcription achieved while

[3]http://www.microsoft.com/en-us/download/details.aspx?id=14373

the supervisor confirms or repeat the initial utterance. As soon as the transcription is validated, it is published using IceStorm. Otherwise, the speech recognition will wait for a new utterance. This process can be observed in Fig. 5.

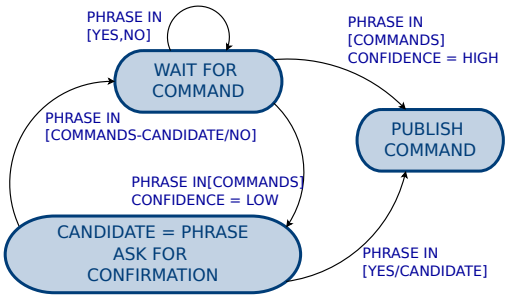

Fig. 5: Dialog process

E. Preliminary Results

In this section, we present the experience of playing the game with the robot through different snapshots. We also introduce some of the main problems that can be faced while playing the game.

Figure 6 shows a sequence where Loki is playing with a human. The top left image shows the initial stage, where the robot is waiting for a human command to start the game. In the next image, the human tells the robot to scan the ball, which involves head and neck movements in Loki. From this moment, the robot starts computing in real time the body configuration that allows it to touch the ball. Then, in the top right image, the human says the order ”touch” and the robot acquires the desired position. While robot is moving its arm, the human player moves the ball to a new position. Since Loki is tracking in real time the ball position, it moves its arm to the new ball position, as can be seen in the bottom left of Fig. 6. The bottom middle image shows a new ”touch” command after the human moved the ball to a new position. Finally, bottom right image shows the end stage, where the human finishes the game.

In spite of fact that the game has been only evaluated in a research laboratory with domain expert players, some conclusions can be drawn. Although a systematic evaluation of the robustness and accuracy of Loki’s movements and perceptions remains to be done, the first set of experiments shows that the overall system is quite stable, moreover ball tracking works reasonably well under a normal playing range of velocities. Also, recovery from tracking failures or when the ball reaches the end of the visual field are correctly handled most of the times. Speech recognition with the Kinect sound technology is more than enough for the kind of dialogs used in the game. Finally, the inverse kinematics algorithm has proved very robust and efficient for real time operation, even when positioning the head and the arm simultaneously. From the point of view of the multimodal interaction, playing with a real robot is an attractive experience for people but repetitive games tend to be boring. New options and possibilities should be added to the ”touch the ball” game to make it more fun and to challenge the cognitive architecture that controls the robot. To handle the complexity of longer sequences of actions and of the unpredictable events that can occur, a generative tool like a symbolic planner must be used. This tool is being included in the current development of RoboCog. Also, human players can be frightened when the robot starts tracking the ball and specially when it moves its arm. This is due in part to the dimensions of the robot, but the effect could be mitigated by working on more natural movements, in which the whole body is involved and special care is take with the accelerations.

VI. Conclusions and Future Work

In this paper we have presented the design and development of a game between a human and a robot involving multimodal interaction. We have evaluated the RoboComp framework in a new challenging scenario in which several modules (speech, inverse kinematics, tracking) have been tested together in real-time. Also, the RoboCog architecture has been briefly described, showing the main lines of research that we are pursuing with this multi-group effort. An implementation of RoboCog over RoboComp has been tested for the ”touch the ball” game, and very promising results have been obtained. The software has been developed on the RCIS simulator and, afterwards, translated to the robot Loki. A set of experiments with humans have shown the validity of the proposal and, also, many interesting ways in which the whole project can be improved.

Based on this experience, we can conclude that RoboComp is a very suitable solution for modular developments. The geographic dispersion of the team members have stretched the current potential of its tools to new limits. But the most important lesson learnt here is the need of a serious discipline and a well-thought methodology if high quality software is to be obtained.

The use of several channels (voice, visual gestures or visual interfaces) improves the acceptance of the game for general public, since the robot is seen as a more friendly and emphatic machine. However, a more complex game has to be implemented if we want to sustain human attention for longer periods of time. A critical part of the system, in which we have already started to work, is a model of the human inside the DR. This model must encompass several levels of information: geometric, physical, symbolic, emotional and intentional. We believe that only when this model of the interacting human could be internally maintained by the robot, its responses will be proactive and identified by humas as truly intentional.

Ongoing work and future research include the use of this architecture and robots for physical and cognitive rehabilitation, and as a household companion. Also, the RoboCog architecture is being improved with new high-level capabilities such as symbolic planning and learning. These improvements will be tested using different serious games as as mean to solve real human problems.

Fig. 6: Loki playing with a human the "touch the ball game". See text for an explanation of the different stages.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación TIN2011-27512-C05-04, AIB2010PT-00149 and TIN2010-20900-C04-03, by the Junta de Extremadura project IB10062, and by JCCM PPII11-0309-6935 project.

REFERENCES

[1] P. Bustos, I. García-Varea, J. Martínez-Gómez, J. Mateos, A. Sánchez, L. Rodríguez "Loki, a mobile manipulator for social robotics" Workshop of Physical Agents, 2012.

[2] E. Gat. "On three-layer architectures". *Artificial Intelligence and Mobile Robots* pp: 195-210, MIT Press, 1998.

[3] O. Holland and R. Goodman "Robots With Internal Models A Route to Machine Consciousness?" *Journal of Consciousness Studies* Num. 4-5, pp 77-109, Vol 10, 2003.

[4] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, R. McEwen, "A deliberative architecture for AUV control". In *International Conference on Robotics and Automation (ICRA)*, pp. 1049–1054, 2008.

[5] J. Bohren, R. Rusu, E.G. Jones, E. Marder and C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the PR2". In *International Conference on Robotics and Automation (ICRA)*, pp. 5568–5575, 2011.

[6] R. Dillmann, R. Becher, P. Steinhaus, "ARMAR IIa learning and cooperative multimodal humanoid robot system". In *International Journal of Humanoid Robotics*, vol. 1, num. 1, pp 143–155, World Scientific, 2004.

[7] M. A. Goodrich and A. C. Schultz. "Human-robot interaction: A survey," *Foundations and Trends in HumanComputer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[8] D.B. Lenat and R. V. Guha "Building large knowledge-based systems: representation and inference in the Cyc project"*Addison-Wesley*, 1989.

[9] M. Veloso, J. Carbonell, A. Pérez, D. Borrajo, E. Fink and J. Blythe "Integrating planning and learning: the PRODIGY architecture" *Journal of Experimental & Theoretical Artificial Intelligence* Vol 7, num. 1, pp 81-120, 1995.

[10] J.E. Laird, K.R. Kinkade, S. Mohan and J.Z. Xu "Cognitive Robotics Using the Soar Cognitive Architecture" *8th International Workshop on Cognitive Robotics* pp. 46-54, 2012.

[11] J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere and Y. Qin "An integrated theory of the maind"*Psychological Review*, 111(4) pp 1036-60, 2004.

[12] R.A. Brooks. "Intelligence without representation" *Artificial Intelligence*. Vol 47, 1991.

[13] W. Duch, R.J. Oentaryo, and M. Pasquier. "Cognitive Architectures: Where do we go from here?" *Frontiers in Artificial Intelligence and Applications* pp: 122-136, IOS Press, 2008.

[14] P. Langley, J.E. Laird, and S. Rogers. "Cognitive architectures: Research issues and challenges" *Cognitive Systems Research*, num. 2, pp. 141–160, vol. 10, 2009.

[15] R. Grush. "The emulation theory of representation: motor control, imagery, and perception" *The Behavioral and brain sciences*, 27(3):377–96, June 2004.

[16] C. Keysers and V. Gazzola. "Integrating simulation and theory of mind: from self to social cognition" *Trends in cognitive sciences*, 11(5):194–6, May 2007.

[17] M. Johnson, Y. Demiris. "Perspective taking through simulation" *Proceedings of TAROS*, 27(3):377–96, 2005.

[18] M. Bar. "Predictions: a universal principle in the operation of the human brain. Introduction" *Philosophical transactions of the Royal Society of London, Series B, Biological sciences*, 364(1521):1181–2, 2009.

[19] M. Bar. "Predictions in the Brain: Using our past to generate or future" *Orford University Press*, 2013.

[20] A. Clark. "Whatever next? Predictive brains, situated agents, and the future of cognitive science" Behav Brain Scicences 36(3):181-204, 2013.

[21] M.A. Gutierrez Giraldo "Progress in RoboComp" *Journal of Physical Agents* Vol 7, Num 1, pp: 38-47, 2013.

[22] B. Hayes-Roth. "A blackboard architecture for control". *Artificial Intelligence 26 (3):* pp. 251321, 1985.

[23] L.J. Manso, P. Bachiller, P. Bustos, P. Núñez, R. Cintas and L. Calderita. RoboComp: a tool-based robotics framework. In *Simulation, Modeling, and Programming for Autonomous Robots*, pp 251–262. Springer, 2010.

[24] M. Henning and M. Spruiell. Distributed programming with ice. ZeroC Inc. Revision, vol. 3. 2003

[25] S.R. Buss, J. Kim. "Selectively Damped Least Squares for Inverse Kinematics" *Journal of Graphics Tools* vol. 10, no. 3, pp: 37-49, 2005.

[26] S.R. Buss. "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods" *Unpublished survey. http://math.ucsd.edu/ sbuss/ResearchWeb/ikmethods/iksurvey.pdf* vol. 10, no. 3, pp: 37-49, 2005.

[27] L. Rabiner, "A tutorial on hidden Markov Models and selected applications in speech recognition" *Proceedings of the IEEE*, pp. 257–286, 1989.

[28] F. Jelinek. "Statistical Methods for Speech Recognition" *MIT press*, 1998.