Towards a Platform-Independent Cooperative Human Robot Interaction System: III. An Architecture for Learning and Executing Actions and Shared Plans

Stephane Lallée, Ugo Pattacini, Séverin Lemaignan, Alexander Lenz, Chris Melhuish, Lorenzo Natale, Sergey Skachek, Katharina Hamann, Jasmin Steinwender, Emrah Akin Sisbot, Giorgio Metta, Tony Pipe, Rachid Alami, Matthieu Warnier, Julien Guitton, Felix Warneken, Peter Ford Dominey

***Abstract*—Robots should be capable of interacting in a cooperative and adaptive manner with their human counterparts in open-ended tasks that can change in real-time. An important aspect of the robot behavior will be the ability to acquire new knowledge of the cooperative tasks by observing and interacting with humans. The current research addresses this challenge. We present results from a cooperative human-robot interaction system that has been specifically developed for portability between different humanoid platforms, by abstraction layers at the perceptual and motor interfaces. In the perceptual domain, the resulting system is demonstrated to learn to recognize objects and to recognize actions as sequences of perceptual primitives, and to transfer this learning, and recognition, between different robotic platforms. For execution, composite actions and plans are shown to be learnt on one robot and executed successfully on a different one. Most importantly, the system provides the ability to link actions into shared plans, that form the basis of human-robot cooperation, applying principles from human cognitive development to the domain of robot cognitive systems.**

I. Introduction

Cooperation is a hallmark of human cognition. Early in their development, human children begin to engage in cooperative activities with other people. Critically, from early on, children are able to cooperate in novel situations, based upon social-cognitive capacities such as representing other people's intentions, visual perspective-taking, and imitation [1, 2]. Crucially, this requires extensive learning in the early years of life. The premise of our research is that similar skills are required also for human-robot cooperation. Specifically, we derive the fundamental skills which enable young children to learn to engage in cooperative activities, and then implement these in an integrated system capable of running on several robotic platforms to study human-robot interactions. The current research builds on our previous work on action perception [3] and execution [4] and extends in the context of shared plan learning, and the exchange of acquired knowledge between different versions of the iCub [5] and the BERT2 [6] robot platforms via the internet.

Our research thus focuses on learning required for cooperation. These methods are all based on a platform independent architecture that allows the knowledge generated by one robot to be used by another one. The targeted skills all involve the concept of Action and they range from action perception and recognition to action execution, sequencing and shared planning. All these implemented skills have been isolated and studied in children, and they are building blocks for higher level cooperative abilities and cognition in general.

During the first years of life, children acquire these skills and use them in order to gain experience about the world and interact socially with other people. This learning process often requires direct experience and feedback to find a good solution. Moreover, even if an individual has acquired the skill to solve a problem in one situation, generalizing to novel situations is often difficult and requires adaptations and often (re-)learning.

Consider an alternative approach in which multiple children had direct access to a combined representation of their distinct experiences, such that each of them could benefit from the experience of all the others. In this situation of distributed and parallel learning the children's shared development would be vastly accelerated because knowledge acquired by one individual would be accessible to all others without loss or distortion. Although human cultural transmission through observational learning and language serves this function, this propagation of knowledge is always error-prone and imperfect so that the knowledge content might be lost or distorted during the learning process [7]. This idea of a shared cognitive system for multiple bodies is still "science fiction" if we speak about biological

This work was fully supported by European FP7 ICT project (CHRIS) www.chrisfp7.eu. Stephane Lallee, and Peter Ford Dominey are with the Stem Cell & Brain Research Institute, INSERM U846, Bron, France. (peter.dominey@inserm.fr). Severin Lemaignan, Emrah Akin Sisbot and Rachid Alami are with LAAS, CNRS, Toulouse, France. Alexander Lenz, Chris Melhuish and Sergey Skachek are with BRL, Bristol, United Kingdom. Ugo Pattacini, Lorenzo Natale and Giorgio Metta are with IIT, Genoa, Italy. Jasmin Steinwender and Katharina Hamann are at thewith Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany. Felix Warneken is atwith Harvard University, Cambridge, USA.

systems; however it need not be the case with robots. Robots can be distributed throughout the world, they can have different bodies, experience different situations but, in contrast to humans, there is no reason why they cannot share their experience directly. This requires an important distinction in the context of embodiment [8]. Knowing how to grasp, or walk is clearly platform specific and cannot be transferred between distinct platforms. A higher level plan, e.g. “walk to the table and grasp the bottle”, can abstract over these platform specific aspects, and can effectively be shared by physically distinct robots that can walk and grasp. Through the internet they can provide knowledge to others about things they have learned, and they can also benefit from others’ knowledge.

In this context, a central idea in the current research is to provide skills and learning mechanisms that are independent of the robot platform used. In this way, robots can communicate the results of their learning over the internet so that other robots will not need to repeat this learning process themselves. In this context, we present learning methods for action recognition, composite action sequencing and shared planning. For each of these skills, we demonstrate that the knowledge generated can be used by other robots to bypass their learning phase.

II. Context: human / robot cooperation

We first identify a set of requirements that the system should meet based in part on an analysis of human cognitive development.

A. *Cooperation requirements*

Studies of human infants [2, 9, 10] indicate that recognizing actions and engaging in cooperative interactions develops over the second and third year of life. From around 14-18 months of age, infants begin to engage in novel cooperative tasks with adults, in which they have to collaborate jointly to achieve a shared goal (such as one agent holding something in place so that another agent can manipulate the object). It has been argued that from this early age, infants are already able to represent a shared plan of action (an action plan encompassing both the child's and the partner's actions taken to bring about a certain change in the world), and are able to reverse complementary roles if necessary. In other words, infants are taking a 'bird's eye view' on the social situation, representing not only their own actions, but both their own and the partner's actions as part of a shared plan [11].

Such a shared plan allows children to demonstrate “role reversal”, where they can take on the role of either partner in a cooperative activity. We have recently implemented this type of shared planning in robotic systems which can observe actions, attribute roles, and then use the resulting shared plan to perform the cooperative task, taking the role of either one of the two participants [12, 13].

This basic representational capacity appears to be in place in human development very early on. However, during development, children become increasingly skilled in coordinating their actions with different social partners. They start to cooperate successfully with more competent adults early in the second year of life, and gradually become able to cooperate with peers also around 2 years of age [10]. Importantly, cooperating in fairly simple novel situations does not require extensive learning [2]**.** However, in more challenging tasks, with complementary actions that require a multi-step sequence and a goal that is not transparent, direct instruction appears to be necessary [14]. Thus, we have used spoken language in human-robot cooperation in order to make the nature of the tasks explicit, so that they can be used by the robot to learn the structure of the task [15, 16]. Spoken language will be central to the interactions by allowing the user to manage the interaction, and to guide the learning of new actions and shared plans (plans that include the coordinated actions of both agents toward their shared goal), as specified below.

A crucial aspect of this human cooperative behavior is the ability to perceive and analyze new actions in real time, during the course of observation of an ongoing cooperation. Children can be exposed to novel physical devices and, within a few trials of observation, learn new actions involved in manipulating these devices [1, 2].

B. *Extracting Meaning from Perception*

Robots will have to demonstrate similar learning capabilities in order to face novel situations they will encounter in the real world. Exhaustive knowledge about the world cannot be provided *a priori* by the programmer, thus the robots need the ability to learn. An important aspect of human social life is our ability to learn from others through observation and instruction [17], which is a faster and more accurate way of acquiring knowledge about complex entities than individual learning, such as trial-and-error learning.

Mandler [18] suggested that the infant begins to construct meaning from the scene, based on the extraction of perceptual primitives. From simple representations such as contact, support and attachment [19] the infant could construct progressively more elaborate representations of visuo-spatial meaning. In this context, the physical event "collision" can be derived from the perceptual primitive "contact". Kotovsky & Baillargeon [20] observed that at 6 months, infants demonstrate sensitivity to the parameters of objects involved in a collision, and the resulting effect on the collision, suggesting indeed that infants can represent contact as an event predicate with agent and patient arguments. Siskind [21] demonstrated that force dynamic primitives of contact, support and attachment can be extracted from video event sequences and used to recognize events including pick-up, put-down, and stack, based on their characterization in an event logic. Related results have been achieved by Steels and Baillie [22]. The use of these intermediate representations renders the systems robust to variability in motion and view parameters. We have used a related approach to categorize movements including touch, push, give, take and take-from in the context of linking these action representations to language [23].

In the current research, we extend these ideas, so that arbitrary novel actions including *cover*, *uncover*, *take*, *put* and *touch* can be learned in real-time with a few examples each, based on invariant sequences of primitive events specific to each action. We subsequently demonstrate that, using the same architecture, such actions can be learned on a different robot platform using an entirely different perceptual system. Finally, and perhaps most interestingly, we demonstrate that knowledge of action recognition learned on one of the robots transfers directly for successful use on another.

C. *Composing New Actions*

In addition to their perception of action, children begin to demonstrate the ability to compose new actions before two years. Indeed, they show this ability to sequence hierarchical elements in several domains including spoken language development, tool use and action [24]. Composite actions can be represented as a hierarchical organization of simpler actions with the leaves of the tree being called “atomic actions”. This is concordant with our approach to action recognition and has been used successfully employed in the action framework [25-28]. A robot with a rich set of atomic actions is useful in many aspects. However, to produce interesting behaviors we need to compose these actions in more evolved sequences. The first level is to name and create composite actions that the robot will be able to produce by executing the underlying sequence of atomic actions.

Initially, the robot is only able to perform the actions defined by the Motor Command Interface (defined below). This defined interface represents the atomic action pool that the user can use to build up composite actions. Using speech, the user is able to instruct the robot about the list of sub actions to execute and to store the definition of this new action in the Action Definitions database. From this point on, the user will be able to use the new action in more evolved sequences. We used this capacity in our experiments by teaching the action “*Put x on y*” which is composed of “*grasp x, , release y*”, and is then used as part of a shared plan. This ability is the basis of the spoken language programming framework and, as we previously demonstrated [16, 29, 30], allows the interaction between the user and the robot to be faster and more efficient.

D. *Using Shared Plans*

Developing the idea of higher level manipulation of the action concept, the next step is to extend the robot’s abilities, so that it will be able to learn and use shared plans in the context of cooperation. One important aspect of cooperative activities is division of labor and the assignment of who performs which role in the joint activity. Human children at 14-24 months display a remarkable ability to observe adults perform a cooperative task (with only one or two demonstrations) and then to engage themselves in that task, taking the role of either of the demonstrating adults [2, 9]. The behavioral data indicate that the children have understood the task in terms of a coordinated succession of actions, rather than a set of specific motor trajectories. They have a common shared action plan for the joint enterprise. (These provide a “bird’s eye view” of the collaboration and can be demonstrated by the agents’ ability to reverse roles.)

We have previously developed in [13] the ability of the iCub to learn and reproduce shared plans, including a role reversal capability. In the current work we allow the new complex action definition to be inserted into a learned shared plan structure, and we are making their implementation platform independent. Different robots can thus share the same representation of shared plans, exchange them, and eventually use them together.

III. The CHRIS Architecture

Given these requirements, we can now begin to specify the corresponding architecture. In order to be platform-independent, a cognitive architecture should abstract away from platform-specific representations of perception and action at the lowest level possible. An overview of our architecture in this context is presented in Figure 1.

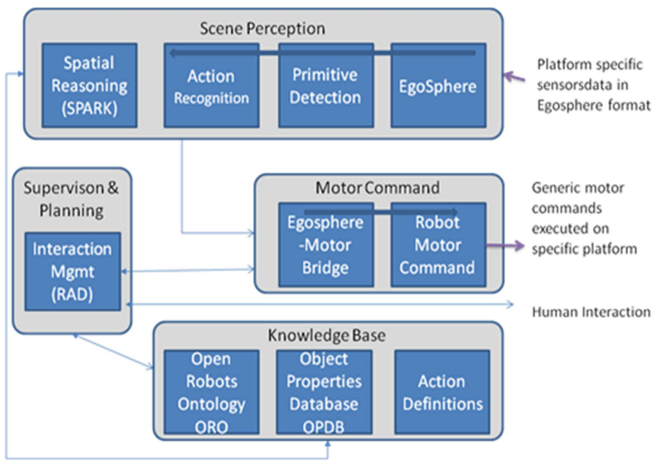

Figure 1: Overview of the Software Architecture. Communication between modules is achieved via YARP (see below). All modules presented here are completely platform independent.

Robot specific components for the 3D-perception and Motor Command levels illustrated on the right are isolated from the rest of the system at the lowest level. The architecture can be most clearly described in terms of information flow for perception and action, respectively, and their interaction. In the Perceptual flow, robot specific sensor data enters Egosphere in standardized, real-time, position and orientation format. Primitives (moving, in contact etc) are extracted by Primitive Detection, and used by Action Recognition. In this context, spatial representations are communicated to and organized in SPARK, which then makes high level spatial information available to the rest of the system via ORO. In the Action flow – the executability for planned or requested actions (issued from the user via Interaction Management, or through Planning) are verified a query of Knowledge Base: Action Definitions to verify the actions are defined, and Knowledge Base : ORO to verify visibility and preconditions. If the execution conditions are met, a request is made to Motor Command where KnowledgeBase:OPDB allows access to position information to translate name based request to robot implementation with spatial coordinates. In case of visually guided actions, the Vision-Motor bridge manages the transformation from object-related labels to the spatial coordinates of those objects in conjunction with OPDB. The following sections describe the architecture in detail.

A. Scene Perception

1) EgoSphere

The first layer of abstraction between the sensory perception system and the higher level cognitive architecture and motor control elements is formed by the EgoSphere. Unlike the sensory ego-sphere (SES) by Peters [31] which implements short term memory, associations, direction of attention in addition to localization, our simpler implementation solely acts as a fast, dynamic, asynchronous storage of object positions and orientations. The object positions are stored in spherical coordinates (radius, azimuth and elevation) and the object orientation is stored as rotations of the object reference frame about the three axes (x,y,z) of a right-handed Cartesian world-frame system. The origin of the world frame can be chosen arbitrarily and, for our experimental work, we located it at the centre of the robot's base-frame. Other stored object properties are a visibility flag and the objectID. The objectID is a unique identifier of an object which acts as a shared key across several databases (described in more detail in B below).

The robot-specific 3D perception system adds objects to the EgoSphere when they are first perceived, and maintains position, orientation or visibility of these objects over time. Modules (e.g. Primitive Detection in Fig 1) requiring spatial information about objects in the scene can query the EgoSphere. No assumptions are made about the nature of an object and any further information (e.g. object name, object type) will have to be queried from the Knowledge Base using the objectID. This architecture makes the EgoSphere particularly useful for storing multi-modal information.

The EgoSphere is implemented in C++ as a client-server system using the YARP infrastructure. Software modules requiring access to the EgoSphere include a client class which provides methods like addObject(.), setObject(.), getObject(.) or getNumberOfObjects(.), etc. Clearly, at the current state, the EgoSphere is merely a convenient abstraction layer. With increasing complexity of human-robot interaction tasks during the course of our research, we plan to add further complexity (human focus of attention, confidence, timeliness etc.), whilst preserving modularity.

2) Primitive Detection

Based on the EgoSphere representation, the robot should be able to recognize actions performed by other agents in order to learn, to cooperate or for safety reasons. We have previously demonstrated in [23] that actions involving change of possession can be described in term of perceptual primitives such as *contact*. Here we extend the primitives to include motion and visibility. Thus, an action such as "Larry takes the ball" can be characterized in terms of a sequence of perceptual primitives:

- *Motion: Larry's hand starts to move*
- *Contact: There is a physical contact between Larry's hand and the ball*
- *Motion: Both Larry's hand and the ball start to move together, then they both stop.*

We refer to these low level events as Perceptual Primitives. Dominey & Boucher [23] demonstrated that a variety of actions can be recognized with the primitive *contact(x,y)*. Here we extend this approach by including, in addition, the primitives *visible(x)* and *moving(x)*. These primitives, and their corresponding arguments and truth values, are computed in the Primitive Detection module, which polls the EgoSphere for changes in position and visibility. Contact is recognized by a minimum distance threshold which is determined empirically. Likewise, motion is detected when the position of an object changes over an empirically determined threshold. Visibility is directly available from the EgoSphere.

3) Action Recognition

It is clear from above that, when a physical action occurs, values encoding object positions in the EgoSphere change accordingly. Primitive Detection transforms this position information into sequences of perceptual primitives. Action Recognition reads this stream of perceptual primitives and groups the elements into candidate actions. Based on empirical measures, we determined that primitives which are separated by less than one second belong to a common action. In other words, a primitive sequence for an action may last several seconds, but no successive primitives are separated by more than 1 second. This limitation on fast successive actions is considered in the discussion section. When an action is performed and processed, its primitive sequence is thus segmented by the Action Recognition module, which tries to recognize it.

The Action Recognition module generates and manipulates the Action Definitions database of primitive sequences as follows. It tries to match the current sequence by an exhaustive search through the database. If the sequence is not recognized, the Action Recognition module triggers the Interaction Management to ask the user for a description of the action, providing the action name, agent and object of the action. It then associates this description with the recorded sequence for future recognition. If the sequence is recognized, the Spoken Language Interface extracts the action and arguments, and reports this to the user. The system thus provides object independent action recognition (i.e. if it has learned "Larry takes the ball", it is able to recognize “Robert takes the coffee-cup”). The module also detects, and stores within an action definition, the initial state of the objects linked to the action, and the consequences of this action on the world (e.g. if Larry covers the ball with a box, then the ball will not be visible anymore); this allows creation of new inference rules within the ORO (Open Robots Ontology server, specified below) module of the Knowledge Base, described below.

4) SPARK

SPARK (for SPAtial Reasoning Knowledge) module is in charge of generating symbolic knowledge from the geometry of the world. The module, which is linked tightly to perception (EgoSphere), builds and maintains a complete 3D model of the environment containing objects, the robot itself, and humans. With a coherent 3D world representation, SPARK assesses the current state of the world, including visual perspective taking, and generates facts in terms of visibility and readability of objects from the robot's and human's points of view, as well as geometric relations between objects such as "in", "on", "next to".

These facts are computed on-line and sent automatically to ORO for maintenance and further inference. Once the facts are stored in ORO, they are available to the rest of the system (under request or as events) via the Knowledge Base.

B. Knowledge Base

Through interaction with the user and the physical world, the system acquires new knowledge, and it is also initialized with certain background knowledge.

1) Object Properties Database

The OPDB is the common namespace manager for objects that can be perceived by the system. It contains physical parameters of objects, including their perceptual signature, as defined by the EgoSphere. Each object that is known to the system (that can be perceived and represented in the EgoSphere) has a unique identifier (the objectID) which serves as an index into the OPDB and the Knowledge Base in general.

2) The Open Robot Ontology

ORO (the "OpenRobot Ontology" server) is the semantic layer of the system. It has been designed to integrate easily in different robotic architectures by ensuring a limited set of architectural requirements. ORO is built around a socket-based server that stores, manages, processes and exposes knowledge. ORO is portable (written in Java), and can be easily extended with plug-ins, making it suitable to new applications. In the frame of the CHRIS project, a YARP bridge has been added, thus exposing the ORO Remote Procedure Call methods in a network-transparent way.

ORO relies internally on the OWL ontology dialect to store knowledge as RDF triples. It uses the open-source Jena[2] RDF graph library for storage and manipulation of statements and the equally open-source Pellet[3] first order logic reasoner to classify/apply rules and compute inferences on the knowledge base.

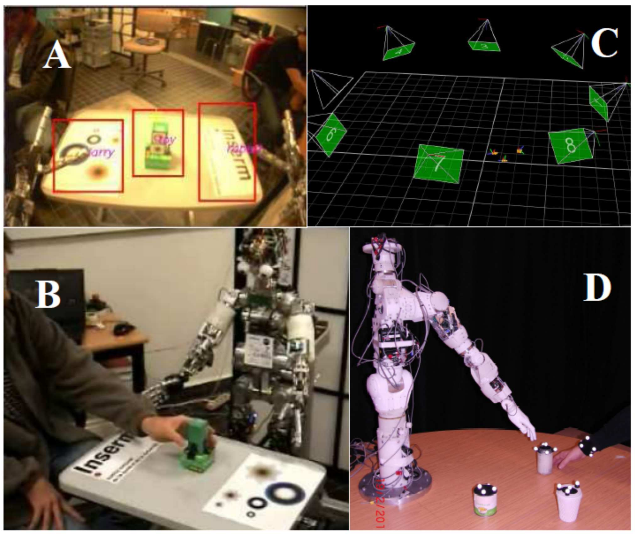

Figure 2: Specific Robotic Platforms. A. Vision processing using Spikenet™ with the video image from the iCubLyon01 robot, pictured in B. C. The Vicon™ configuration for visual perception with the BERT2 Robot, pictured in D.

In addition to storing and reasoning about knowledge, ORO offers several useful features for human-robot interaction: event registration (e.g. "Tell me when any kind of tableware appears on the table."), categorization capabilities, independent cognitive models for each agent the robot knows and different profiles of memory (short-term, episodic, long-term).

The server loads an initial ontology at startup, the so-called OpenRobots Common-Sense Ontology. This initial ontology contains a set of concepts (over 400 in the last version), relationships between concepts and rules that define the cultural background of the robot, i.e. the concepts the robot knows a priori. This common-sense knowledge is very focused on the requirement of our scenarios, namely, human-robot interaction with some well-known everyday objects (cups, cans, etc.). It also contains broader concepts such as agents, objects, and location. The common-sense ontology relies heavily on the de-facto standard OpenCyc upper-ontology for the naming of concepts, thus ensuring a good compatibility with other knowledge sources (including Internet-based ones, like WordNet[4] or DBPedia[5]).

The ontology then dynamically evolves as the robot acquires new facts; provided from one of two sources. One is the EgoSphere, via the primitive detection module, which provides the list of known objects, and asserts object relationships like: “robot sees the object or not”, “object is moving or not”, “object is touching another object or not”. The other source of new facts is the Interaction Management

[2] http://jena.sourceforge.net

[3] http//clarkparsia.com/pellet
[4] http://wordnet.princeton.edu
[5] http://dbpedia.or

human-robot interface we have built with the CSLU Toolkit [32] (described below).

3) *Action Definitions*

Actions are defined in terms of perception and execution. For perception, actions that have been learned are stored in the Action Definitions Database. Actions are defined in terms of three types of information. The Enabling State defines the state of the objects involved in the action before the action takes place. The Primitive Sequence is the time ordered set of primitive events that make up the dynamic component of the action. Finally, the Resulting State is the (potentially) new state of affairs after the action is completed. The action recognition capability described above relies primarily on the Primitive Sequence for action recognition. For execution, composite actions are defined by the user, through spoken language, in terms of the sequence of primitive actions in the learned sequential order.

C. *Motor Command*

Analogous to the Egosphere, which is the abstraction layer for Perception, we need the same mechanism for dealing with motor actions in a platform independent manner. The module that handles this task is called Robot Motor Command. A second module, called Egosphere Motor Bridge, coordinates communication between Interaction Management, Egosphere and Motor Command.

1) *Robot Motor Command*

We have identified an initial pool of atomic actions required for the robot to be able to participate in rich human-robot cooperative activities. The actions identified were the following: *Grasp, Release, Point, Reach, Orient.*

A C++ library has been devised to provide the higher level modules a suitable abstraction layer to cope with the need of generating, independently from the specific platform, the movements required by these atomic actions. The underlying idea is that any kind of goal-oriented action can be effectively described by a collection of primitives expressed both in the operational and configuration space of the robot along with the time order according to which they have to be executed. Actions such as *Orient, Reach, Point* are actually concerned with movements of the robot end-effector in the task space, whereas *Grasp* and *Release* are a combination of the *Reach* action followed by proper activation of the end-effector (fingers or pincer) in the joint space. To this end, the library exposes a set of C++ methods that enable the caller to access the internal timeline (the so called *action queue* depicted in Figure 3) by sequencing the action in terms of its primitives, without reference to the motion control details. This is the way the atomic actions have been specified, and the same approach can be easily generalized to define more sophisticated tasks starting from *Reach, Grasp, Release* as building blocks.

In order to address the requirement of being independent from any specific robot in the CHRIS community (or potentially any other robot using a suitable YARP interface), the library implementation relies on the bio-inspired Cartesian controller based on the *iKin* framework [33] that, in turn, allows generation of human-like movements of the end-effector in the task space given the kinematic description of the robot's structure.

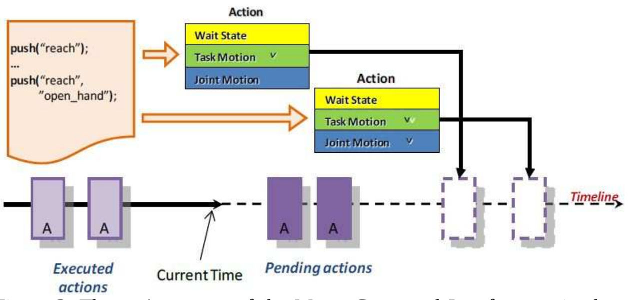

Figure 3. The *action queue* of the Motor Command Interface: notice how the action executions called within the code are defined in terms of primitives with respect to the internal timeline.

2) *EgoSphere Motor Bridge*

The Robot Motor Command capability requires information about the localization of the object being manipulated. The role of the Egosphere Motor Bridge is to translate high level commands that include the object identification, e.g. "grasp the toy". In this example, it would query the Egosphere to get information about localization and orientation of the "toy", and then propagate it to the Robot Motor Command. It is also responsible for checking certain validity aspects of the required action: if the targeted object is not in the world, then an error code will be returned to Supervision and Planning. The module also maintains a representation of the objects held by the robot, e.g. it keeps in memory which hand is holding which object. It uses this representation in order to choose which hand to use in case of a "grasp" or a "release" action (e.g. "Release the toy on the left" requires the robot to know which hand is holding the toy). This functionality is currently being tested only on humanoid robots, so the word "hand" was used, but indeed it could be generalized to any kind of manipulator.

Error codes generated by the Egosphere Motor Bridge are then handled by the Interaction Management to translate them into proper speech. The robot will say "Sorry, I cannot grasp the toy because I don't know where it is." or "From what I know none of my hands is holding the toy, I cannot release it". Speech thus provides a mechanism for pertinent state information to be made available to the user.

D. *Supervision and Planning*

1) *Interaction Management*

Interaction Management is provided by the CSLU Toolkit [32] Rapid Application Development (RAD) state-based dialog system which combines state-of-the-art speech synthesis (Festival) and recognition (Sphinx-II recognizer) in a GUI programming environment. RAD allows scripting in the TCL language and permits easy and direct binding to the YARP domain, so that all access from the Interaction

Management function with other modules in the architecture is via YARP.

Our system is state based, with the user indicating the nature of the current task, e.g., including whether the user wants to interact in the context of object recognition, action recognition or cooperative interaction tasks. In each of these sub domains, the user can then indicate readiness to show the robot a new example (object, action perception or execution, or cooperative shared plan), and the robot will attempt to recognize or learn what is shown. Interaction management also allows the system to indicate error states to the user, thus allowing the user to explore alternative possibilities.

2) Planning

The core aspect of planning is the capability to learn and execute shared plans. As defined above, a shared plan is a sequence of actions, with each action attributed to one of two agents. Shared plans can be learned via two mechanisms. The first involves perceptual action recognition: the robot observes two agents perform a cooperative task, and decomposes the perceptual sequence into a discrete sequence of action-agent components [12, 13]. The second method involves a form of spoken language programming, in which the user verbally describes the succession of action-agent components that make up the shared plan. We have also used a mixed method that involves visual demonstration and spoken language for assignment of roles [12, 13]. The robot can then use the resulting shared plan to take the role of either agent, thus demonstrating the crucial role-reversal capability that is the signature of shared planning [1].

E. YARP

Software modules in the architecture are interconnected using YARP [34], an open source library written to support software development in robotics. In brief, YARP provides an intercommunication layer that allows processes running on different machines to exchange data. Data travels through named connection points called ports. Communication is platform and transport independent: processes are not aware of the details of the underlying operating system or protocol and can be relocated at will across the available machines on a network. More importantly, since connections are established at runtime, it is easy to dynamically modify how data travels across processes, as well as addition of new modules or removal of existing ones.

Interface between modules is specified in terms of YARP ports (i.e. port names) and the type of data these ports receive or send (respectively for input or output ports). This *modular* approach allows minimizing the dependency between algorithm and the underlying hardware/robot; different hardware devices become interchangeable as long as they export the same interface.

Finally, YARP is written in C++, so it is normally used as a library in C++ code. However, any application that has a TCP/IP interface can talk to YARP modules using a standard data format. Within the CHRIS project this turned out to be of fundamental importance as it allowed us to ‘glue’ together different applications (e.g. the RAD toolkit, the ORO server or the VICON system) into a single integrated, working system.

F. Internet Interaction Mechanism

The exchange of knowledge over the internet occurs at two levels.

1) Software Versioning

The cognitive architecture software is available to all the CHRIS project partners over our subversion repository. This has allowed fast and painless integration of new components, and remote collaboration of developers.

2) Internet Knowledge Exchange

The robot knowledge is also stored on the SVN repository. All the information about actions, plans, and even some platform specific perception patterns are available. While allowing the robot to update itself to the latest version of the shared knowledge base, it also permits the human and robot to contribute to the extension of this knowledge. Since all these definitions are stored in text or xml files, the versioning system used allows merging different knowledge bases. While, at the moment, the resolution of conflicts is done by the user, we could consider interfacing this process with speech, so that the user could be free of the keyboard. Part of the overall objective of this research is to provide a basis for increasingly technically naïve users to cooperate with these robots. The speech based interaction contributes to this objective.

IV. INTEGRATION PLATFORMS

The CHRIS Software Architecture has been successfully tested on three different robotics platforms illustrated in Figures 2 and 4.

A. Platform iCubLyon01(in Lyon)

1) Robot Platform

The iCub [5] is an open-source robotic platform shaped as a three and a half year-old child (about 104cm tall), with 53 degrees of freedom distributed on the head, arms, hands and legs. The current work was performed on the iCubLyon01 at the INSERM laboratory in Lyon, France. The head has 6 degrees of freedom (roll, pan and tilt in the neck, tilt and independent pan in the eyes). Three degrees of freedom are allocated to the waist, and 6 to each leg (three, one and two respectively for the hip, knee and ankle). The arms have 7 degrees of freedom, three in the shoulder, one in the elbow and three in the wrist. The iCub has been specifically designed to study manipulation, for this reason the number of degrees of freedom of the hands has been maximized with respect to the constraint of the small size. The hands of the iCub have five fingers and 19 joints. All the code and documentation is provided open source by the RobotCub Consortium, together with the hardware documentation and CAD drawings. The robot hardware is based on high-performance electric motors controlled by DSP-based custom electronics. From the sensory point of view, the robot is equipped with cameras, microphones, a gyroscope, position sensors in all joints, and force/torque sensors in each limb.

2) 3D Spatial-Temporal Object Perception

The iCubLyon01 platform employs vision based perception operating on the image streams from the robot's cameras. Objects are recognized based on detection of predefined object templates using the commercial system Spikenet [35]. It uses a spiking neural network technology to provide fast recognition of objects in an image. Under the assumption that the robot is manipulating objects on a flat table, we can use an orthographic projection to estimate the Cartesian coordinates of the objects and feed the EgoSphere. To do so, a simple wrapper around the Spikenet API is used for retrieving the camera images, processing them with Spikenet, and broadcasting the results over the network via YARP. Another module is then used to read this data, filter the noise and update the EgoSphere appropriately. Once in the EgoSphere, the spatial-temporal object information is platform-independent.

B. Platform iCubGenova01(in Genoa)

1) Robot Platform

Compared to the iCubLyon01 the iCub robot at the Italian Institute of Technology in Genoa is equipped with additional sensory capabilities such as the measurement of forces and torques applied at each limb [36], and sensorized skin. Such an upgraded version of the iCub allows better adaptation and control of the interaction with the environment. The capacitive touch sensors composing the skin have been mounted on the iCubGenova01 only recently, and therefore are not used for the experiments described in this paper. On the other hand, the force/torque limb sensors have been exploited to detect external forces during reaching or object grasping. In fact, unless a perfect tuning of cameras parameters is performed, a procedure that usually requires a considerable effort, the vision system returns an inaccurate estimation of the object position. This affects the accuracy with which the arm is controlled and might cause an object grasp to fail. To compensate for this, the robot moves the hand to touch the objects, but it immediately stops the movement if an external contact is detected (i.e. the force applied by the arm exceeds a given threshold). This strategy has been demonstrated to be effective, as it significantly increased the likelihood of successful grasps.

Furthermore, the robot can enter a special 'coaching phase' during verbal interaction. In this situation the arm is compliant and the operator can 'teach' the robot the precise location of the hand to grasp a certain object by manually guiding the end-effector. In this way the iCub builds a map that stores the correct positioning of the hand with respect to the object, thus achieving reliable object grasping.

Finally, the implementation details concerning force control on the iCubGenova01 have been hidden inside the Motor Command Interface so they can be handled transparently by higher level modules. For instance a call to the *grasp*(.) method produces two different behaviors depending on the platform on which it is executed.

2) 3D Spatial-Temporal Object Perception

The perception system of these two versions of the iCub is the same: purely vision based. However, it is interesting to test it in different environments. Moreover, the model file used to recognize different objects is shared over the SVN repository, so that the iCubLyon01 and iCubGenova01 can both use it and contribute to it.

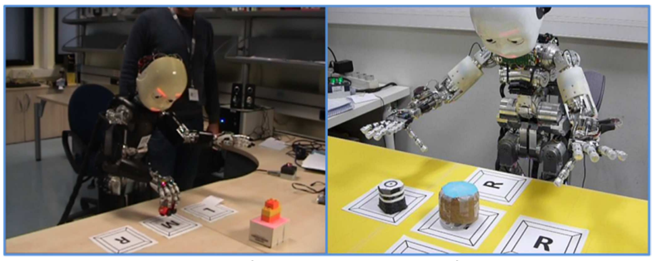

Figure 4 : iCubGenova01 (left, resorting to external forces measurements) and iCubLyon01 (right, pure position control).

C. Platform BERT2 BRL

1) Robot Platform

BERT2 (Bristol-Elumotion-Robot-Torso-2) [6] is an upper-body humanoid robot that was designed, and is currently still under construction, at Bristol Robotics Laboratory in close co-operation with their mechanical engineering partner Elumotion[6]. The torso comprises four joints (hip rotation, hip flexion, neck rotation and neck flexion). The hip rotation forms the most proximal joint to the rigid mounting base. Each arm is equipped with seven degrees-of-freedom. The shoulder flexion joint forms the mounting point of the arm to the torso and the wrist flexion joint is the most distal joint of the arm. The wrist provides a mounting interface for a sophisticated humanoid hand or a simple gripper. Each of these 18 joints is actuated by a brushless DC motor via a Harmonic Drive (TM) gear box. Low level motor control is achieved through EPOS motor controllers from Maxon Motor.

One of the main motivations that guided the design of BERT2 was the suitability to interact with humans safely and naturally using Expressive Face and Gaze Tracking. One important non-verbal communication channel we have focused on is facial expression with a particular emphasis on gaze, as used in human-human interaction [37].

2) 3D Spatial-Temporal Object Perception

The BERT2 platform uses the VICON motion capture system (with 8 stationary IR cameras) and light reflective markers arranged into unique patterns, to distinguish between scene objects and to detect their position and

[6] www.elumotion.com

orientation in 3D space. This provides reliable and robust 360 degree scene perception. The human interacting with the robot also wears a garment equipped with markers, thus body positions and postures are also available to the robot.

There are several layers of abstraction in BERT2 VICON perception. At the lowest level there is VICON hardware and software together with VICON object and actor model templates, which store information about the marker topology of the objects to be captured. The VICON software broadcasts this captured data on the network, using TCP/IP. This data is picked up by the module “ViconLink”, which is the first layer of abstraction of the VICON perception sub-system. The main purpose of “ViconLink” is to create an easily reconfigurable data bridge between the VICON software and the YARP framework.

The next layer of abstraction is the “Object Provider” module which is largely based on YARP classes. Its main purpose is to update the EgoSphere module with the most recent object positions and also to prevent the false updates when objects are not moving, even though the values are changing due to the noise in the VICON data. To do so, during initialization, the module estimates the levels of noise and calculates the noise thresholds. In its main loop "ObjectProvider" constantly polls data from "ViconLink", filters out the noise, and monitors the translation and rotation quantities for all objects. It triggers the EgoSphere update as soon as translation or rotation exceeds some pre-set minimum value for any of the objects. In this way "ObjectProvider" updates EgoSphere as soon as minimal prescribed movement occurs, but does not allow false triggering. Again, once in the EgoSphere, the spatial-temporal object information is platform-independent.

V. Experiments

Multiple experiments have been performed in a distributed manner on the three platforms. Experiments A and B are related to perception and were performed on iCubLyon01 and BERT2. Experiments C and D are related to motor execution, and were performed on iCubLyon01 and iCubGenova01. Again, the two iCubs used are different at the motor control level and in this respect we consider them as two different platforms for the motor command implementation.

A. *Object learning*

The goal of the experiment is to allow the user to teach the system the names and properties of new objects. In these experiments, two sets of objects have been pre-specified respectively for each of the two 3D perception systems. This corresponds to visual templates for Spikenet on the iCub, and reflective marker topologies for VICON on BERT2. Initially, the objects can thus be recognized and tracked, but they have no associated semantics.

In the experiment, the human moves an object to indicate the focus of attention to the robot, and then the robot asks for the name and the type of the object. Learning the object’s type (i.e. “cup”) links its semantics to the other concepts the robot already knows, including initial commonsense knowledge from ORO.

When an object moves, the platform specific perception systems identify and accurately localize the object. The respective object perception module then updates the EgoSphere in real time. At this point, we are entering the platform-independent CHRIS architecture. The Primitive Detection module regularly polls the EgoSphere for visibility and object coordinates, and sends extracted primitives to other interested modules. In this case, it sends to ORO a notification when an object starts or stops moving. In parallel, the Interaction Management system handles the verbal human-robot interaction. It queries ORO to know which objects are currently moving and if the names of these objects are known. If they are unknown then it asks the human for more information, as described in the dialog below.

- [Robot] Initializing... *about 5 sec* ...What's next?
- [Human] m*oves an object*
- [Robot] *does not know the object.* What is it called?
- [Human] coffee cup
- [Robot] Did you say coffee cup?
- [Human] Yes
- [Robot] Ok. Now I know. What kind of object is coffee cup?
- [Human] A cup
- [Robot] Did you say cup?
- [Human] Yes
- [Robot] So coffee cup is a cup. What's next?

During this interaction, two new statements are added to the ORO database. The first associates the objectID for the recognized object with the label “coffee cup”. The second associates this objectID with the type “cup” which allows the system to exploit inheritance of knowledge about cups from the preloaded ontology.

The robot is then able to use this additional knowledge in order to reason and describe objects. The next time a human shows the robot a coffee cup the robot will say “this is a coffee cup”. In addition, we could ask to the robot “could you show me a cup?” to have the robot point to the coffee cup or to a tea cup, exploiting the type information. This naming experiment was successfully completed for multiple objects on the iCub and BERT2 platforms.

B. *Action Perception Learning*

As described above, an important skill in cooperation is the robot’s ability to understand its partner’s actions, or the actions of two humans cooperating while the robot learns by observing. Figure 2 illustrates the BERT2 and iCubLyon01 platforms used in this experiment.

1) *Learning to Perceive New Actions*

In the “watch action” mode, the robot tries to recognize and describe the actions it sees. It can then store the recognized actions in a plan that can be executed later, or it can ask for more information, if the observed action is not known or uncertain.

Regarding the flow of information through the system, it is the same as in the object naming experiment, but it now includes the Action Recognition module. Under control of Interaction Management, Action Recognition receives primitive events from Primitive Detection. It compares the segmented primitive event sequences with known actions in the Action Database and provides, in real time, actions that are recognized. Alternatively, if an action has been segmented but not recognized, it reports a 'special event'. These recognitions are then processed by Interaction Management, which manages the situation either by announcing the name and arguments of the recognized action, or prompting the user to describe the action and its arguments, as illustrated in this dialog excerpt.

- [Robot] Initializing... *about 5 sec* ...What's next?
- [Human] Watch my actions
- [Robot] Ok. Show me something
- [Human] *puts a coffee cup on the table*
- [Robot] Need to learn this one.
- [Human] John put the coffee cup
- [Robot] Ok. Show me something
- [Human] *removes the coffee cup from the table*
- [Robot] Need to learn this one.
- [Human] John takes the coffee cup
- [Robot] Ok. Show me something
- [Human] *put a book on the table*
- [Robot] I saw John put the book

At this point the robot is able to recognize the two simple actions *put* and *take* independently of the object targeted by the action. In this scenario a single demonstration was sufficient. In practice the robot may need to see the same action several times before being able to recognize it. Lallee et al [38] preformed extensive testing of this system on the iCubLyon01 platform. In over 100 action presentations, with the actions *cover, uncover, put, take* and *touch*, on average the system required less than three examples to correctly learn a given action so that it could subsequently be recognized without error.

The crucial experiment here involved performing the same action learning tests on the BERT2 platform, where visual perception based on pattern matching with Spikenet would be replaced by reflective marker tracking provided by VICON. We tested BERT2 with the actions *put*, *take*, and *touch*. These actions were successfully learned, and generalized to new objects. This indicates that by abstracting 3D spatial-temporal information in the EgoSphere, the CHRIS architecture is indeed platform-independent.

The next component of this experiment replies to the question "can knowledge about the spatial-temporal characteristics of an action learned on one platform be used for action recognition on another?"

2) Knowledge transmission between Robots

Following an interaction session with humans, the robot Knowledge Base acquires new knowledge (of object and action definitions) through learning. This acquired knowledge is stored prior to system shutdown and reloaded at subsequent system startup, thus allowing progressive accumulation of experience over extended time. In the current experiment, the Action Recognition database that was generated while actions were being learned on BERT2 was transferred via internet to the iCubLyon01 file system, and loaded at startup on the iCubLyon01. We then tested the Action Recognition capability, by performing *put* and *take* actions.

In a set of 20 trials (10 each for *put* and *take*) we observed an overall recognition accuracy of 85%. The errors were due to noise in the vision system which produces false indications of motion (see discussion). Importantly, the iCubLyon01 was able to recognize actions that had been learned on BERT2, thus exploiting the experience of a different robot via internet knowledge transfer.

Action recognition learning can be seen in this video: http://youtu.be/NDgBcDi3axw

C. Action Execution Learning

The complement to 'learning to *perceive* new actions' is 'learning to *perform* new actions'. The Motor Command interface provides a set of low level actions that can be used to instruct the robot about what to do. However, using actions at this level is not very convenient to produce complex behaviors. The robot needs to build new skills on the top of these basic motor commands.

1) Learning to Perform New Actions

The learning process for new actions takes place in a natural way for the user. When being instructed to do a task, the robot is commanded by the user. If an instruction is not known by the robot, then it will ask for explanation from the user. At this point, the user has to instruct the robot about which commands compose the unknown instruction. After this learning process, the new instruction is added to the known commands and the interaction can go on.

This type of learning can occur at different levels, a composite action can be made only of the simplest motor commands, but it can also be made of other composite actions. This allows the actions known by the robot to become richer and richer, and the learning process to be facilitated by the possibility to use already complex actions. Here is a dialog occurring during the learning of a new action:

- [Robot] Tell me what to do.
- [Human] Put the toy on the left.
- [Robot] Sorry, I don't know what "put" means. Explain it to me.
- [Human] Grasp the toy.
- [Robot] Grasping… *Robot grasps the toy*
- [Human] Release toy on left.

- [Robot] Releasing… *Robot releases the toy on the left marker*
- [Human] It is finished.
- [Robot] Ok, I know what "put" means. What should I do now?

We performed this experiment on the iCubLyon01, and demonstrated that the put action definition could be generalized to different objects and different locations.

2) Transmission of Action Performance Knowledge Between Robots

The list of all actions known and their definitions is stored in a file that is loaded at startup. The user can ask the robot to update its knowledge base from the SVN repository, thus loading the new actions that the current user, or previous/remote users, taught it. In our case described here, we thus transferred the representation of the "put" action via the internet from Lyon to Genoa, and this definition was used by the iCubGenoa01. We demonstrated that this action definition, which had been learned in Lyon, was successfully used by the iCubGenoa01 in Genoa. Figure 4 illustrates the two platforms.

D. Shared Plan Learning

Shared planning represents the current summit of our robot cognition capability. The current experiment tests the ability to learn and use a shared plan on one platform, and then to verify that the resulting shared plan can be used remotely on a different platform.

1) Learning to Perform New Shared Plans

There are several ways to teach a new shared plan. We have previously demonstrated the system's ability to learn shared plans by observing two humans perform the cooperative action. Here we demonstrate a second method in which the user instructs the robot via spoken interaction, in order to build the list of *action(agent,arguments)* that will then be executed by the robot and its human partner. Here is an example of learning a simple cooperative game using speech interaction. The game involves retrieving a toy that is covered by a box.

- [Robot] Action or shared plan?
- [Human] Learn a Shared plan
- [Robot] Ok what is the first action?
- [Human] Robert grasps the box.
- [Robot] And then?
- [Human] Larry puts the toy on the left.
- [Robot] And then?
- [Human] Robert releases the box on middle.
- [Robot] And then?
- [Human] It is finished.

This spoken interaction results in the following plan to be created and stored:

robert grasp box
larry put toy left
robert release box middle

Such plans can then be parsed and used in execution mode to send actions sequentially to the EgosphereMotorBridge. In this context, Interaction Management allows the user and robot to determine together who goes first in the cooperation, and then manages the smooth unfolding of the cooperation.

While this learning method is convenient, human beings demonstrate another capability: they can watch other humans performing a cooperative task and extract their shared plan by observation. The action recognition system of the architecture has been shown to provide this ability [13]. The robot can be asked to watch two humans executing a shared plan. Each action they perform is recognized by the robot; the agent and the action performed are associated, and integrated in the new shared plan.

After a shared plan has been learned, the robot can execute it. It prompts the user for who will be the agent starting the first action (robot or user) and then proceeds, step by step, to the execution of the plan. The iCubLyon01 was thus able to perform a shared plan with a human, and to demonstrate role reversal.

Execution of the shared plan, with the learned action "put" can be seen here:
http://www.youtube.com/watch?feature=player_detailpage&v=kqWm8LCTxPs
(paste link into browser)

2) Transmission of Shared Plan Knowledge between Robots

The principle for sharing knowledge about plans between the robots is the same as for composite actions. Since these definitions are stored in a text file on the SVN repository, it is possible either to update the robot or to contribute to the shared knowledge.

The shared plan that originated in Lyon was thus transferred via the internet to Genoa where it was used by the Supervision and Planning component of the architecture. The shared plan was successfully executed on the iCubGenoa01 in order to allow the robot to perform the cooperative task with a human.

Importantly, this shared plan required both the shared planning capability, and the use of a learned composite action within the shared plan.

Remote execution of the shared plan, can be seen here:
http://www.youtube.com/watch?v=qe0QuPaNDDc&feature=player_embedded
(paste link into browser)

In summary, the Interaction Management capability which includes the spoken language dialog management implements four types of learning dialogs:

A.Object learning: allows the user to associate a name with an object in the EgoSphere that has not yet been named.

B. Action perception learning: allows the user to associate a name and arguments with a learned sequence of perceptual primitives. This has been demonstrated with the following actions: Cover(x,y), uncover(x,y), put(x,y), take(x) and touch(x).

C. Action execution learning: allows learning composite actions by composing primitives (grasp, release, reach). Learned actions include: Cover(x,y) {grasp(x), release(y)}; Uncover(x,y) {grasp(x)}; Put(x, y){grasp(x), release(y)}.

D. Shared plan learning: allows the linking of actions and agents into an articulated plan. An example is for the shared plan to "uncover the toy" which is decomposed into: robert grasp box, larry put toy left, robert release box middle.

VI. Discussion

We present an architecture that exploits the idea of abstracting the cognitive architecture from the robot specific body and sensors. Specifically, this abstraction takes place in two domains. In the perceptual domain, all perceptual information is encoded in the EgoSphere so that independent of the sensor (e.g. stereo vision, motion capture) the final 3D coordinate representation is the same. Likewise, in the motor domain, a set of baseline actions is defined in Robot Motor Command (including grasp(x), release(x); reach(x)), that can be performed on different robots, but with the same name, argument structure and effects. It should be noted that the cognitive function of the robot can still be considered embodied as the architecture acquires all its information from interaction between the robot and the world, via the low level abstraction of the EgoSphere. Thanks to this abstraction, we were able to provide different robots with the same high level capabilities for perception and action in the context of learning new cooperative shared plans, and to share this knowledge over the internet between different robots.

A. Limitations and future development:

The work described here emphasizes abstraction at the sensory motor level by requiring a common format for spatial input to the system from diverse sensors, and providing this same format to the modules responsible of motor commands implementation. This provides a capability consistent with that described by Demiris & Johnson [39] where action execution and performance can mutually benefit from shared representations.

Action Recognition provides real-time formation and recognition of sequential patterns of primitive events (motion, visibility and contact) specific to different actions. It is thus sensitive to noise in the 3D perception sensors; we are currently rendering this approach more robust. This includes the use of a probabilistic approach for matching the segmented primitive event sequences with the learned actions, optimization of spatio-temporal filtering to reduce false motion from visual jitter, and inclusion of the initial-to-final state transitions as additional components in definition of an action. Likewise, in the current version, successive actions (e.g. taking an object, then putting it at a new location) should be separated by at least one second, so that the system can automatically distinguish and segment the perceptual primitive sequences. This situation is consistent with our current constraint, i.e., that when demonstrating action, users show actions one after another, then wait to see if the robot recognizes, before proceeding. Future work will address more fluent action sequences in the context of learning from demonstration [40].

The speech that we have used here is relatively primitive and sometimes ungrammatical. We have previously explored the more extensive possibilities of relating the argument structure of grammatical sentences to the argument structure of actions in terms of execution [16, 29, 30]. We are now extending these approaches to action observation and description with the use of more appropriate grammar.

The current method for sharing the different level of knowledge of the robots over the internet is still very basic, and probably not easy to carry out for a naïve user. However, we have already made a step towards a more human friendly interface, by allowing the most standard operations to be carried out using the speech interface. Apart from improving the sharing mechanism, we will also extend the type of content which is shared: robot vocabulary, platform specific parameters and even real time information (which robot is on, where, what does it see) could be shared by the robot community, so as to allow us to proceed to more evolved interactions. An example of the benefit of such real time sharing, is illustrated by the following example: Larry asks iCub if it sees the toy, iCub doesn't, but BERT2 which is in another room does. iCub could then use the perceptual information acquired by BERT2 to answer the user about the toy location.

B. Conclusions

While robotic platforms are becoming increasingly complex, the development of cognitive systems can be advanced by the definition of more standard ways to access the sensory-motor layer. Our system independent architecture contributes to the deployment of cognitive abilities on diverse robot platforms that can interface with the abstraction layer defined by the EgoSphere and the Motor Command Interface. We believe that the continued development of increasingly well defined and standard interfaces between robot platforms and cognitive system can accelerate the development of robot intelligence, and we are taking a first step in that direction.

In doing so, we have also taken our first steps towards the idea of having different learning machines (the robots individuals) updating and sharing a common global knowledge base, thus leveraging experience from multiple sources [22]. High bandwidth internet could be the support for much advanced sharing of knowledge: while we are storing at the moment only "static" information, the same idea could be extended to a more dynamic and real time system. For example, one could imagine a shared online egosphere database that robots in different locations would update and use in real time. This would transform our actual shared memory system into a global perception-action system, distributed over the sensory-motor network of robots. Possibilities of such a global entity are endless, however, developing and maintaining such a system are challenges for the upcoming years.

VII. Acknowledgment

This research was supported by the European Commission under the Robotics and Cognitive Systems, ICT Project CHRIS (FP7-215805).

VIII. References

[1] M. Tomasello, M. Carpenter, J. Call, T. Behne, and H. Moll, "Understanding and sharing intentions: The origins of cultural cognition," *Behavioral and Brain Sciences*, vol. 28, pp. 675-691, 2005.

[2] F. Warneken, F. Chen, and M. Tomasello, "Cooperative activities in young children and chimpanzees," *Child Development*, vol. 77, pp. 640-663, 2006.

[3] S. Lallée, S. Lemaignan, A. Lenz, C. Melhuish, L. Natale, S. Skachek, T. van Der Zant, F. Warneken, and P. Dominey, "Towards a Platform-Independent Cooperative Human-Robot Interaction System: I. Perception," in *IROS*, Taipei, 2010, pp. 4444 - 4451

[4] S. Lallée, U. Pattacini, J. Boucher, S. Lemaignan, A. Lenz, C. Melhuish, L. Natale, S. Skachek, K. Hamann, J. Steinwender, E. A. Sisbot, G. Metta, R. Alami, M. Warnier, J. Guitton, F. Warneken, and P. F. Dominey, "Towards a Platform-Independent Cooperative Human-Robot Interaction System: II. Perception, Execution and Imitation of Goal Directed Actions," in *IROS*, San Francisco, 2011, pp. 2895 - 2902.

[5] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The iCub humanoid robot: an open platform for research in embodied cognition," in *PerMIS: Performance Metrics for Intelligent Systems Workshop*, Washington DC, USA, 2008, pp. 19-21.

[6] S. S. Lenz A, Hamann K, Steinwender J, Pipe A G, Melhuish C R,, *The BERT2 infrastructure: An integrated system for the study of human-robot interaction.*, 2010.

[7] D. Sperber and L. A. Hirschfeld "The cognitive foundations of cultural stability," *Trends in Cognitive Sciences*, vol. 8, p. 6, 2004.

[8] T. Froese and T. Ziemke, "Enactive artificial intelligence: investigating the systemic organization of life and mind. ," *Artificial Intelligence*, vol. 173:, pp. 466-500, 2009

[9] F. Warneken and M. Tomasello, "Helping and cooperation at 14 months of age," *Infancy*, vol. 11, pp. 271-294, 2007.

[10] C. Brownell, G. Ramani, and S. Zerwas, "Becoming a social partner with peers: Cooperation and social understanding in one-and two-year-olds," *Child Development*, vol. 77, pp. 803-821, 2006.

[11] M. Carpenter, M. Tomasello, and T. Striano, "Role reversal imitation and language in typically developing infants and children with autism," *Infancy*, vol. 8, pp. 253-278, 2005.

[12] P. Dominey and F. Warneken, "The basis of shared intentions in human and robot cognition," *New Ideas in Psychology*, p. (in press), 2009.

[13] S. Lallée, F. Warneken, and P. Dominey, "Learning to collaborate by observation," in *Epirob*, Venice, 2009.

[14] J. Ashley and M. Tomasello, "Cooperative problem-solving and teaching in preschoolers," *Social Development*, vol. 7, pp. 143-163, 1998.

[15] P. Dominey, G. Metta, F. Nori, and L. Natale, "Anticipation and initiative in human-humanoid interaction," in *International Conference on Humanoid Robotics*, 2008.

[16] P. Dominey, A. Mallet, and E. Yoshida, "Real-time cooperative behavior acquisition by a humanoid apprentice," in *International Conference on Humanoid Robotics*, Pittsburg, Pennsylvania, 2007.

[17] M. Tomasello, *The cultural origins of human cognition*: Harvard University Press Cambridge, MA, 1999.

[18] J. Mandler, Ed., *Preverbal representation and language* (Language and space. MIT Press, 1996, p.^pp. Pages.

[19] L. Talmy, "Force dynamics in language and cognition," *Cognitive science*, vol. 12, pp. 49-100, 1988.

[20] L. Kotovsky and R. Baillargeon, "The development of calibration-based reasoning about collision events in young infants," *Cognition*, vol. 67, pp. 311-351, 1998.

[21] J. Siskind, "Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic," *Journal of Artificial Intelligence Research*, vol. 15, pp. 31-90, 2001.

[22] L. Steels and J. Baillie, "Shared grounding of event descriptions by autonomous robots," *Robotics and Autonomous Systems*, vol. 43, pp. 163-173, 2003.

[23] P. Dominey and J. Boucher, "Learning to talk about events from narrated video in a construction grammar framework," *Artificial Intelligence*, vol. 167, pp. 31-61, 2005.

[24] P. Greenfield, "Language, tools and brain: The ontogeny and phylogeny of hierarchically organized sequential behavior," *Behavioral and Brain Sciences*, vol. 14, pp. 531-551, 1991.

[25] M. Ryoo and J. Aggarwal, "Recognition of composite human activities through context-free grammar based representation," 2006, pp. 1709-1718.

[26] J. Allen and G. Ferguson, "Actions and events in interval temporal logic," *Journal of logic and computation*, vol. 4, p. 531, 1994.

[27] S. Park and J. Aggarwal, "A hierarchical bayesian network for event recognition of human actions and interactions," *Multimedia Systems*, vol. 10, pp. 164-179, 2004.

[28] S. Park and J. Aggarwal, "Semantic-level understanding of human actions and interactions using event hierarchy," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04*, 2004, p. 12.

[29] P. Dominey, A. Mallet, and E. Yoshida, "Progress in programming the hrp-2 humanoid using spoken language," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2169-2174.

[30] P. Dominey, A. Mallet, and E. Yoshida, "Real-Time spoken-language programming for cooperative interaction with a humanoid apprentice," *Intl J. Humanoids Robotics*, vol. 6, pp. 147-171, 2009.

[31] I. R. A. Peters, K. A. Hambuchen, and R. E. Bodenheimer, "The sensory ego-sphere: a mediating interface between sensors and cognition," *Auton. Robots*, vol. 26, pp. 1-19, 2009.

[32] S. Sutton, R. Cole, J. Villiers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, B. Rundle, and K. Shobaki, "Universal speech tools: The CSLU toolkit," in *Fifth International Conference on Spoken Language Processing*, 1998.

[33] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An Experimental Evaluation of a Novel Minimum-Jerk Cartesian Controller for Humanoid Robots," in *IROS*, Taipei, 2010.

[34] P. Fitzpatrick, G. Metta, and L. Natale, "Towards Long-Lived Robot Genes," *Robotics and Autonomous Systems*, vol. 56, pp. 29-45, 2007.

[35] S. Thorpe, R. Guyonneau, N. Guilbaud, J. Allegraud, and R. VanRullen, "SpikeNet: Real-time visual processing with one spike per neuron," *Neurocomputing*, vol. 58, pp. 857-864, 2004.

[36] M. Fumagalli, M. Randazzo, F. Nori, L. Natale, G. Metta, and G. Sandini, "Exploiting Proximal F/T Measurements for the iCub Active Compliance," in *IROS*, Taipei, 2010.

[37] A. Senju and G. Csibra, "Gaze following in human infants depends on communicative signals," *Current Biology*, vol. 18, pp. 668-671, 2008.

[38] S. Lallée, C. Madden, M. Hoen, and P. Dominey, "Linking language with embodied teleological representations of action for humanoid cognition," *Frontiers in Neurobotics*, 2010.

[39] Y. Demiris and M. Johnson, "Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning," *Connection Science*, vol. 15, pp. 231-243, 2003.

[40] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, pp. 469-483, 2009.