The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Towards a Platform-Independent Cooperative Human-Robot Interaction System: I. Perception

Stephane Lallée, Séverin Lemaignan, Alexander Lenz, Chris Melhuish, Lorenzo Natale, Sergey Skachek, Tijn van Der Zant, Felix Warneken, Peter Ford Dominey

Abstract— **One of the long term objectives of robotics and artificial cognitive systems is that robots will increasingly be capable of interacting in a cooperative and adaptive manner with their human counterparts in open-ended tasks that can change in real-time. In such situations, an important aspect of the robot behavior will be the ability to acquire new knowledge of the cooperative tasks by observing humans. At least two significant challenges can be identified in this context. The first challenge concerns development of methods to allow the characterization of human actions such that robotic systems can observe and learn new actions, and more complex behaviors made up of those actions. The second challenge is associated with the immense heterogeneity and diversity of robots and their perceptual and motor systems. The associated question is whether the identified methods for action perception can be generalized across the different perceptual systems inherent to distinct robot platforms. The current research addresses these two challenges. We present results from a cooperative human-robot interaction system that has been specifically developed for portability between different humanoid platforms. Within this architecture, the physical details of the perceptual system (e.g. video camera vs IR video with reflecting markers) are encapsulated at the lowest level. Actions are then automatically characterized in terms of perceptual primitives related to *motion*, *contact* and *visibility*. The resulting system is demonstrated to perform robust object and action learning and recognition on two distinct robotic platforms. Perhaps most interestingly, we demonstrate that knowledge acquired about action recognition with one robot can be directly imported and successfully used on a second distinct robot platform for action recognition. This will have interesting implications for the accumulation of shared knowledge between distinct heterogeneous robotic systems.**

Manuscript received March 10, 2010. This work was fully supported by European FP7 ICT project CHRIS).

Stephane Lallee, Tijn van Der Zant and Peter Ford Dominey are with the Stem Cell & Brain Research Institute, INSERM U846, Bron, France. (stephane.lallee@inserm.fr; robotijn@gmail.com; peter.dominey@inserm.fr).

Severin Lemaignan is with LAAS, CNRS, Toulouse, France. (severin.lemaignan@laas.fr) Alexander Lenz, Chris Melhuish and Sergey Skachek are with BRL, Bristol, United Kingdoms. (alex.lenz@brl.ac.uk ; Chris.Melhuish@brl.ac.uk; Sergey.Skachek@brl.ac.uk).

Lorenzo Natale is with IIT, Genoa, Italy. (lorenzo.natale@iit.it).

Felix Warneken is with Harvard University, Cambridge, USA (warneken@wjh.harvard.edu)

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]. The premise of our research is that similar skills are required also for human-robot cooperation. Specifically, in the CHRIS project[1], we derive the fundamental skills which enable young children to engage in cooperative activities and implement these in an integrated system capable of running on several robotic platforms to study human-robot interactions. The current research reports on this integrated system and resulting experiments with iCub [3] and the BERT2 robot platforms.

The novelty of the current research is twofold: First, we present an on-line learning method for recognition of simple human action related to object manipulation. Some research has already been done in the area of action learning and recognition by robots [4-7], however our approach is based on detection of simple perceptual primitives that can be processed independently from the perceptual system used. Second, we demonstrate that this platform-independent architecture operates successfully on two very distinct physical robot platforms, using highly distinct perceptual systems. Finally we demonstrate that because of the perceptual abstraction in the architecture, knowledge acquired about recognizable actions on one robot can be used to recognize actions (with a completely different perceptual system) on a different robot.

II. Context: Human / Robot Cooperation

A. Cooperation requirements

Studies of human infants [2, 8, 9] show that recognizing actions is a task that gradually 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

[1] www.chrisfp7.eu

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 terms, 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 [10]. Such a shared plan allows the child to demonstrate "role reversal" where she 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 could 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[11, 12]. This basic representational capacity appears to be in place in human development very early on. However, over 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 becoming able to cooperate also with peers around 2 years of age [9]. Importantly, cooperating in fairly simple novel situations does not require extensive learning [2]**.** In more challenging tasks with complementary actions that require a multi-step sequence and a goal that is not transparent, direct instructions appears to be necessary [13]. 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 [14, 15] A crucial aspect of this human cooperative behavior is the ability to observe and understand 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 trails 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 an ability to learn. An important aspect of human social life is our ability to learn from others through observation and instruction [16], which is a faster and more accurate way of acquiring knowledge about complex entities than individual learning, such as trial-and-error learning. Mandler [17] 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 [18] the infant could construct progressively more elaborate representations of visuospatial meaning. In this context, the physical event "collision" can be derived from the perceptual primitive "contact". Kotovsky & Baillargeon [19] 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 [20] 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 [21]. 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 link these action representations to language [22]. 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 the other.

III. THE CHRIS ARCHITECTURE

In order to be platform-independent, a cognitive architecture should abstract away from platform-specific representations at the lowest level possible. An overview of our architecture in this context is presented in Figure1.

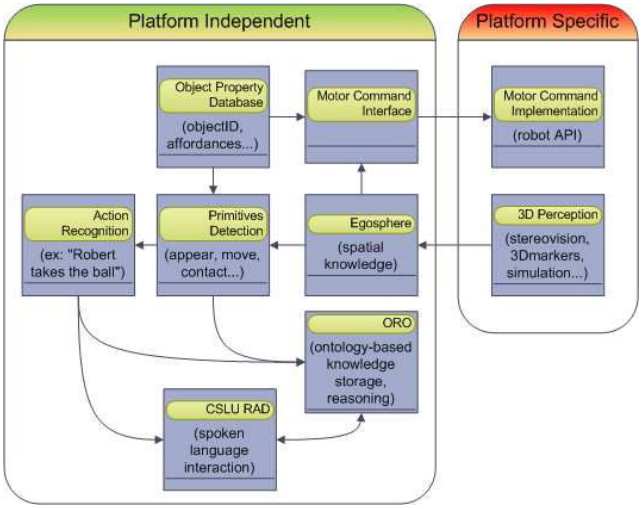

Figure 1: Overview of the Software Architecture. Each block is a stand alone software module that execute a certain function. The interface between the platform specific software and the generic architecture is the Egosphere, which allow abstraction from low level perception. Arrows represent the flow of information (data, commands), which are transported over the network via YARP. All the left part is robot independent and been tested on the iCub and BERT robots.

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.

A. *Scene Perception*

1) *EgoSphere*

The first layer of abstraction between the sensory perception systems, the higher level cognitive architecture and motor control elements is formed by the EgoSphere. Unlike the sensory ego-sphere (SES) by Peters [23] 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*

The robot should be able to recognize actions performed by other agents in order to learn, to cooperate or for safety reasons. A few systems are performing action learning and recognition [4-7, 24, 25], however none of them is completely platform independent. Since our system is taking inputs from the Egosphere, it allows applying learning and recognition algorithm that are not at all related to a specific robot. Moreover, our algorithm is using a novel approach : we have previously demonstrated [22] that actions involving change of possession could 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 and then they both stop.*

We refer to these low level events as Perceptual Primitives. Dominey & Boucher [22] demonstrated that a variety of actions could be recognized with the single 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*

Thus, 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 sec. belong to a common action. A

primitive sequence for an action may last several seconds, but no successive primitives are separated by more than 1 sec. 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 Spoken Language Interaction 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. 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 concerned by the action, and the consequences of this action on the world (i.e. if Larry covers the ball with a box, then the ball will not be visible anymore) which will allow creation of new inference rules within the ORO module of the Knowledge Base, described below.

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 architecture 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 RPC methods in a network-transparent way. ORO relies internally on the OWL ontologies 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.

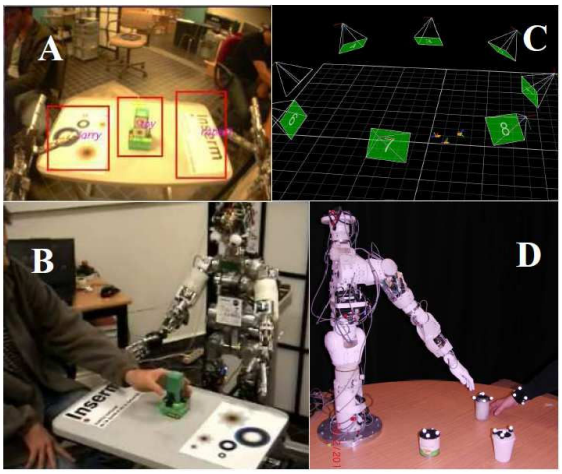

Figure 2: Specific Robotic Platforms. A. Vision processing using Spikenet™ with the video image from the iCubLyon01 robot, pictured in B. C. The Vicon™ configuration for visual perception with the Bert Robot, pictured in D.

Besides simply storing and reasoning about knowledge, ORO offers several useful features for human-robot interaction: events 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 defines 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 contains as well broader concepts like agents, objects, location, etc. 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: these are provided either from the EgoSphere via the primitive detection module, or via spoken language interaction with human.

3) Action Definitions

Actions that have been learned are stored in the Action Definitions Database. Actions are defined in terms of three

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

[3] http//clarkparsia.com/pellet

[4] http://wordnet.princeton.edu

[5] http://dbpedia.or

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.

C. Spoken Language Interaction

The spoken language interaction is provided by the CSLU Toolkit [26] 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. Our system is thus state based, with the user indicating the nature of the current task (including whether he wants interact in the context of object recognition, action recognition or action sequence recognition tasks). In each of these subdomains, the user can then indicate that he is ready to show the robot a new example (object, action or action sequence) and the robot will attempt to recognize or learn what is shown. RAD scripts are in done in TCL which allows communication of speech data to other modules through YARP.

D. YARP

Software modules in the architecture are interconnected using YARP [27], 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 the network. More importantly, since connections are established at runtime it is easy to dynamically modify how data travels across processes, add new modules or remove 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 to "glue" together different applications (e.g. the RAD toolkit, the ORO server or the VICON system) into a single integrated, working system.

IV. Integration Platforms

The CHRIS Software Architecture has been successfully tested on two different platforms illustrated in Figure 2.

A. Platform iCubLyon01

1) Robot Platform

The iCub [3] is an open-source robotic platform shaped as three and a half year-old child (about 104cm tall), with 53 degrees-of-freedom (DOF) distributed on the head, arms, hands and legs. The current work was performed on the iCubLyon01 at the INSERM laboratory in Lyon, France. The DOF are distributed over the full body: 6 for the head, 3 for the waist, 6 in each leg and 7 for each arm. The iCub has been specifically designed to study manipulation, for this reason the number of DOF 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 a DSP-based custom electronics. From the sensory point of view the robot is equipped with cameras, microphones, gyroscope, position sensors in all joints, 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 stereo cameras. Objects are recognized based on detection of predefined object templates using the commercial system Spikenet [28]. It uses a spiking neural network technology to provide fast recognition of objects in an image. By doing this with the two stereo cameras of the robot, we can 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 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 BERT2 BRL

1) Robot Platform

BERT2 (Bristol-Elumotion-Robot-Torso-2) is an upper-body humanoid robot designed, and 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). Each arm is equipped with 7 DOF. 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. One of the

[6] www.elumotion.com

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 [29].

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 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 an easily reconfigurable data bridge between the VICON software and the YARP framework. The next layer of abstraction is the "Object Provider" module. Its main purpose is to update the EgoSphere with the most recent object positions and to filter the noise in the VICON data. Again, once in the EgoSphere, the spatial-temporal object information is platform-independent.

V. EXPERIMENTS

Diverse experiments have been performed in a distributed manner on the two platforms. The first goal of these experiments is to show the portability of the full cognitive system between multiple robots, more than giving precise benchmarks of the skills provided by the system. The experiments reported on here are those which were run on the iCub and BERT2 platforms using the identical CHRIS architecture (see accompanying video).

A. Object learning

The goal of the experiment is to allow the user to teach the system the name 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 to the robot, which then 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 Spoken Language Interaction system manages 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 to the human for more information as described in the dialog below.

- [Robot] Initializing... *about 5 sec* ...What's next?
- [Human] *moves 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 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 to 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 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. 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 the Spoken Language Interface, 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 or a special event if an action has been segmented but not recognized. These recognitions are then processed by Spoken Language Interaction 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] What happened?
- [Human] John puts the coffee cup
- [Robot] Ok. Show me something
- [Human] *removes the coffee cup from the table*
- [Robot] What happened?
- [Human] John takes the coffee cup
- [Robot] Ok. Show me something
- [Human] *put a book on the table*
- [Robot] I saw John puts 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 [30] 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 BERT platform, where visual perception based on pattern matching with Spikenet would be replaced by reflective marker tracking provided by VICON. We tested BERT 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. Our final 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?"

C. *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, we took the Action Recognition database that was generated while actions were being learned on BERT, and loaded it 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 iCub was able to recognize actions that had been learned on BERT, thus exploiting the experience of a different robot.

VI. DISCUSSION

We present an architecture that exploits the idea of abstracting the cognitive architecture from the robot specific body and sensors. 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 to different robots the same high level capabilities for perception and reasoning, and to share knowledge acquired via different sensors.

A. *Limitations and future development:*

The work described here emphasizes abstraction at the sensory level (and does not address motor control), by requiring a common format for spatial input to the system from diverse sensors. A parallel approach is to be taken at the motor command level (Motor Command Interface, Fig 1). This is based on the definition of a set of actions including *give, take, put, point* and their arguments. Their initial and final states are defined in a platform independent manner, but the specific joint-level implementation is specified in the context of the corresponding robot platforms. This will provide a capability consistent with that described by Demiris & Johnson [31] 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 is consistent with our current constraint that when demonstrating action, users show actions one after another, and wait to see if the robot recognizes, before proceeding. Future work will address

more fluent action sequences in the context of learning from demonstration [32].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 [15, 33, 34]. We are now extending these approaches to action observation and description with the use of more appropriate grammar.

B. Conclusions

While robotic platforms are becoming increasingly complex, the development of cognitive systems can be advanced by the development 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 taking the 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 [21]. Further work will investigate methods to enhance this ability and to allow robot platforms distributed over the world to take advantage of it.

VII. Acknowledgment

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

VIII. References

[1] Tomasello, M., et al., *Understanding and sharing intentions: The origins of cultural cognition.* Behavioral and Brain Sciences, 2005. **28**(05): p. 675-691.

[2] Warneken, F., F. Chen, and M. Tomasello, *Cooperative activities in young children and chimpanzees.* Child Development, 2006. **77**(3): p. 640-663.

[3] Metta, G., et al. *The iCub humanoid robot: an open platform for research in embodied cognition*. in *PerMIS: Performance Metrics for Intelligent Systems Workshop*. 2008. Washington DC, USA.

[4] Yamato, J., J. Ohya, and K. Ishii. *Recognizing human action in time-sequential images using hidden Markov model*. in *IEEE Proc. Computer Vision and Pattern Recognition*. 1992.

[5] Johnson, M. and Y. Demiris, *Perceptual perspective taking and action recognition.* International Journal of Advanced Robotic Systems, 2005. **2**(4): p. 301-308.

[6] Bobick, A. and Y. Ivanov. *Action recognition using probabilistic parsing*. in *IEEE Proc. Computer Vision and Pattern Recognition*. 1998.

[7] Demiris, Y. and B. Khadhouri, *Hierarchical attentive multiple models for execution and recognition of actions.* Robotics and Autonomous Systems, 2006. **54**(5): p. 361-369.

[8] Warneken, F. and M. Tomasello, *Helping and cooperation at 14 months of age.* Infancy, 2007. **11**(3): p. 271-294.

[9] Brownell, C., G. Ramani, and S. Zerwas, *Becoming a social partner with peers: Cooperation and social understanding in one-and two-year-olds.* Child Development, 2006. **77**(4): p. 803-821.

[10] Carpenter, M., M. Tomasello, and T. Striano, *Role reversal imitation and language in typically developing infants and children with autism.* Infancy, 2005. **8**(3): p. 253-278.

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

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

[13] Ashley, J. and M. Tomasello, *Cooperative problem-solving and teaching in preschoolers.* Social Development, 1998. **7**(2): p. 143-163.

[14] Dominey, P., et al. *Anticipation and initiative in human-humanoid interaction*. in *International Conference on Humanoid Robotics*. 2008.

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

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

[17] Mandler, J., ed. *Preverbal representation and language*. Language and space. 1996, MIT Press. 365–384.

[18] Talmy, L., *Force dynamics in language and cognition.* Cognitive science, 1988. **12**(1): p. 49-100.

[19] Kotovsky, L. and R. Baillargeon, *The development of calibration-based reasoning about collision events in young infants.* Cognition, 1998. **67**(3): p. 311-351.

[20] Siskind, J., *Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic.* Journal of Artificial Intelligence Research, 2001. **15**(1): p. 31-90.

[21] Steels, L. and J. Baillie, *Shared grounding of event descriptions by autonomous robots.* Robotics and Autonomous Systems, 2003. **43**(2-3): p. 163-173.

[22] Dominey, P. and J. Boucher, *Learning to talk about events from narrated video in a construction grammar framework.* Artificial Intelligence, 2005. **167**(1-2): p. 31-61.

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

[24] Kaiser, M. and R. Dillmann. *Building elementary robot skills from human demonstration*. in *Proceedings of the International Conference on Robotics and Automation*. 1996.

[25] Nicolescu, M. and M. Mataric. *Natural methods for robot task learning: Instructive demonstrations, generalization and practice*. in *Proceedings of the Second International Joint*
Conference on Autonomous Agents and Multiagent Systems. 2003. Malbourne: ACM.

[26] Sutton, S., et al. *Universal speech tools: The CSLU toolkit*. in *Fifth International Conference on Spoken Language Processing*. 1998.

[27] Fitzpatrick, P., G. Metta, and L. Natale, *Towards Long-Lived Robot Genes.* Robotics and Autonomous Systems, 2007. **56**(1): p. 29-45.

[28] Thorpe, S., et al., *SpikeNet: Real-time visual processing with one spike per neuron.* Neurocomputing, 2004. **58**: p. 857-864.

[29] Senju, A. and G. Csibra, *Gaze following in human infants depends on communicative signals.* Current Biology, 2008. **18**(9): p. 668-671.

[30] Lallée, S., et al., *Linking language with embodied teleological representations of action for humanoid cognition.* Frontiers in Neurobotics (submited), 2010.

[31] Demiris, Y. and M. Johnson, *Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning.* Connection Science, 2003. **15**(4): p. 231-243.

[32] Argall, B., et al., *A survey of robot learning from demonstration.* Robotics and Autonomous Systems, 2009. **57**(5): p. 469-483.

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

[34] Dominey, P., A. Mallet, and E. Yoshida, *Real-Time spoken-language programming for cooperative interaction with a humanoid apprentice.* Intl J. Humanoids Robotics, 2009. **6**(2): p. 147-171.