Teaching Nullspace Constraints in Physical Human-Robot Interaction using Reservoir Computing

Arne Nordmann, Christian Emmerich, Stefan Ruether, Andre Lemme, Sebastian Wrede and Jochen Steil

Abstract—A major goal of current robotics research is to enable robots to become co-workers that collaborate with humans efficiently and adapt to changing environments or workflows. We present an approach utilizing the physical interaction capabilities of compliant robots with data-driven and model-free learning in a coherent system in order to make fast reconfiguration of redundant robots feasible. Users with no particular robotics knowledge can perform this task in physical interaction with the compliant robot, for example to reconfigure a work cell due to changes in the environment. For fast and efficient learning of the respective null-space constraints, a reservoir neural network is employed. It is embedded in the motion controller of the system, hence allowing for execution of arbitrary motions in task space. We describe the training, exploration and the control architecture of the systems as well as present an evaluation on the KUKA Light-Weight Robot. Our results show that the learned model solves the redundancy resolution problem under the given constraints with sufficient accuracy and generalizes to generate valid joint-space trajectories even in untrained areas of the workspace.

I. INTRODUCTION

The configuration problem of advanced robotics systems is entailed as one of the major challenges of current robotics research [1]. The potential gain in efficiency through automation technology is often reduced through high costs for the reconfiguration of robot systems due to adaptation of manufacturing processes. Configuration changes yield high costs for manual re-programming and testing of robotic systems and their accompanying software.

Future applications will employ robotic systems for tasks such as multi-part assembly and use several non-standard, e.g., redundant manipulators or other specific tools in close HRI scenarios [2] resulting in even more complex adaptation processes. While redundant manipulators provide high flexibility for the realization of complex scenarios, e.g., in car-manufacturing [3], the gained flexibility induces additional challenges such as solving inverse kinematics with redundancy resolution in joint-space for arbitrary movements.

Analytic approaches for solving the inverse kinematics under specific constraints [4], [5] usually require expert knowledge, the availability of rigorous kinematic models of the robot and tedious manual programming. In order to minimize manual programming effort, we present a programming-by-demonstration approach for redundancy resolution based on physical human-robot interaction [6] (pHRI), neural learning and a hybrid control scheme.

Fig. 1. During execution of trajectories, the FlexIRob system respects null-space constraints taught in physical human-robot interaction.

Our approach allows a single human tutor to efficiently teach a compliant robot several null-space constraints in different areas of the workspace. Users with no particular robotics knowledge can perform this task in physical interaction with the robot system in a few minutes by recording a small set of training examples. During an exploration phase, training data is recorded, which serves as input to a purely data-driven learning algorithm. A single recurrent neural network encodes the inverse kinematics mapping with null-space constraints. After training, the learned inverse kinematics controller is embedded in a hybrid control architecture, allowing for execution of arbitrary motions in task space, respecting the learned null-space constraints. The approach is developed and evaluated in a robotics system concept termed FlexIRob (Flexible Interactive Robot) using a recent version of the KUKA Light-Weight Robot [7] (LWR IV).

The remainder of the paper is structured as follows: Section II presents an overview of the interaction work-flow facilitating fast reconfiguration. Subsequently, Section III introduces the applied learning algorithm which allows the encoding of inverse kinematics mapping with null-space constraints in a single recurrent neural network. Section IV describes the hybrid robot control and software architecture while Section V presents results from task-oriented evaluation experiments. Last but not least, Section VI reviews related work on redundancy resolution, physical interaction and learning for constrained movement generation and adaptation before Section VII concludes this contribution.

II. INTERACTION MODEL

In the following, we give an overview about the fundamental modus operandi, the integration of physical human-robot interaction and the work-flow of the entire reconfiguration procedure. We point out, that the entire reconfiguration procedure can be done within few minutes (depending on the number of different areas in the workspace). It is managed...
exclusively by means of physical human-robot interaction and requires only a single human tutor.

As a first design decision, we introduce two distinct operation phases, namely an exploration and execution, to clearly structure the desired human-robot interaction. Furthermore, to control the work-flow during the operation phases and to switch the robot’s control modes, we provide an interaction controller. The trigger mechanisms of that controller completely rely on physical human-robot interaction, such as affecting the robot by external forces or using the attached tool, and observed properties of the robot system itself, such as measured axis values or end-effector positions. Fig. 2 shows a UML activity diagram highlighting the high-level activities of the robot system and the human teacher in both of the two phases.

During the exploration phase parts of the task space are explored in physical interaction with a human tutor, providing the training data for the so far untrained RNN. First, the interaction controller switches the robot to gravity compensation mode, triggered by the tutor through touching the robot and thus applying external forces to it. This mode enables the tutor to move all joints easily while maneuvering the robot to the desired area and thus adapting its configuration to constraints in the environment, e.g. to avoid collisions with fixed physical obstacles by selecting a certain elbow configuration (see Fig. 1). Second, when the system recognizes that the robot is no longer moved and has reached its training position, it is switched to impedance control. In this stopped moving mode, the robot compliantly tries to stick to the current posture allowing the tutor to move it locally, i.e. within the desired area, while globally securing respecting of the previous demonstrated constraints. Hence, training data is recorded that reflects the respective redundancy resolution of the inverse kinematic in that area. This procedure can be repeated for an arbitrary number of desired workspace areas and is entirely guided by physical interaction triggers arising from physical interactions of the tutor with the robot. After recording, the training data is passed to the RNN for data-driven learning of the taught constraints. Learning usually takes less than one minute depending on the number of training areas.

Once the RNN is trained, the execution phase starts, allowing to execute a movement task specified by a human user or the task component. The task, which in the FlexIRob demo system is usually defined as a 2D trajectory on a tablet PC or by 3D trajectories between specified points in the workspace, is sent to the control service, which from now on includes the RNN, for executing the resulting joint-space trajectories.

III. LEARNING REDUNDANCY RESOLUTION

The learning framework we utilize in FlexIRob to address the aforementioned learning aspects is a Reservoir Computing approach. It consists of a recurrent neural network (RNN) architecture (cf. Fig. 3) combined with efficient supervised synaptic learning and local unsupervised adaption of the neurons’ excitability. The RNN implements a dynamical mapping between end-effector positions $u$ and joint values $q$. Learning is completely data-driven and has therefore no explicit model knowledge about the robot platform. This approach has proven to work well in the context of inverse kinematic learning in general [8] and also in the context of learning inverse kinematics from pose-constraint movements [9].

Besides the input and output layer $u$, $q$, the network architecture comprises a fixed hidden recurrent neural network layer $y$, the reservoir. We denote the network state at time step $k$ by:

$$z(k) = (u(k)^T, y(k)^T, q(k)^T)^T$$

The matrix $W^{net}$ captures all connection strength submatrices between neurons in the RNN and is defined by

$$W^{net} = \begin{pmatrix} W^{res}_{inp} & W^{res}_{res} & W^{res}_{out} \\ 0 & W^{out}_{res} & 0 \end{pmatrix},$$

where we denote by $W^{\square}_{\star}$ all connections from $\star$ to $\square$ using $inp$ for input, $out$ for output, and $res$ for inner reservoir neurons. Only connections $W^{out}_{res}$ projecting to the output
neurons are trained by error correction (illustrated by dashed arrows in Fig. 3). All other weights are initialized randomly with small weights and remain fixed. We consider recurrent network dynamics

\[
x(k+1) = (1-\Delta t) x(k) + \Delta t W^{net} z(k)
\]

(1)

\[
z(k) = f(x(k)),
\]

(2)

where for small \(\Delta t\) continuous time dynamics are approximated. However, here we consider a static mapping between input and output, which allows us to use \(\Delta t = 1\). \(z\) is obtained by applying activation functions component-wise to the neural activations \(x_i, i=1\ldots N\). We use parametrized logistic activation functions \(y_i = f(x_i, a_i, b_i) = (1 + \exp(-a_i x_i - b_i))^{-1}\) for the reservoir neurons. Input and output neurons have an identity as activation function, i.e. are linear neurons.

Subsequently, a fast learning scheme based on backpropagation-decorrelation (BPDC) configures a recurrent neural network to approximate the inverse kinematics.

\[
\Delta a_i(k) = \eta_{IP} \frac{1}{a_i(k)} + x_i(k) \Delta b_i(k).
\]

(5)

This unsupervised self-adaptation rule is local in time and space and therefore efficient to compute. In [12] IP has been applied to reservoir networks and was shown to tune the reservoir’s neurons to an optimized working regime while improving robustness of the networks’ performance.

IV. SYSTEM ARCHITECTURE

To combine the flexibility of the learned RNN controller with the precision of classical control approaches, we developed a hybrid control scheme. To facilitate the necessary teaching in physical human-robot interaction and allow for fast learning process, we integrated the necessary components in a coherent system architecture using a state-of-the-art compliant robot.

A. Robotics Platform

While most parts of the presented approach are not bound to a specific robot platform, we developed our current system using a recent version of the KUKA Light-Weight Robot [7] (LWR IV). The LWR IV is a redundant manipulator with seven joints allowing a manifold of configurations in joint-space for a single end-effector position and thus provides high flexibility for complex movements in workspace. We chose this robot for its impedance-based control scheme [13] resulting in active compliance of the manipulator which is beneficial for the desired physical human-robot interaction. Additional to the active compliance control the LWR IV system provides a gravitation compensation mode in which the robot holds its position by compensating the gravitation force, but at the same time fully complies to external forces applied to the manipulator. While physical interaction is also possible with conventional 6 DOF industrial robots if force sensors are added at the end-effector, only the redundancy of the LWR IV in combination with the force sensors in each of the joints allows intuitive teaching of null-space constraints along the entire joint configuration via physical interaction.
B. Control Architecture

In order to explain the hybrid control scheme of FlexIRob, including the use of compliance features of the robot for kinesthetic teaching and position impedance control with the learned RNN controller, we provide a systems engineering perspective to the system. Fig. 4 depicts the major functional blocks and their control flow. In its two modes, exploration and execution, the system selectively activates different control strategies as explained in the following sections.

1) Exploration: In a first phase of exploration the robot is in its gravitation compensation mode, where forces applied to the robot are not counteracted by the controller. This phase is used to position the robot manually to a desired joint angle configuration \( q_{\text{init}} \). After the configuration is reached, the robot is switched to impedance control for recording training data via kinesthetic teaching. In this phase the cartesian position control and joint position control is inactive. The only active control loop in this phase is the inner impedance control loop of the LWR IV. Reference for this control loop is the interaction force \( f_{\text{int}} \) applied by the operator, \( f_{\text{int}} \), and the chosen joint angle configuration \( q_{\text{init}} \). Parameters for the impedance controller are stiffness and damping in cartesian coordinates, which were carefully chosen to allow easy physical interaction with the robot.

During kinesthetic teaching the current joint angles \( q_{\text{cur}} \) and the corresponding cartesian end-effector positions \( x_{\text{cur}} \) are recorded and after successful training passed as training data to the RNN to enable the network to learn the taught constraints. Once learning is finished, the trained network is embedded in the hybrid control scheme explained in the following.

2) Execution: During the execution phase a 3D task-space trajectory \( x_{\text{des}} \) is provided by the user and the position controllers are active. As shown in Fig. 4 the end-effector positions \( x_{\text{des}} \) are passed to the RNN to map them to desired joint values \( q_{\text{des}} \). Note that the the RNN that was trained in the exploration phase, now serves as open-loop controller. In order to control the execution of the RNN output \( q_{\text{des}} \) and \( x_{\text{des}} \) an additional hierarchical position controller, based on ideas by Grupen and Huber[14], was integrated into the feedback control loop of the execution phase. The primary task of this hierarchical controller is the end-effector task \( x_{\text{des}} \), the secondary task is to full-fill the joint-space task \( q_{\text{des}} \), given by the RNN output. This way the fulfillment of the taught constraints and therefore the redundancy resolution is provided by the network, and the precise fulfillment of the task-space task is ensured by the combination of the hierarchical and the RNN controller.

In our experiments, the outer position control loop is executed with a 12 ms control cycle, the inner impedance control loop on the LWR IV is executed with a 1 ms control cycle. Note that the outer control loop, the cartesian position control, is an open-loop control in our current setup.

C. Software Architecture

The developed distributed system architecture provides a set of coherent tools for high-level simulation and collision detection for the KUKA Light-Weight Robot. The distributed system is designed according to information-driven design principles [15] and common interfaces for compliant robots as developed in the AMARSi EU project. The different components are designed to be reusable across a wide range of robotic setups featuring compliance features and aiming at interactive learning.

In order to send commands to the robot and receiving status updates in real-time communication we use the Open KUKA Control [16] (OpenKC) library. It provides a C wrapper around KUKA’s Fast Research Interface [17] (FRI), which enforces real-time client-server communication. The client is implemented by a KRL application running on the KUKA Robot Control (KRC) system while the server interface is part of the OpenKC library.

The hierarchical controller is implemented using the Control Basis Framework, which is an open source C++ library ¹ for realization of closed loop controllers through individual simple components.

V. EVALUATION

In order to evaluate the performance of the FlexIRob system, two questions have to be considered: (i) Does the system respect the taught null-space constraints and ii) to what extend is the task performance in cartesian space affected by those constraints, i.e. what is the task performance? Our hypothesis is (i) that the constraints are learned not only in the trained areas but also generalized in a reasonable manner in untrained areas and (ii) that due to the hierarchical controller, that prioritizes the cartesian task over the joint-space task, the task-space performance is not affected. Nevertheless, we want to point out the task-space accuracy of the entire system.

A. Evaluation Setup

While the second question (ii) can be evaluated by simply measuring the deviation of the performed movement from the desired trajectory, i.e. by calculating the euclidean distance between target points and measured actual end-effector position, for (i) we need to define an appropriate measurement. Directly calculating the “deviation from trained constraints” in joint-space is not suitable, because: (a) the desired constraints would have to expressed formally in joint-space as well as (b) a corresponding meaningful distance measurement would have to be defined, and (c) ground truth data for that kind of evaluation is not available in untrained areas.

In contrast, we introduce a more tangible measurement motivated from the viewpoint of a co-worker scenario that is motivation for the current FlexIRob setup. In that scenario a human tutor adapts the robot’s configuration to obstacles that constrain the robot’s movements. For evaluation we therefore introduce physical obstacles in the robot’s environment. Intending that the robot avoids these obstacles during execution, counting collisions between the robot and

¹Available at: https://github.com/fps/CBF
its environment is a natural measurement for evaluating the constraints in areas where training data was recorded, as well as in untrained areas. For answering questions (i) and (ii), we therefore calculate both measurements in trained and in the untrained areas of the robot’s workspace.

For that purpose, we define a set of workspace configurations, depicted in Figures 5. Each of them comprises a left and a right training area (above the blue boxes, respectively), which are placed at both ends of the robot’s evaluation workspace. Hence, each workspace configuration contains an untrained area in between where the system has to generalize the taught constraints to unseen data points (interpolation area). We defined reasonable obstacle setups that might occur in real world applications: In setup A, depicted in Fig. 5(a), the robot has to reach over an obstacle in its entire workspace. In setup B, cf. Fig. 5(b), the robot has to reach around an obstacle in the left training area and in the rest of the workspace the elbow movement is constrained in height. Setup C, cf. Fig. 5(c), constrains the height of the elbow movement on the left side and forces an "elbow up" configuration on the right side to avoid collisions with the wall. The robot has to perform a change of its elbow configuration while moving through the interpolation area. Setup D, cf. Fig. 5(d), is the most difficult of the four depicted due to its narrow obstacle setup. The robot has to reach around obstacles in both training areas and has to move the elbow to a large extend from "elbow left" to "elbow right" configuration while moving through the untrained interpolation area.

Teaching the actual constraints for these setups is done as described in Section II on the real robot system, while monitoring collisions in simulation: During the exploration phase each of the training areas is explored in a 3D helix-like trajectory, that consists of 150 up to 300 points. The recorded training data reflects the respective desired elbow configuration (e.g. "elbow up", "elbow down") determined by the obstacles in the robot’s workspace. Of course, we therefore avoided any collisions during kinesthetic teaching in all our setups. After recording, the data is used to train the RNN as described in Section III with parameters given in Table I. The parameters are based on a proposal by [9] and [8] for controlling a humanoid robot respecting specific constraints. The RNN learning is organized in epochs, whereby each epoch constitutes a sweep through the training data set while adapting the network and a subsequent sweep for online evaluation of that epoch. In order to obtain well performing networks for evaluation we conduct learning until a MSE of the network output drops below a threshold of 0.2, limiting to a maximum number of 1000 epochs if the threshold is not reached yet. The threshold was obtained manually from previous experiments and seems to constitute a reasonable lower bound.

In order to define the mentioned areas numerically for the execution phase, we segmented the workspace in three cubes: Two cubes, which we call "left and right training area" (from the user’s viewpoint), are defined by bounding boxes around each of the two training data trajectories. The third cube is defined by the area between those cubes and termed "interpolation area". In each of the setups, we generate straight 3D trajectories between 100 random points, 25 per training area and 50 for interpolation area, and evaluate the task-space accuracy as well as the number of collisions in the respective area. Note, that the evaluation is done on the real robot with the above mentioned simulation visualizing the internal model of the robot and checking for collisions with obstacles.

B. Results

First we want to point out the task-space accuracy of the entire FlexIRob system. The average error measured by euclidean norm between target end-effector positions and actually performed positions is about 2mm, cf. Table II. This error is independent of the evaluated area and even of the evaluated setup, and hence independently of the taught constraints. The relation between left and right training data, the task-space trajectories in all three areas during evaluation

![Fig. 5. Different workspace configurations used for evaluation. During exploration phase desired elbow configurations are taught to the robot through kinesthetic teaching to avoid collisions with obstacles.](image-url)
TABLE II
EXPERIMENTAL RESULTS OF THE FLEXIROB EVALUATION.

<table>
<thead>
<tr>
<th>Training</th>
<th>Evaluation</th>
<th>collisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>epochs</td>
<td>MSE</td>
</tr>
<tr>
<td>Setup A</td>
<td>177 + 195</td>
<td>202 &lt; 0.2</td>
</tr>
<tr>
<td>Setup B</td>
<td>250 + 232</td>
<td>237 &lt; 0.2</td>
</tr>
<tr>
<td>Setup C</td>
<td>125 + 155</td>
<td>275 &lt; 0.2</td>
</tr>
<tr>
<td>Setup D</td>
<td>159 + 163</td>
<td>1000 0.2017</td>
</tr>
</tbody>
</table>

Fig. 6. Illustration of training data and trajectories during evaluation in left and right training area and interpolation area for the first workspace configuration.

and corresponding target movements is depicted in Fig. 6. Note that the task-space precision is not due to the learning approach, but shows that our hybrid approach manages to show reasonable task-space performance independent from the trained constraints.

Concerning the ability to learn the null-space resolution, the results show that the taught elbow configurations are learned and securely retrievable from the RNN, at least in the training areas. While performing robot movements in the training areas no collisions were detected in all four evaluation setups, cf. Table II. Furthermore, in the setups A, B and C the redundancy resolution, and hence the taught elbow configurations, is generalized by the RNN also to the interpolation area well enough, so that no collision was recorded either. The robot's behavior in that area is represented by the center robot arm in each of the setup visualizations in the Figures 5. Interestingly, in setup D the robot collided during 16 of the performed 50 trajectories in the interpolation area.

For further discussion of these results, we have to distinguish the task difficulties of the designed workspace setups. The design of both setups A and B determines the desired elbow configurations in left and right training area to be very similar, and therefore generalization in between those areas is fairly easy. Setup C constitutes a problem, where the robot has to switch between two different elbow configurations, from "elbow down" to "elbow up" and the other way around while moving through the untrained area. It is worth noting, that the RNN takes the "right guess" in this case for how to generalize even though there was no data provided. In setup D, the task difficulty is further increased due to stronger constraints induced by the narrow obstacle setup. The robot has to change its elbow configuration from "elbow left" and "elbow right" while moving through the untrained interpolation area. As a result, during 16 of the 50 performed trajectories collisions with the environment occurred.

We argue, that learning in our approach is model-free and therefore data-driven, and thus strong posture constraints in a workspace area has to be present in the training data. As a proof of concept to that argument we repeated training and evaluation of setup D, but now using three training areas: a left, right and an additional third one in the former interpolation area where the collisions occurred. The results of this setup D* show, that the RNN benefits from the additional training area, as the number of collisions in the interpolation area was reduced from formerly 16 to 3, still by kinesthetic teaching the configuration within less than 5 minutes.

VI. RELATED WORK

From the viewpoint of efficient reconfiguration and taking on a high-level perspective, available methods on constraining the movement generation of redundant robots can be classified in either offline or online methods. Offline methods specifically enforce constraints defined in preface of system
runtime and require upfront programming or reconfiguration, e.g., by directly modifying the implementation of the inverse kinematics solution. In contrast, online methods exploit techniques such as programming-by-demonstration to learn symbolic or statistical models of generalized robot trajectories adapting to user-definable constraints at system runtime either in a specific learning phase or during motion execution.

Offline approaches include the pre-defined selection of single redundancy solutions in joint-space, the minimization of an energy function [5] or are based on linear programming methods [4]. In the latter approaches, null-space configurations are selected based on additional criteria such as "home" positions, joint angle and velocity or acceleration limits. Alternative schemes realize biomimetic models such as [18] using Bayesian modeling techniques to build an inverse kinematics solver preferring human-like joint configurations for redundancy resolution of humanoid robot arms.

In contrast to analytic approaches, an important area of work is concerned with learning inverse kinematics of redundant robots. However, only a limited number of these consider the introduction of additional constraints for redundancy resolution. An exemplary approach in this category is constrained supervised learning [19] which explicitly models configurational and temporal constraints for solving inverse kinematics problems. While this is conceptually similar, the actual constraints are not learned from training data but explicitly specified. More recently, Neumann et al. utilized a similar learning scheme on a humanoid robot as used in this contribution for pose-constraint bi-manual movements [9].

A large body of work is available which aims at online reconfiguration or learning and encoding of motion skills either at a symbolic or at trajectory level exploiting programming-by-demonstration [20] concepts. A recent example for an HMM-based motion generation approach that integrates physical human-robot interaction is presented in [21]. An advanced interaction concept allows iterative refinement of motion primitives subsequent to observational learning of a reference trajectory. Another example of motion learning and adaptation in physical HRI describes a combination of programming-by-demonstration and an adaptive control algorithm [22] facilitating adaptive impedance control of robot motions in human-robot collaboration scenarios.

Ito and Tani present a related approach based on online imitation learning [23] inspired by the idea of mirror neurons. Through observation of a human tutor movement trajectories are learned and encoded in a recurrent neural network model. A particular benefit of their method is that generation and recognition of motion patterns is supported.

While these approaches allow sophisticated teaching and adaptation of task specific robot movements, an inherent limitation is that learning and adaption is targeted to a limited number of specific movement tasks. In contrast, the constraint learning and hybrid control approach presented in this contribution is designed to be task independent and shall allow the robot to execute arbitrary trajectories respecting the trained constraints.

Compared to the initial prototype of the FlexIRob system [24], major extensions are the transition from 2D to 3D trajectories, improved combination of RNN and hierarchical controllers and eased interaction as well as a completely new evaluation setup.

VII. SYNOPSIS

In this contribution, we presented an integrated interaction and machine learning approach for re-configuration of redundant manipulators. This approach exploits the compliance of current robot manipulators for physical human-robot interaction. Configuration of the system can be achieved by users without deep robotics knowledge, using kinesthetic teaching to gather training data intrinsically containing constraints given by the environment or required by the intended task. The learning component uses a data-driven and model-free approach for training the recurrent neural net, which becomes an embedded part of a hybrid control scheme effective during execution. The applied learning rule allows to perform the training of the underlying RNN controller in usually less than one minute, which makes reconfiguration of the system fast and flexible.

As a task-oriented evaluation of our prototype system, we demonstrated its ability to execute user-generated trajectories under different constraints using a hardware-in-the-loop evaluation setup. This showed that the FlexIRob allows execution of arbitrary tasks not only in the trained areas of the workspace, but also generalizes providing reasonable intermediate solutions for untrained areas.

Further work will assess the applicability and robustness of the approach in industrial co-worker scenarios as part of a dedicated ECHORD experiment and thus extend the evaluation to analyze stability aspects, influence of training data sample rates and execution sample rates. Functionally, we will extended our system to handle arbitrary 6D-trajectories and include perceptual processing for online trajectory adaption in task-space.

Concluding, the features and the promising results of this FlexIRob prototype system present a first step towards easier reconfiguration of robotic systems through integration of learning technology, physical interaction and advanced robotics technology.

VIII. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contributions of Alicia Weirich and Michael Johannfunke as well as the reviewers’ comments. The research leading to these results has received funding from the European Community’s Seventh Framework Programme FP7/2007-2013 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No [248311]-[AMARSi].
REFERENCES


