Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Project No. 611391

DREAM
Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Grant Agreement Type:	Collaborative Project
Grant Agreement Number:	611391

D6.3.4 Deliberative Subsystem

Due date: **01/04/2018**
Submission Date: **30/03/2018**

Start date of project: **01/04/2014** Duration: **54 months**

Organisation name of lead contractor for this deliverable: **Vrije Universiteit Brussel**

Responsible Person: **Bram Vanderborght** Revision: **1.7**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Executive Summary

Deliverable D6.3 defines the specification, design, implementation and validation of the Deliberative subsystem within the cognitive architecture in WP6. Specifically, this report presents the advances in task T6.3 for the first, second, third and fourth years of the DREAM project. During the first year the cognitive architecture and the Deliberative subsystem were designed. During the second year the focus was on the autonomous acquisition of action selection through machine learning. In the third year, the design of the deliberative subsystem was improved and implemented to enable the delivery of the core robot behaviour for use in interventions. During the fourth year, the deliberative subsystem has been evaluated in interventions, successive improvements of performance have been done iteratively. The subsystem is fully deployed and running with the cognitive architecture developed by WP6 for use in interventions.

Note: this is a living document and extends on the preliminary version of this deliverable which was submitted for review last year.

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Paul Baxter, Plymouth University
Tony Belpaeme, Plymouth University
Hoang-Long Cao, Vrije Universiteit Brussel
Albert de Beir, Vrije Universiteit Brussel
Pablo Gómez, Vrije Universiteit Brussel
Daniel Hernandez Garcia, Plymouth University
James Kennedy, Plymouth University
Emmanuel Senft, Plymouth University
Greet Van de Perre, Vrije Universiteit Brussel
Bram Vanderborght, Vrije Universiteit Brussel

Revision History

Version 1.0 (P.B. 17-02-2016)
Skeleton structure and appendices.

Version 1.1 (T.B. 19-02-2016)
Contents from previous preliminary deliverable report to build on.

Version 1.2 (T.B. 21-03-2016)
Added new papers published in Period 2 and added section 4.

Version 1.3 (J.K. 30-03-2016)
Further updates for Period 2.

Version 1.4 (J.K. 13-01-2017)
Contents copied from previous preliminary deliverable to build on.

Version 1.5 (J.K. 03-02-2017)
First draft for Period 3.

Version 1.6 (J.K. 30-03-2017)
Final proof read following incremental updates prior to Period 3 submission.

Version 1.7 (D.H.G. 26-03-2018)
Added related information with the fourth year of the project and new papers published in Period 4.

1 Overview of WP6 Architecture

In DREAM we are moving away from Wizard of Oz-controlled (WoZ) behaviour for the robot, which too often is the de facto mode of interaction in Robot Assisted Therapy [1]. Therefore, work package WP6 aims to progress the theoretical and methodological understanding of how an embodied system can interact autonomously with young users in a learning task, specifically developed for atypically developing children. WP6 is concerned with the development of the robot behaviour subsystems to provide social robots with a behaviour underlying social interaction, which permits the robot to be used in Robot Enhanced Therapy (RET) autonomously with supervision. This involves both autonomous behaviour and behaviour created in supervised autonomy, whereby an operator requests certain interventions, which are then autonomously executed by the robot.

A general high level description of the robot control system is shown in Figure 1 (also see Annex 5.2). This describes how the autonomous controller is informed by three external sources: the child behaviour description, sensory information, and current intervention script state. Input from a therapist, e.g., emergency stop, is also present, but not shown in the diagram. Combining these sources, the autonomous controller should trigger an appropriate sequence of action primitives to be performed (as well as some feedback via a graphical user interface), which then gets executed on the robot.

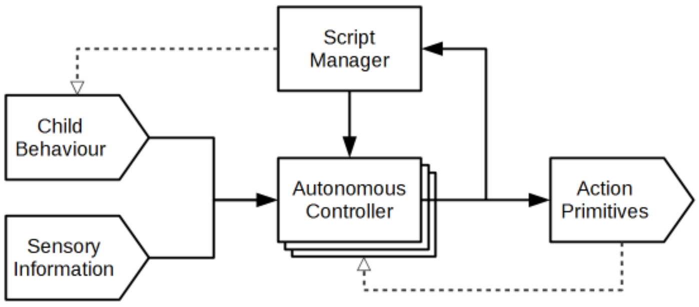


Figure 1: High level description of the robot control system. Child behaviour interpretation (WP5) and sensory information (WP4) provide the context for the autonomous action selection (as well as feedback from motor command execution), in combination with the particular intervention script being applied. The intervention script provides context for child behaviour interpretation.

The autonomous controller is composed of a number of subsystems, as described in the DoW: Reactive, Attention, Deliberative, Self-Monitoring, and Expression and Actuation. In the Reactive subsystem, sensory inputs are immediately acted upon with appropriate actuator outputs. The Attention subsystem determines the robot's focus of attention. In the Deliberative subsystem, the necessary interventions are implemented in a general approach so it is not scenario-specific. The Self-Monitoring subsystem acts as an alarm system in two specifications. An internal one when the robot detects that it cannot act because of a technical limitation or an ethical issue. An external alarm is one where the therapist overrules the robot behaviour selection. Finally, the Expression and Actuation subsystem is responsible for generating believable human/animal-like smooth and natural motions and sounds that are platform independent. These subsystems interact, and must combine their suggested courses of actions to produce a coherent robot behaviour, in the context of constraints laid down by the therapist (for example, the script to be followed, types of behaviour not permissible for this particular child because of individual sensitivities, etc). As a result, we have formulated the following architecture describing how cognitive control informed by the therapy scripts is to be achieved (Figure 2). This design is an iterative improvement on earlier plans laid out in Annex 5.2; the fundamental principles remain the same, but the design required modification to ensure logical information flow when specified to a lower level. Additional components were also added due to the inclusion of the Sandtray as a core aspect for many of the interventions.

A detailed description of the cognitive architecture was provided in deliverable D6.1 at month 18. Within this report we describe the functionality of the Deliberative subsystem in connection with Milestone 4: core functionality in robot behaviour. The final version of this document will be ready for month 54.

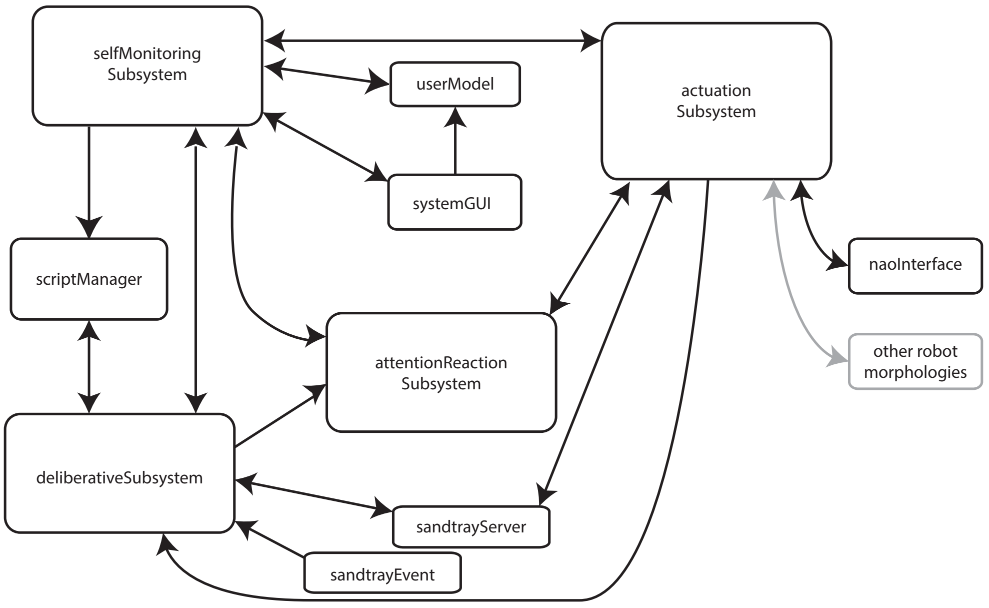


Figure 2: Diagram of the cognitive controller subsystem. The overall WP6 architecture decomposes into 10 components for delivery as part of the DREAM integrated system for use in therapeutic evaluations. This deliverable is concerned with the deliberative aspects of the controller; this includes the following components: deliberativeSubsystem, scriptManager, userModel, systemGUI, sandtrayServer, and sandtrayEvent. The remaining components are discussed in other WP6 deliverables.

2 The Deliberative Subsystem

The main goal for this subsystem is to make decisions on which behaviour has to be selected based on the requirements of the therapy; what the Attention subsystem is capturing from the surroundings; whether or not the child is motivated enough and how he or she is performing in each of the scenarios; and finally, the on-line feedback that the therapist could be providing through the Graphical User Interface (systemGUI component). Such behaviour is sent to the Expression and Actuation subsystem.

2.1 Overview

A central aspect of the cognitive controller is its ability to follow intervention scripts as defined by the clinicians for both diagnosis and therapy. These scripts describe the high-level desired behaviour of the robot[1], and the expected reactions and behaviours of the child, in a defined order.

The decision was made to separate the script manager from the Deliberative subsystem itself (Figure 3). This decision was taken for a number of reasons. Firstly, it enables the cognitive control of the robot to be independent of the precise application domain - with the intention that the developments made would be more generally applicable within the field of social robotics, although the script-based behaviours remain a central part of the behaviour generation of the system. Secondly, it ensures that it would be possible to change the scripts to alter their relative difficulty, by for example including further steps in the intervention, changing the type of intervention, or creating different activities, due to a modular design[2]. As a consequence of this, the Deliberative subsystem is now focussed on action selection considerations, making use of a range of algorithms and methodologies under research (more details will follow later in this document). Thirdly, this division of the script manager from the Deliberative subsystem enables the system to generate coherent behaviour even if there is not a script active at a given moment. This could be useful for periods between the explicit intervention sessions for example, where the robot would then still be able to respond appropriately to environmental stimuli, if so desired by the therapists. These are consistent with the aims expressed within the WP6 Description of Work.

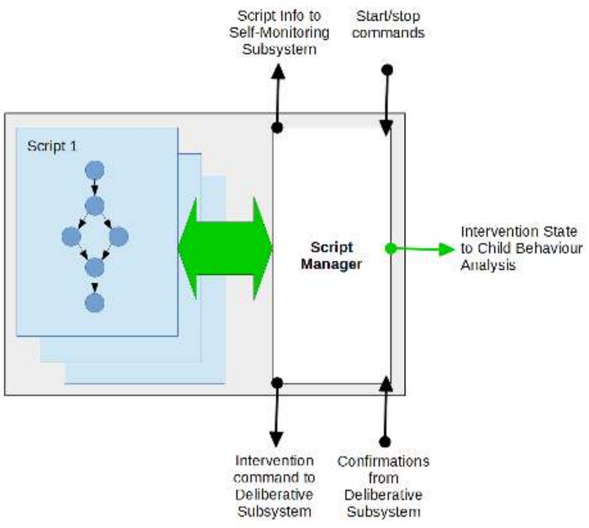

Script Info to Self-Monitoring Subsystem
Start/stop commands
Script 1
Script Manager
Intervention State to Child Behaviour Analysis
Intervention command to Deliberative Subsystem
Confirmations from Deliberative Subsystem

Figure 3: Overview of the script manager subsystem. The scripts are defined independently of the script manager, which is responsible for stepping through the script as appropriate and communicating with the other subsystems as required.

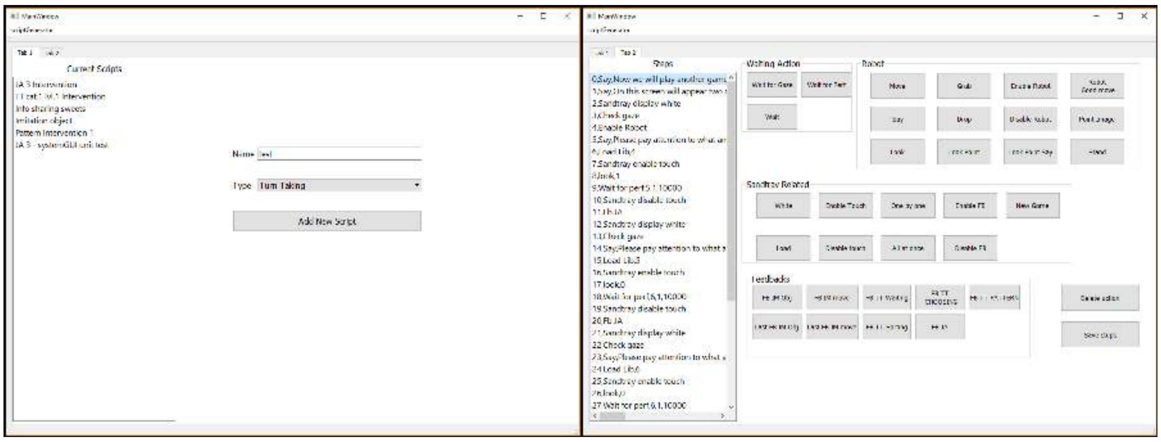
The script manager itself separates the logic necessary to manage progression through the script (by taking into account the available sensory feedback after actions for example) from the script itself. This makes it straightforward to add new scripts or modify existing scripts as required. This logic management is achieved by using a Finite State Machine (FSM). Defining each step in the script as a 3-tuple of the form: *[existing_state, proposed_action, consequent_state]* was found to be insufficient

[1]These predefined robot behaviours differ from the the low-level motor control of the robot, as these may be mixed with other aspects of behaviour not specified explicitly in the high-level intervention script; e.g., the addition of attention to unexpected events in the environment.

[2]As noted above, these high-level scripts do not necessarily completely define the behaviour of the robot, and are distinct from any predefined robot motor control sequences that may be used, such as waving or nodding.

when the details of the intervention scripts were finalised. Some script steps also need to define a series of parameters, such as an expected action for the child to make (to be used in autonomously evaluating their performance) and the time in which this action should be performed. As a consequence, scripts are defined in an XML format, with each step consisting of a series of comma separated values. These values take the form: *[step_id, script_step, (comma separated parameters)]*, where values in parenthesis are optional depending on the proposed action.

As all of the script steps are encoded in unique identifiers, they are not easily read or modified by a human. To address this, a graphical tool was developed (E.S., PLYM) so that the therapists can create and modify scripts in a straightforward manner. This tool is not delivered as part of the core YARP system functionality, but is committed to the project repository as an auxiliary tool. A screenshot of the application can be seen in Figure 4, with the associated output in Figure 5.

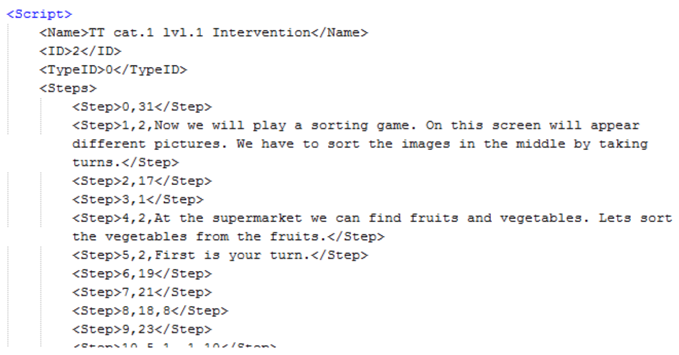


IA 3 Intervention
TT cat.1 lvl.1 Intervention
Info sharing sweets
Imitation object
Pattern Intervention 1
IA 3 - systemGUI unit test

Name: test
Type: Turn Taking

0.Say,Now we will play another gam
1.Say,1:In this screen will appear two
2.Sandtray display white
3.Check gaze
4.Enable Robot
5.Say,Please pay attention to what ar
6.Load Lib,4
7.Sandtray enable touch
8.lock,1
9.Wait for part,5,1,10000
10.Sandtray disable touch
11.Fb,JA
12.Sandtray display white
13.Check gaze
14.Say,Please pay attention to what a
15.Load Lib,5
16.Sandtray enable touch
17.lock,0
18.Wait for part,6,1,10000
19.Sandtray disable touch
20.Fb,JA
21.Sandtray display white
22.Check gaze
23.Say,Please pay attention to what a
24.Load Lib,6
25.Sandtray enable touch
26.lock,0
27.Wait for part,6,1,10000

Figure 4: Screenshots from the scriptGenerator tool developed for therapists to create scripts for the system. The *left* pane shows all current scripts. The *right* pane shows all possible intervention actions as defined by the therapists and the current steps for the script being created. Where parameters are required for a script step, the GUI will automatically request them where necessary.



```
<Script>
    <Name>TT cat.1 lvl.1 Intervention</Name>
    <ID>2</ID>
    <TypeID>0</TypeID>
    <Steps>
        <Step>0,31</Step>
        <Step>1,2,Now we will play a sorting game. On this screen will appear
        different pictures. We have to sort the images in the middle by taking
        turns.</Step>
        <Step>2,17</Step>
        <Step>3,1</Step>
        <Step>4,2,At the supermarket we can find fruits and vegetables. Lets sort
        the vegetables from the fruits.</Step>
        <Step>5,2,First is your turn.</Step>
        <Step>6,19</Step>
        <Step>7,21</Step>
        <Step>8,18,8</Step>
        <Step>9,23</Step>
```

Figure 5: Example output from the scriptGenerator tool. The scripts are stored as XML, which can then be read in code with ease.

The Deliberative subsystem is the primary locus of autonomous action selection in the cognitive controller (Figure 2). This subsystem takes as input sensory data, child behaviour information,

information on what step should be next executed from the therapy script, and higher-level direction from the Self-Monitoring subsystem. It then proposes what action should be taken next by the robot (this proposal is sent to the Expression and Actuation subsystem). In a normal script execution context, the Deliberative subsystem is the primary driver of behaviour, which would typically propose the next script step. Details of the deliberative subsystem implementation are found below in Section 2.2.

There are however a number of circumstances in which this may not the most appropriate action to perform. For example, if the child is detected to have very low engagement with the task (as determined from the WP5 component/s, and/or information from WP4 sensory system saying the child is looking away for example), then it would be appropriate to attempt to re-engage the child with the robot/task prior to executing the next stage in the therapy script. In this case, the Deliberative subsystem can choose to depart from the behaviour defined in the script. For the current implementation in the core robot functionality, it the therapist who suggest this corrective behaviour. but it is part of the development in the Deliberative subsystem for the robot to autonomously learn the correct course of action and be able to suggest the corrective behaviour (see Section 2.3 for further details about research in this direction).

2.2 Core Deliberative Components

Based on the functional description of the cognitive controller system of the DREAM architecture (see section above, and Annex 5.2), core implementations of all WP6 components have been formulated. This section will discuss the Deliberative subsystem component. For full context, the sandtrayServer, sandtrayEvent, systemGUI and scriptManager component descriptions are also described here as they tightly interact with the deliberative subsystem.

The components are defined in terms of the input and output ports, following the guidelines established in the software engineering standards (WP3). These are directly informed by the development of the WP6 control architecture in Y1, where each subsystem was defined in terms of the interactions with other subsystems, and their functions as outlined in the DREAM DoW. Please refer to Figure 1 to provide this context. In the Y3 of the project, the components were more tightly specified (down to the port level) for functional implementation; please see the diagram in Annex 7.3 for full details of port names, port types, and message structures for primitives.

2.2.1 Script Manager

The functions of the script manager are described above: the main point is that the script manager is separated from the rest of the Deliberative subsystem, which is instead focussed on autonomous action selection. The ports for this component are described in Figure 6.

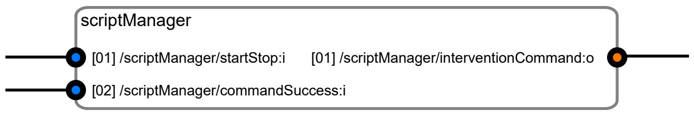

scriptManager
[01] /scriptManager/startStop:i
[01] /scriptManager/interventionCommand:o
[02] /scriptManager/commandSuccess:i

Figure 6: Script Manager component YARP ports. This component handles the script state and provides this information to other components.

2.2.2 Deliberative Subsystem

An overview of the ports for this component are shown in Figure 7. Further to the definition of this version of the YARP component to fit into the DREAM architecture according to the established software engineering framework (see D3.1), a range of work has been conducted on the theoretical basis of autonomous action selection mechanisms for social robots. In addition to the technical principles described below (Section 2.3), this development work has also included the principled examination of what sort of functionality should be undertaken by this component, and what the limitations of this should be with respect to the supervisory oversight provided by the ever-present therapist (as defined by the *supervised autonomy* objective of DREAM). Our work in this regard is summarised in Annexes 5, 6, and 7.

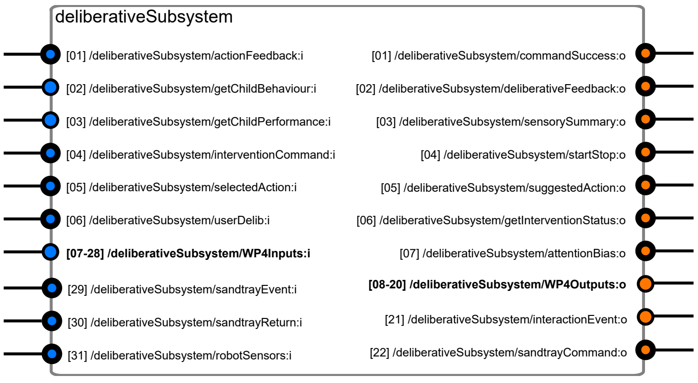

deliberativeSubsystem
[01] /deliberativeSubsystem/actionFeedback:i
[02] /deliberativeSubsystem/getChildBehaviour:i
[03] /deliberativeSubsystem/getChildPerformance:i
[04] /deliberativeSubsystem/interventionCommand:i
[05] /deliberativeSubsystem/selectedAction:i
[06] /deliberativeSubsystem/userDelib:i
[07-28] /deliberativeSubsystem/WP4Inputs:i
[29] /deliberativeSubsystem/sandtrayEvent:i
[30] /deliberativeSubsystem/sandtrayReturn:i
[31] /deliberativeSubsystem/robotSensors:i
[01] /deliberativeSubsystem/commandSuccess:o
[02] /deliberativeSubsystem/deliberativeFeedback:o
[03] /deliberativeSubsystem/sensorySummary:o
[04] /deliberativeSubsystem/startStop:o
[05] /deliberativeSubsystem/suggestedAction:o
[06] /deliberativeSubsystem/getInterventionStatus:o
[07] /deliberativeSubsystem/attentionBias:o
[08-20] /deliberativeSubsystem/WP4Outputs:o
[21] /deliberativeSubsystem/interactionEvent:o
[22] /deliberativeSubsystem/sandtrayCommand:o

Figure 7: Deliberative subsystem component YARP ports: the prefix for the port names is listed in the main text. This component is responsible for the autonomous action selection in cases of deviation from the script, etc. Ports to/from the sensory systems (WP4) have been condensed for clarity.

2.2.3 Sandtray Server and Event

Following earlier studies, the therapists conducting the evaluations as part of the DREAM project (partner UBB) found that children responded well to the use of the Sandtray in interventions. The Sandtray is a large horizontal touch screen that can be used to display various media and games, and for both children and the robot to interact with (such as selecting or moving images); the Sandtray can be seen in Figure 14. As the use of the Sandtray in the interventions increased, it required incorporating into the WP6 system design. This was done in two components: sandtrayServer and sandtrayEvent. sandtrayServer communicates tightly with the Deliberative subsystem component: the Deliberative subsystem can modify what is shown on the screen and can access information from the screen to transmit to the robot. sandtrayEvent raises events when the child interacts with the screen, such as when they select an image, which can be forwarded to as sensory input to WP5 (to be used, for example, in calculating the child performance). The description of these components can be seen in Figure 8. As an additional part of this work, the Sandtray tasks for use in interventions were also developed (PLYM), with images provided by the therapists (UBB). The Sandtray components and code do not rely on any specific hardware, so these components and the Sandtray software can flexibly be re-used should other scenarios or hardware require them.

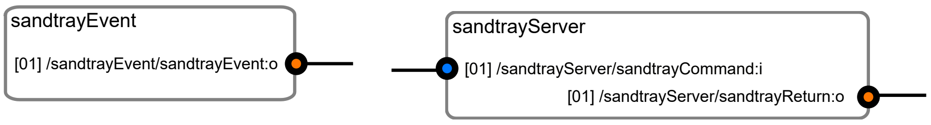

sandtrayEvent
[01] /sandtrayEvent/sandtrayEvent:o
sandtrayServer
[01] /sandtrayServer/sandtrayCommand:i
[01] /sandtrayServer/sandtrayReturn:o

Figure 8: Sandtray event and sandtray server components. Sandtray event notifies the DREAM system of child actions on screen, whereas sandtray server is used for managing robot interaction with the screen.

2.2.4 User Model

The core user model component was developed in period 3. The aim of this component is to store prior diagnosis information about each child, as well as their ongoing intervention session performance with both the robot and the therapist. This information can then be used as part of analysis of the system, and is displayed in the DREAM system GUI as it is useful information for the therapists to view when interventions are being conducted. Each user has a file storing their information, this is accessed through the user model component, which has the function of reading the information, transmitting it, and saving any updates. The user model files are implemented using XML so that they can be extended in the future. It is anticipated that subsequent versions of the user model files may store data required by the SPARC action selection mechanism, so the file format and port pathways for this data to be transmitted to and from the Deliberative subsystem component have been designed in the current version, but without passing the data. The user model port descriptions can be seen in Figure 9. Figure 10 shows a GUI tool created for therapists to input initial user model data from pre-existing diagnosis data.

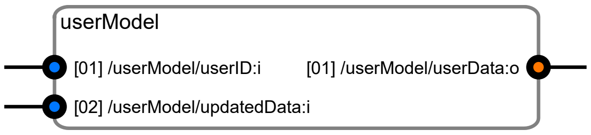
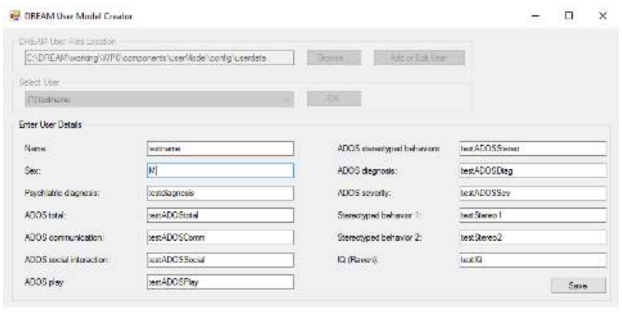


Figure 9: User model component YARP ports. This component is responsible for reading and writing data about the intervention outcomes and child preferences.



C:\DREAM\working\WP6\components\userModel\config\userdata

[1] testname

Enter User Details

Name:	testname	ADOS stereotyped behaviors:	testADOSStereo
Sex:	M	ADOS diagnosis:	testADOSDiag
Psychiatric diagnosis:	testdiagnosis	ADOS severity:	testADOSSev
ADOS total:	testADOStotal	Stereotyped behavior 1:	testStereo1
ADOS communication:	testADOSComm	Stereotyped behavior 2:	testStereo2
ADOS social interaction:	testADOSSocial	IQ (Raven):	testIQ
ADOS play:	testADOSPlay		

Figure 10: GUI tool for generating user models (which are saved as XML documents).

2.2.5 System GUI

The system GUI is based on requirements of supervised autonomy, prototyping with end users, and research into appropriate interface mechanisms [2]. The final design from a port description perspective can be seen in Figure 11. The deliberative subsystem suggests an action for the therapist through the self monitoring subsystem, which is displayed on the GUI. The action will automatically execute after some period of time, or the therapist can speed up the selection, or select an alternative action if the robot needs to go off-script. These corrections are communicated back to the deliberative and self monitoring subsystems. This provides the opportunity to learn from the therapist corrections (as per the SPARC principle; Section 4) to improve the autonomy of the robot in the future. A screenshot of the GUI can be seen in Figure 12, and a technical report for partners using the software can be seen in Annex 7.1.

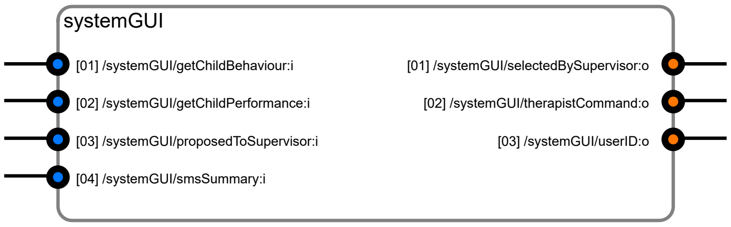
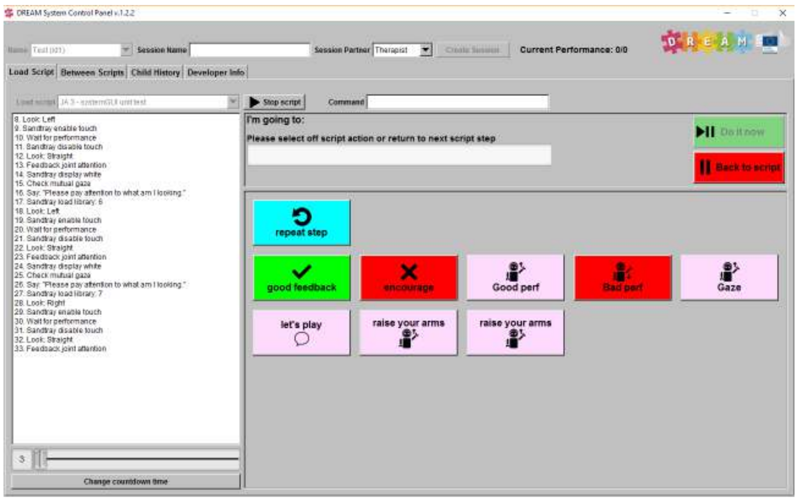

systemGUI
[01] /systemGUI/getChildBehaviour:i
[02] /systemGUI/getChildPerformance:i
[03] /systemGUI/proposedToSupervisor:i
[04] /systemGUI/smsSummary:i
[01] /systemGUI/selectedBySupervisor:o
[02] /systemGUI/therapistCommand:o
[03] /systemGUI/userID:o

Figure 11: System GUI component YARP ports. This component acts as the interface between the therapist and the DREAM system.



8. Look: Left
9. Sandtray enable touch
10. Wait for performance
11. Sandtray disable touch
12. Look: Straight
13. Feedback joint attention
14. Sandtray display white
15. Check mutual gaze
16. Say: "Please pay attention to what am I looking."
17. Sandtray load library: 6
18. Look: Left
19. Sandtray enable touch
20. Wait for performance
21. Sandtray disable touch
22. Look: Straight
23. Feedback joint attention
24. Sandtray display white
25. Check mutual gaze
26. Say: "Please pay attention to what am I looking."
27. Sandtray load library: 7
28. Look: Right
29. Sandtray enable touch
30. Wait for performance
31. Sandtray disable touch
32. Look: Straight
33. Feedback joint attention

I'm going to:

Please select off script action or return to next script step

Figure 12: System GUI screenshot. The layout is based on prototyping with end-users and research into appropriate interface methods. Script steps can be seen to the left, with supervised autonomy options to the right, with possible corrective actions loaded depending on the requirements of the script.

2.3 Action Selection Mechanism

2.3.1 Context

The intervention script provides a series of actions that the robot should execute and that are expected to be followed by a child-specific reaction. However, the actual reaction from the young user can be different to the one expected. To be able to follow the script, the robot needs to find a way to execute the appropriate behaviour in order to obtain the desired reaction from the child. To succeed in such a challenging task, a social Action Selection Mechanism (ASM) is required.

This social ASM has to fulfil multiple specifications: the first one is to allow the robot to detect a state where the execution of an action not planned in the script is required. This can be done by comparing the child's current state to the expected one. Then, once the need to act outside of the script is detected, the ASM has to select an action that would obtain the expected reaction from the child and thus allow continuation of the script.

Having to select an action in a Robot Assisted Therapy (RAT) scenario gives rise to several concerns. First of all, at every moment of the interaction, the action executed by the robot needs to be the correct one, i.e. the one desired to maximise the positive effect of the therapy. Furthermore, as we are working with children, the environment is highly unpredictable and there is probably not a unique general solution for every child and a solution for one child might not be appropriate later in the interaction: the robot needs to be able to adapt to different interaction partners and also to the same partner at different times in the interaction. Lastly, with RAT, we have access to experts with good knowledge of the environment (both the children and the task to be accomplished): the therapists. The therapists can be use to obtain some knowledge, but we can not rely totally on them as this would impose a high workload on them and this is not scalable.

Consequently, the ASM has to follow three principles:

1. The action selected has to be therapeutically correct at every stage of the interaction.
2. The ASM has to be adaptive, i.e. be able to change the action policy over time.
3. The workload on the therapist should be as low as possible.

2.3.2 State of the Art

When having to design an ASM, the simplest possibility is to use a static predefined behaviour, such as a reactive system or a finite state machine, as has been used in some experiments of the Aurora project [3]. This has the advantage of not relying at all on the therapist, but as we are working in the real world, the sensors are represented in a high-dimensional and continuous space. This means that either the behaviour expressed would be simple, or that a more complex controller would probably not be able to be designed manually; there is no way to know in advance what the best action to perform in each state is. Similarly, as it is not possible to have a precise enough model of the child, pure planning is not suitable in our case.

The other main approach used in RAT is the Wizard of Oz (WoZ) paradigm [4, 5]. The robot is not autonomous, but is instead fully teleoperated by a therapist. This can fulfil principles 1 and 2 as the action is always the one that the therapist would desire and the human also provides adaptivity. However, as explained by Thill et al. in [1], there are many reasons which motivate us to move away from WoZ, such as the reliance on the humans violating principle 3 through the imposition of a heavy workload.

A way to fulfil principles 2 and 3 is to provide a robot with learning capabilities. If the robot can learn from successes and errors, it becomes adaptive and does not rely on a human to select its actions. A classical technique used to allow an agent to learn by itself when interacting with an environment is Reinforcement Learning (RL) [6]. With this approach, the robot explores its environment by interacting with it. Through receiving positive or negative rewards from its environment, it optimises an action policy after a learning phase, which can achieve high task performance. This technique could asymptotically also fulfil principle 1, but at the start of the interaction the robot relies on exploration to create its action policy. As such, random actions can be executed to obtain knowledge, and this has the potential to violate principle 1, with possible negative therapeutic outcomes. Furthermore, without external help, this method can take a long time to converge, and depends on rewards from the environment to learn, which might be hard to define explicitly for RAT.

Some researchers have worked on ways to improve RL, for example in [7], authors assume that the environment is not giving rewards, and that a human can give them. This could be suitable for our case as the therapist possesses the adequate knowledge to evaluate the robot actions. A similar approach has been followed by Thomaz et al. in [8] where they combined rewards from the environment, reward from a user and guidance from a user. This allows faster convergence toward an efficient policy and reduces the number of potentially incorrect actions. However in all of these methods, the evaluation is done a posteriori, so if an action is not correct, it will not be executed again, but it has still been executed once. In RAT, this is something we have to avoid as even one incorrect action could have negative consequences.

To cope with this exploration problem, some authors proposed Safe RL [9], in this case, an additional mechanism is combined with the RL policy to prevent dangerous actions from being executed. They present two main ways to make RL safer: using initial knowledge to prevent the execution of actions in specific cases, or to bootstrap the learning with safe demonstration, and to only explore around them. But even these methods can not guarantee that only correct actions will be executed as there is still a reliance on exploration and all of the potentially dangerous state-action pairs can not be defined in advance.

Another method proposed to achieve autonomy in RAT without having the robot to explore environment by itself is inspired by Learning from Demonstration and the WoZ paradigm. In [10], authors propose that the interaction is begun in a classical WoZ setup. The robot is only controlled by the therapist initially, then when enough data has been gathered, batch learning is applied and an action policy is derived. As only correct actions have been given to the robot, and if we assume that enough data points are obtained to cover the environment space, the action policy derived is correct. Only therapeutically valid actions will be executed, and this is achieved without further reliance on the therapist to control the robot. However, as no further learning is used, the ASM is not adaptive, so it can not cope with additional changes in the child behaviour. As such, even if principle 1 and 3 are fulfilled, principle 2 is still missing.

2.3.3 Proposed Solution

As shown previously, there is currently no solution in the literature fulfilling all of the desiderata for our Action Selection Mechanism. Methods either require important knowledge to be hard-coded inside the software, rely on exploration involving randomness, impose a heavy workload on the therapists, or are not adaptive once the learning is finished.

To be able to fulfil principles 2 and 3, the ASM has to include a machine learning component. This is the only way to provide the required adaptivity without relying on a human. However, the majority of the algorithms used for learning face a trade-off between exploitation and exploration: the agent

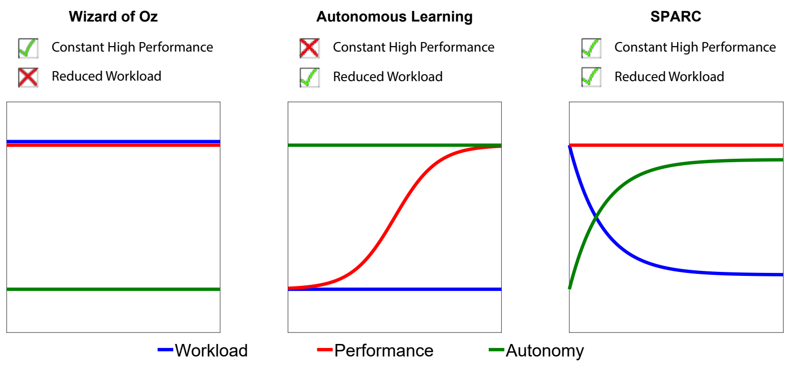


Figure 13: Comparison of SPARC with WoZ and Reinforcement Learning in terms of autonomy, supervisor workload, and performance. The aim is for workload to be low, whilst autonomy and performance are high.

first needs to explore its environment to be able to select the best action to perform later, and generally there is an element of randomness in the exploration. As encompassed by principle 1, in RAT there is no place for randomness. We need to find an ASM allowing the robot to become autonomous, to learn the best action to execute in a highly unpredictable, continuous and complex space without relying on random exploration.

As developers, we have limited knowledge of the best action to execute in an unexpected state; this knowledge is the expertise of the therapist. This expertise can be used to provide the initial knowledge for our learning mechanism. Furthermore, this knowledge can also prevent incorrect actions from being executed during the learning phase, or indeed in any part of the interaction.

This is the reason why we propose Supervised Progressively Autonomous Robot Competencies as a solution (SPARC; see Section 4). This technique relies on a system of suggestion/correction: the robot selects an action according to the ASM and suggests it to the supervisor (in this case, the therapist). In response, the therapist can either do nothing and thus the suggested action is executed, or select a different action for the robot to execute. This concept ensures that the right action is always executed: fulfilling principle 1. Simultaneous learning on the robot side allows the suggested action to be more appropriate with more interactions, reducing the workload on the therapist over time and fulfilling principles 2 and 3 (behavioural adaptivity and low therapist workload), whilst maintaining principle 1 (correct therapeutic action). This method is described in more detail in [11] and Section 4 presents the work done in evaluating this approach.

Figure 13 presents the concept of SPARC compared to WoZ or RL in terms of workload on the supervisor, autonomy and performance. With WoZ, at all times, there is no autonomy and a high workload on the supervisor providing a high performance. With RL, the human is not involved, so the autonomy is high and the workload is low. At the start, the robot is exploring the environment resulting in low task performance in the learning phase, and this performance rises until reaching an asymptotically high value once the robot knows how to act. SPARC imposes a high workload on the therapist at the start, when the robot is still learning. This provides faster learning, which allows the robot to be more autonomous with time and decreases the workload on the supervisor, whilst maintaining good task performance throughout the interaction.

2.4 Planned Work

In Period 1, the WP6 and Deliberative subsystem architectures were established from a theoretical basis, leading to the development of preliminary Deliberative subsystem and script manager components (see Sections 2.1 and 2.2). During this period, WP6 also supported manually controlled evaluation of the diagnosis/intervention scripts (defined in deliverable D1.1), further detail can be seen in Section 3 below. This provided guidance for further development within Period 2, which gave increased focus on how machine learning can be utilised to improve the autonomy of the robot in interactions (see Section 4 for details).

During Period 3, the Deliberative subsystem was developed in accordance with milestone MS4 specified within the WP6 Description of Work ("core functionality in robot behaviour"). Specifically, this included completed versions of the script manager, sandtray event, and sandtray server components, alongside core versions of the user model, GUI and deliberative subsystem components, as described in Sections 2.1 and 2.2. This provides the software for autonomous progression through the intervention scripts as defined in deliverable D1.1.

Section 2.3 described the approach to machine learning (SPARC) adopted within the context of the Deliberative subsystem that will strive to increase the autonomy of the robot behaviour. Successful evaluation of this approach in non-therapeutic environments in Period 2 (discussed in Annex 6.3 and Section 4) provided a solid platform for further exploration in Period 3.

The SPARC model had already been tested with a Neural Network and Reinforcement Learning, with preliminary results suggesting promising performance when compared to WoZ or classical Interactive Reinforcement Learning (Section 4). Period 3 has seen a comparison with another method from the field of Interactive Machine Learning. Results from a study with 40 participants [2] show that the principles advocated by SPARC allow a safer, quicker and more efficient learning than Interactive Reinforcement Learning as proposed in [8]. The current SPARC implementation exists outside of the core Deliberative subsystem component, but the inclusion of the SPARC approach was allowed for in the design adopted in Period 3. It is planned for the approach to be evaluated in settings more closely resembling the therapeutic environment of the DREAM project to provide additional validation prior to potential integration into the cognitive controller moving forwards.

Period 4 has seen the continuation of the evaluation of the SPARC approach, and the exploration of how SPARC and the ideas developed in DREAM could be applied to other fields of HRI. A new algorithm has been proposed in 8.1, to learn quickly a complex action policy for teaching robots to interact with humans, it extends SPARC to allow the supervisor to highlight features in the environment relevant for the selected action. A study is currently being carried out to evaluate how human supervision can teach a robot to support child learning during an educational game using an implementation of the SPARC approach 8.2.

3 Script Following

A primary objective of the first year of work was the evaluation of manually controlled ('wizarded' or tele-operated) versions of the diagnosis/intervention scripts defined in deliverable D1.1, as relevant to T2.1. This evaluation provides guidance for the further development of the autonomous interpretation and behaviour systems for the DREAM architecture.

In Y1, WP6 provided substantial support to provide the systems necessary for these evaluations. The robot behaviour capabilities to enable execution of each of the basic versions of the scripts has been implemented: imitation task, joint attention task, and the turn-taking task. These systems have been deployed and used at partner UBB.

Two methods were used to provide this functionality. For the imitation and joint attention tasks, behaviours were constructed in the Aldebaran-produced Choregraphe suite, such that a therapist could manually control the robot behaviours for each of the stages of the task. Details of this system can be found in Annex 5.4 of this deliverable. For the turn-taking task, since the Sandtray device is used, a standalone system using the software engineering standards defined in WP3 were used. Details of this system can be found in Annex 5.3 of this deliverable.

This work provided the basis for further developments within WP6 in Y2. The development of the behaviours for each of the intervention tasks was reused in the autonomous versions of these tasks, along with further behaviours as required. Furthermore, the establishment of preliminary versions of the various components using the software engineering framework, and the development of the WP6 cognitive control architecture, facilitated the implementation of the autonomous versions of these components, and their subsequent integration, in Y3, deployment and evaluation, in Y4, with the rest of the DREAM architecture.

4 Increasing Robot Autonomy

In the second year of work there was an increased focus on studying how machine learning could be used to gradually take over from the therapist, or in the broader sense of the word, the “Wizard”. For this, a new method was developed, dubbed SPARC (Supervised Progressively Autonomous Robot Competencies). SPARC proposes actions to the supervisor and observes which actions the Wizard takes in which states; the states are comprised of internal states of the robot and external states in the social and physical environment, including the child. SPARC gradually builds up a state-action model, and as the interaction progresses, suggests more appropriate actions to the Wizard. The Wizard can relinquish control to SPARC by accepting its proposed actions.

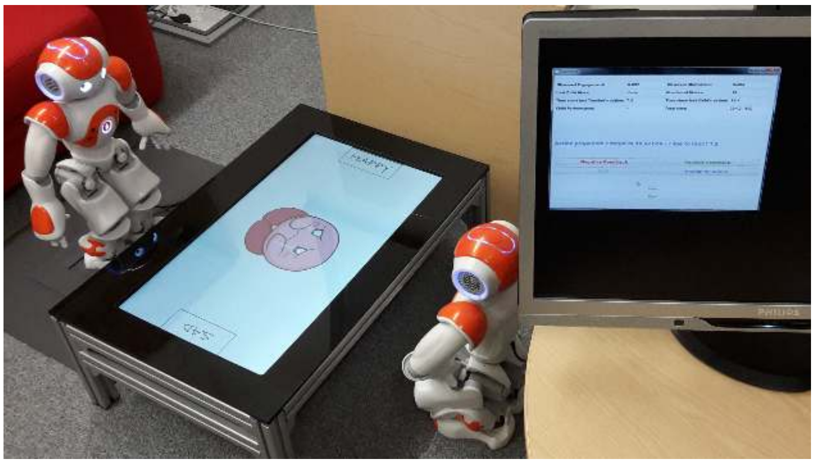


Figure 14: Setup used for the user study from the perspective of the human supervisor. The *child-robot* (left) stands across the touchscreen (centre-left) from the *wizarded-robot* (centre-right). The supervisor can oversee the actions of the *wizarded-robot* through the GUI and intervene if necessary (*right*).

In Period 2, the architecture was further developed from the simulation model presented in Period 1. The SPARC model was tested both with a Neural Network and Reinforcement Learning and developed to suit the context of Robot Assisted Therapy [11, 12]. A main focus has been on evaluating how human operators, or Wizards, perceive the gradually increasing autonomy of the robot and the impact on the task performance.

To this end we used a novel method, in which the child in the interaction is substituted by a robot running a "child" model (Figure 14). This allows experimenting without putting undue pressure and stress on young participants, and provides a setup with high repeatability, which is required for rigorously testing the SPARC architecture. A number of hypotheses were evaluated in a study [11]. Overall, the study showed that controlling a learning robot enables supervisors to achieve similar task performance as with a non-learning robot, but with both fewer interventions and a reduced perception of workload. These results demonstrate the utility of the SPARC concept and its potential effectiveness to reduce the cognitive and workload on human operators.

SPARC has recently been implemented in a restricted (non-therapeutic) environment and compared to previous work done in Interactive Reinforcement Learning (IRL): the environmental reward is combined with reward given by the user after the action execution. Analysis of the results indicates that SPARC is compatible with Reinforcement Learning, and it leads to faster and better results than classical IRL with a lower workload on the supervisor. From these results, several limits of the current proposition of SPARC have been highlighted, such as the limitation to a single type of inputs from the user, assumption of a perfect supervisor and the current discretisation of time. To be useful in real human-robot interactions, SPARC will have to tackle these challenges.

Year 4 saw the implementation of a more complete version of SPARC and this new implementation is being tested in a human-robot interaction with children in the wild where the system faces challenges not explored in previous studies. The context for the study is teaching children a food web by playing an interactive game supported by the robot. The learning algorithm is a version of Nearest Neighbours which has been adapted to be suitable to complex environments such as HRI and human supervision during the teaching. The algorithm has been published in [13] and the study is currently being ran. Expected results should show that SPARC can be applied to teach a robot a complex action policy for interacting with humans.

References

[1] Serge Thill, Cristina A Pop, Tony Belpaeme, Tom Ziemke, and Bram Vanderborght. Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. *Paladyn*, 3(4):209–217, 2012.

[2] Emmanuel Senft, Séverin Lemaignan, Paul E Baxter, and Tony Belpaeme. Sparc: an efficient way to combine reinforcement learning and supervised autonomy. In *FILM Workshop at NIPS'16*, 2016.

[3] Kerstin Dautenhahn. Robots as social actors: Aurora and the case of autism. In *Proc. CT99, The Third International Cognitive Technology Conference, August, San Francisco*, volume 359, page 374, 1999.

[4] Jelle Saldien, Kristof Goris, Bram Vanderborght, Johan Vanderfaeillie, and Dirk Lefeber. Expressing emotions with the social robot probo. *International Journal of Social Robotics*, 2(4):377–389, 2010.

[5] Ben Robins, Kerstin Dautenhahn, and Paul Dickerson. From isolation to communication: a case study evaluation of robot assisted play for children with autism with a minimally expressive humanoid robot. In *Advances in Computer-Human Interactions, 2009. ACHI'09. Second International Conferences on*, pages 205–211. IEEE, 2009.

[6] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

[7] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009.

[8] Andrea L Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6):716–737, 2008.

[9] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

[10] W Bradley Knox, Samuel Spaulding, and Cynthia Breazeal. Learning social interaction from the wizard: A proposal. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[11] Emmanuel Senft, Paul Baxter, James Kennedy, and Tony Belpaeme. SPARC: Supervised Progressively Autonomous Robot Competencies. In *Social Robotics*, pages 603–612. Springer, 2015.

[12] Emmanuel Senft, Paul Baxter, and Tony Belpaeme. Human-guided learning of social action selection for robot-assisted therapy. In *4th Workshop on Machine Learning for Interactive Systems*, 2015.

[13] Emmanuel Senft, Séverin Lemaignan, Paul Baxter, and Tony Belpaeme. Toward supervised reinforcement learning with partial states for social hri. In *Proceedings of 2017 AAAI Fall Symposium - AIHRI*, 2017.

5 Period 1 Annexes

5.1 Senft, E. et al. (2015), When is it better to give up? Towards autonomous action selection for robot assisted ASD therapy

Bibliography - Senft, E., Baxter, P., Kennedy, J., Belpaeme, T (2015), "When is it better to give up? Towards autonomous action selection for robot assisted ASD therapy", HRI'15 Extended Abstracts, doi: 10.1145/2701973.2702715

Abstract - Robot Assisted Therapy (RAT) for children with ASD has found promising applications. In this paper, we outline an autonomous action selection mechanism to extend current RAT approaches. This will include the ability to revert control of the therapeutic intervention to the supervising therapist. We suggest that in order to maintain the goals of therapy, sometimes it is better if the robot gives up.

Relation to WP - This work directly contributes to Task T6.3.

5.2 Baxter, P. et al. (2015), Technical Report: Organisation of Cognitive Control and Robot Behaviour

Abstract - The purpose of this technical report is to summarise the motivations and constraints underlying the cognitive control structures, and to outline an organisation of these subsystems. This is a proposal only; this document is intended to be a working one, to be updated as required during development. This version of the report is based primarily on the discussions that took place in Brussels (23/01/15).

Relation to WP - This work directly contributes to Task T6.3.

5.3 Baxter, P. et al. (2015), Technical Report: Sandtray Wizard-of-Oz System for Turn-taking Intervention

Abstract - In this technical report we describe the software organisation of the Sandtray system created for the turn-taking diagnosis/intervention interactions. This system is based on the organisation defined by the WP3 software engineering standards, although at the moment does not fit into the rest of the DREAM system: this was to facilitate ease of setup and launch for the end-user (i.e. minimal installation, and no compilation required). The WoZ system provides a GUI from which the therapist can control the robot behaviour in the turn-taking task, and logs of the interaction are automatically stored for retrospective analysis.

Relation to WP - This work provides the basis of work in Task T6.3, and is relevant to T2.1.

5.4 Esteban, P.G. et al. (2015), Technical Report: Manual for the use of Choregraphe boxes in Wizard of Oz experiments

Abstract - In this technical report we describe a manual to help UBB team in the development of the Wizard of Oz experiments within Work Package 2. Both PLYM and VUB have collaborated to develop the corresponding modules in Choregraphe. This manual aims at being a reference point to ease the habituation of the therapists to the software.

Relation to WP - This work provides the basis of work in Task T6.3, and is relevant to T2.1.

6 Period 2 Annexes

6.1 Baxter, P. et al. (2015), Touchscreen-Mediated Child-Robot Interactions Applied to ASD Therapy

Bibliography - Baxter, P., Matu, S., Senft, E., Costescu, C., Kennedy, J., David, D., and Belpaeme, T. (2015) Touchscreen-Mediated Child-Robot Interactions Applied to ASD Therapy. New Friends symposium, Almere, The Netherlands.

Abstract - Robots are finding increasing application in the domain of ASD therapy as they provide a number of advantageous properties such as replicability and controllable expressivity. In this abstract we introduce a role for touchscreens that act as mediating devices in therapeutic robot-child interactions. Informed by extensive work with neurotypical children in educational contexts, an initial study using a touchscreen mediator in support of robot assisted ASD therapy was conducted to examine the feasibility of this approach, in so doing demonstrating how this application provides a number of technical and potentially therapeutic advantages.

Relation to WP - This paper summarises our use of touchscreen devices as mediating devices in child-robot interaction, and its specific use in diagnosing ASD.

6.2 Senft, E. et al. (2015) Human-Guided Learning of Social Action Selection for Robot-Assisted Therapy

Bibliography - Senft, E., Baxter, P., and Belpaeme, T. (2015). Human-guided learning of social action selection for robot-assisted therapy. In 4th Workshop on Machine Learning for Interactive Systems.

Abstract - This paper presents a method for progressively increasing autonomous action selection capabilities in sensitive environments, where random exploration-based learning is not desirable, using guidance provided by a human supervisor. We describe the global framework and a simulation case study based on a scenario in Robot Assisted Therapy for children with Autism Spectrum Disorder. This simulation illustrates the functional features of our proposed approach, and demonstrates how a system following these principles adapts to different interaction contexts while maintaining an appropriate behaviour for the system at all times.

Relation to WP - This paper sketches the early ideas on progressively learning autonomous behaviour from a human Wizard.

6.3 Senft, E. et al. (2015) SPARC: Supervised Progressively Autonomous Robot Competencies

Bibliography - Senft, E., Baxter, P., Kennedy, J., and Belpaeme, T. (2015). SPARC: Supervised Progressively Autonomous Robot Competencies. In Social Robotics (pp. 603-612). Springer International Publishing.

Abstract - The Wizard-of-Oz robot control methodology is widely used and typically places a high burden of effort and attention on the human supervisor to ensure appropriate robot behaviour, which may distract from other aspects of the task engaged in. We propose that this load can be reduced by enabling the robot to learn online from the guidance of the supervisor to become progressively more autonomous: Supervised Progressively Autonomous Robot Competencies (SPARC). Applying this concept to the domain of Robot Assisted Therapy (RAT) for children with Autistic Spectrum Disorder, a novel methodology is employed to assess the effect of a learning robot on the workload of the human supervisor. A user study shows that controlling a learning robot enables supervisors to achieve similar task performance as with a non-learning robot, but with both fewer interventions and a reduced perception of workload. These results demonstrate the utility of the SPARC concept and its potential effectiveness to reduce load on human WoZ supervisors.

Relation to WP - This paper describes the SPARC architecture, which can learn which actions to take in which states by observing a Wizard. The paper also presents a first user study which validates the concept.

6.4 Senft, E. et al. (2016) Providing a Robot with Learning Abilities Improves its Perception by Users

Bibliography - Senft, E., Baxter, P., Kennedy, J., Lemaignan, S. and Belpaeme, T. (2016) Providing a Robot with Learning Abilities Improves its Perception by Users. In Proceedings of the 11th Annual ACM/IEEE International Conference on Human-Robot Interaction, Christchurch, New Zealand.

Abstract - Subjective appreciation and performance evaluation of a robot by users are two important dimensions for Human- Robot Interaction, especially as increasing numbers of people become involved with robots. As roboticists we have to carefully design robots to make the interaction as smooth and enjoyable as possible for the users, while maintaining good performance in the task assigned to the robot. In this paper, we examine the impact of providing a robot with learning capabilities on how users report the quality of the interaction in relation to objective performance. We show that humans tend to prefer interacting with a learning robot and will rate its capabilities higher even if the actual performance in the task was lower. We suggest that adding learning to a robot could reduce the apparent load felt by a user for a new task and improve the users evaluation of the system, thus facilitating the integration of such robots into existing work flows.

Relation to WP - This study looks into how an operator (a Wizard) subjectively experiences a robot which gradually learns and takes over the operator's task.

6.5 Baxter, P. et al. (2016) Cognitive Architectures for Social Human-Robot Interaction

Bibliography - Baxter, P., Lemaignan, S. and Trafton, G. (2016) Cognitive Architectures for Social Human-Robot Interaction. In Workshop on Cognitive Architectures in Human-Robot Interaction, at the 11th Annual ACM/IEEE International Conference on Human-Robot Interaction, Christchurch, New Zealand.

Abstract - Social HRI requires robots able to use appropriate, adaptive and contingent behaviours to form and maintain engaging social interactions with people. Cognitive Architectures emphasise a generality of mechanism and application, making them an ideal basis for such technical developments. Following the successful first workshop on Cognitive Architectures for HRI at the 2014 HRI conference, this second edition of the workshop focusses specifically on applications to social interaction. The full-day workshop is centred on participant contributions, and structured around a set of questions to provide a common basis of comparison between different assumptions, approaches, mechanisms, and architectures. These contributions will be used to support extensive and structured discussions, with the aim of facilitating the development and application of cognitive architectures to social HRI systems. By attending, we envisage that participants will gain insight into how the consideration of cognitive architectures complements the development of autonomous social robots

Relation to WP - A position paper framing the need and state-of-the-art in cognitive architectures for social HRI, relevant to the deliberative subsystem in WP6.

6.6 Baxter, P. (2016) Memory-Centred Cognitive Architectures for Robots Interacting Socially with Humans

Bibliography - Baxter, P. (2016) Memory-Centred Cognitive Architectures for Robots Interacting Socially with Humans. In Workshop on Cognitive Architectures in Human-Robot Interaction, at the 11th Annual ACM/IEEE International Conference on Human-Robot Interaction, Christchurch, New Zealand.

Abstract - The Memory-Centred Cognition perspective places an active association substrate at the heart of cognition, rather than as a passive adjunct. Consequently, it places prediction and priming on the basis of prior experience to be inherent and fundamental aspects of processing. Social interaction is taken here to minimally require contingent and co-adaptive behaviours from the interacting parties. In this contribution, I seek to show how the memory-centred cognition approach to cognitive architectures can provide an means of addressing these functions. A number of example implementations are briefly reviewed, particularly focusing on multi-modal alignment as a function of experience-based priming. While there is further refinement required to the theory, and implementations based thereon, this approach provides an interesting alternative perspective on the foundations of cognitive architectures to support robots engage in social interactions with humans.

Relation to WP - A paper providing theoretical insights in how associative memories can serve as the backbone of a cognitive architecture. This approach is at present not implemented in DREAM, but is being explored in the context of the SPARC architecture.

7 Period 3 Annexes

7.1 Kennedy, J. et al. (2017), Technical Report: A Guide to Using systemGUI

Abstract - The purpose of this technical report is to provide a guide for all technical partners and the end users of the systemGUI developed as part of WP6. Technical partners may need this information in order to perform tests with an expanded subset of the integrated DREAM system. Therapists will need to understand how the systemGUI interface is used in order to run planned experiments effectively. This report is based on the current systemGUI at the production date; details may change to align with the GUI changes in the future.

Relation to WP - This work directly contributes to Task T6.3.

7.2 Lemaignan, S. et al. (2016), Towards "Machine-Learnable" Child-Robot Interactions: the PInSoRo Dataset

Bibliography - Lemaignan, S., Kennedy, J., Baxter, P., Belpaeme, T (2015), Towards "Machine-Learnable" Child-Robot Interactions: the PInSoRo Dataset, Workshop on Long-term Child-Robot Interaction at RO-MAN 2016

Abstract - Child-robot interactions are increasingly being explored in domains which require longer-term application, such as healthcare and education. In order for a robot to behave in an appropriate manner over longer timescales, its behaviours should be coterminous with that of the interacting children. Generating such sustained and engaging social behaviours is an on-going research challenge, and we argue here that the recent progress of deep machine learning opens new perspectives that the HRI community should embrace. As an initial step in that direction, we propose the creation of a large open dataset of child-robot social interactions. We detail our proposed methodology for data acquisition: children interact with a robot puppeted by an expert adult during a range of playful face-to-face social tasks. By doing so, we seek to capture a rich set of human-like behaviours occurring in natural social interactions, that are explicitly mapped to the robots embodiment and affordances.

Relation to WP - A paper providing theoretical insights in how current trends in deep neural networks may assist in generating autonomous robot behaviours, specifically with a focus on child-robot interactions. This approach is at present not implemented in DREAM, but forms part of the ongoing research in developing deliberative cognitive controllers as per Task T6.3.

7.3 Kennedy, J. et al. (2017), Technical Report: WP6 Full Port Descriptions

Abstract - The purpose of this technical report is to provide a guide for all technical partners to the port-level implementation of WP6. Port names and types are described for all WP6 components, with additional detail for those ports which are exposed to components outside of WP6 (primitives). This is useful for both developers of WP6 and other work packages to see the information flow and decomposition of WP6.

Relation to WP - This work directly contributes to Task T6.3.

7.4 Senft, E. et al. (2016), SPARC: an efficient way to combine reinforcement learning and supervised autonomy.

Bibliography - E. Senft, S. Lemaignan, P. E. Baxter, and T. Belpaeme, Sparc: an efficient way to combine reinforcement learning and supervised autonomy, at workshop on Future of Interactive Learning Machine at NIPS 2016

Abstract - Shortcomings of reinforcement learning for robot control include the sparsity of the environmental reward function, the high number of trials required before reaching an efficient action policy and the reliance on exploration to gather information about the environment, potentially resulting in undesired actions. These limits can be overcome by adding a human in the loop to provide additional information during the learning phase. In this paper, we propose a novel way to combine human inputs and reinforcement by following the Supervised Progressively Autonomous Robot Competencies (SPARC) approach. We compare this method to the principles of *Interactive Reinforcement Learning* as proposed by Thomaz and Breazeal. Results from a study involving 40 participants show that using SPARC increases the performance of the learning, reduces the time and number of inputs required for teaching and faces fewer errors during the learning process. These results support the use of SPARC as an efficient method to teach a robot to interact with humans.

Relation to WP - Research on the combination of SPARC and Reinforcement Learning - initial results of study.

7.5 Senft, E. et al. (2016), Supervised Autonomy for Online Learning in Human-Robot Interaction

Submitted - Special issue on User Profiling and Behavior Adaptation for Human-Robot Interaction (Letters in Pattern Recognition)

Abstract - When a robot is learning it needs to explore its environment and how its environment responds on its actions. When the environment is large and there are a large number of possible actions the robot can take, this exploration phase can take prohibitively long. However, exploration can often be optimised by letting a human expert guide the robot during its learning. Interactive machine learning, in which a human user interactively guides the robot as it learns, has been shown to be an effective way to teach a robot. It requires an intuitive control mechanism to allow the human expert to provide feedback on the robot's progress. This paper presents a novel method which combines Reinforcement Learning and Supervised Progressively Autonomous Robot Competencies (SPARC). By allowing the user to fully control the robot and by treating rewards as implicit, SPARC aims to learn an action policy while maintaining human supervisory oversight of the robot's behaviour. This method is evaluated and compared to Interactive Reinforcement Learning in a robot teaching task. Qualitative and quantitative results indicate that SPARC allows for safer and faster learning by the robot, whilst not placing a high workload on the human teacher.

Relation to WP - Research on the combination of SPARC and Reinforcement Learning - full results of study.

7.6 Senft, E. et al. (2017), Leveraging Human Inputs in Interactive Machine Learning for Human Robot Interaction

Bibliography - E. Senft, S. Lemaignan, P. E. Baxter, and T. Belpaeme, Leveraging Human Inputs in Interactive Machine Learning for Human Robot Interaction, HRI'17 Extended Abstracts.

Abstract - A key challenge of HRI is allowing robots to be adaptable, especially as robots are expected to penetrate society at large and to interact in unexpected environments with non-technical users. One way of providing this adaptability is to use Interactive Machine Learning, i.e. having a human supervisor included in the learning process who can steer the action selection and the learning in the desired direction. We ran a study exploring how people use numeric rewards to evaluate a robot's behaviour and guide its learning. From the results we derive a number of challenges when designing learning robots: what kind of input should the human provide? How should the robot communicate its state or its intention? And how can the teaching process by made easier for human supervisors?

Relation to WP - Research on the combination of SPARC and Reinforcement Learning - Additional results of the study, and challenges.

8 Period 4 Annexes

8.1 Senft, E. et al. (2017), Toward Supervised Reinforcement Learning with Partial States for Social HRI

Bibliography - E. Senft, S. Lemaignan, P. E. Baxter, and T. Belpaeme, Toward Supervised Reinforcement Learning with Partial States for Social HRI at 4th AAAI FSS on Artificial Intelligence for Social Human-Robot Interaction (AI-HRI), USA, Arlington, November 2017.

Abstract - Social interacting is a complex task for which machine learning holds particular promise. However, as no sufficiently accurate simulator of human interactions exists today, the learning of social interaction strategies has to happen online in the real world. Actions executed by the robot impact on humans, and as such have to be carefully selected, making it impossible to rely on random exploration. Additionally, no clear reward function exists for social interactions. This implies that traditional approaches used for Reinforcement Learning cannot be directly applied for learning how to interact with the social world. As such we argue that robots will profit from human expertise and guidance to learn social interactions. However, as the quantity of input a human can provide is limited, new methods have to be designed to use human input more efficiently. In this paper we describe a setup in which we combine a framework called Supervised Progressively Autonomous Robot Competencies (SPARC), which allows safer online learning with Reinforcement Learning, with the use of partial states rather than full states to accelerate generalisation and obtain a usable action policy more quickly.

Relation to WP - New algorithm to learn quickly a complex action policy for teaching robots to interact with humans. Good combination with SPARC.

8.2 Senft, E. et al. (2018), Robots in the classroom: Learning to be a Good Tutor

Bibliography - E. Senft, S. Lemaignan, M. Bartlett, P. E. Baxter, and T. Belpaeme, Robots in the classroom: Learning to be a Good Tutor at Robots for Learning R4L : Inclusive Learning Workshop at

HRI 2018, USA, Chicago, March 2018.

Abstract - To broaden the adoption and be more inclusive, robotic tutors need to tailor their behaviours to their audience. Traditional approaches, such as Bayesian Knowledge Tracing, try to adapt the content of lessons or the difficulty of tasks to the current estimated knowledge of the student. However, these variations only happen in a limited domain, predefined in advance, and are not able to tackle unexpected variation in a student's behaviours. We argue that robot adaptation needs to go beyond variations in preprogrammed behaviours and that robots should in effect learn online how to become better tutors. A study is currently being carried out to evaluate how human supervision can teach a robot to support child learning during an educational game using one implementation of this approach.

Relation to WP - Proposition of a new study to explore how SPARC and the ideas developed in DREAM could be applied to other fields of HRI.

8.3 Cao, Hoang-Long et al. (2018), An End-User Interface to Generate Homeostatic Behavior for NAO Robot in Robot-Assisted Social Therapies

Bibliography - Cao, H. L., De Beir, A., Esteban, P. G., Simut, R., Van de Perre, G., Lefeber, D., and Vanderborght, B., An End-User Interface to Generate Homeostatic Behavior for NAO Robot in Robot-Assisted Social Therapies. In: Rojas I., Joya G., Catala A. (eds) Advances in Computational Intelligence. IWANN 2017. Lecture Notes in Computer Science, vol 10306. Springer, Cham.

Abstract - Homeostatic drive theory is a popular approach for decision making of robot behavior in social robotic research. It is potentially to be used in social therapies. To increase the involvement of end-users in the robot's control, we present an end-user interface allowing the therapists to generate homeostatic behavior for NAO robot in social skills training for children. We demonstrate the system by two interactions in which the robot homeostatic behavior is adapted to children's behavior. The result shows that the system provides a practical solution for therapists to implement interaction scenarios to robot behavior.

Relation to WP - Research on homeostatic behavior controller. This approach forms part of the ongoing research in developing deliberative cognitive controllers as per Task T6.3.

8.4 Cao, Hoang-Long et al. (2018), A Collaborative Homeostatic-Based Behavior Controller for Social Robots in HumanRobot Interaction Experiments

Bibliography - Cao, H. L., Esteban, P. G., De Beir, A., Simut, R., Van de Perre, G., Lefeber, D., and Vanderborght, B., A Collaborative Homeostatic-Based Behavior Controller for Social Robots in HumanRobot Interaction Experiments in Int J of Soc Robotics (2017) 9: 675.

Abstract - Robots have been gradually leaving laboratory and factory environments and moving into human populated environments. Various social robots have been developed with the ability to exhibit social behaviors and collaborate with non-expert users in different situations. In order to increase the degree of collaboration between humans and the robots in humanrobot joint action systems, these robots need to achieve higher levels of interaction with humans. However, many social robots are

operated under teleoperation modes or pre-programmed scenarios. Based on homeostatic drive theory, this paper presents the development of a novel collaborative behavior controller for social robots to jointly perform tasks with users in humanrobot interaction (HRI) experiments. Manual work during the experiments is reduced, and the experimenters can focus more on the interaction. We propose a hybrid concept for the behavior decisionmaking process, which combines the hierarchical approach and parallel-rooted, ordered, slip-stack hierarchical architecture. Emotions are associated with behaviors by using the two-dimensional space model of valence and arousal. We validate the usage of the behavior controller by a joint attention HRI scenario in which the NAO robot and a therapist jointly interact with children.

Relation to WP - Research on homeostatic behavior controller. This approach forms part of the ongoing research in developing deliberative cognitive controllers as per Task T6.3.

8.5 Cao, Hoang-Long et al. (2018), A personalized and platform-independent behavior control system for social robots in therapy: development and applications

Bibliography - Cao, H. L., Van de Perre, G., Kennedy, J., Senft, E., Esteban, P. G., De Beir, A., Simut, R., Belpaeme, T., Lefeber, D., and Vanderborght, B. A personalized and platform-independent behavior control system for social robots in therapy: development and applications. *To be published in IEEE Transactions on Cognitive and Developmental Systems*.

Abstract - Social robots have been proven beneficial in different types of health care interventions. An ongoing trend is to develop (semi-)autonomous socially assistive robotic systems in health care context to improve the level of autonomy and reduce human workload. This paper presents a behavior control system for social robots in therapies with a focus on personalization and platform-independence. This system architecture provides the robot an ability to behave as a personable character, which behaviors are adapted to user profiles and responses during the human-robot interaction. Robot behaviors are designed at abstract levels and can be transferred to different social robot platforms. We adopt the component-based software engineering approach to implement our proposed architecture to allow for the replaceability and reusability of the developed components. We introduce three different experimental scenarios to validate the usability of our system. Results show that the system is potentially applicable to different therapies and social robots. With the component-based approach, the system can serve as a basic framework for researchers to customize and expand the system for their targeted health care applications.

Relation to WP - This work directly contributes to Task T6.3, and the general organisation of the other systems within WP6.

Period 1

When is it better to give up?

Towards Autonomous Action Selection for Robot Assisted ASD Therapy

Emmanuel Senft, Paul Baxter, James Kennedy, Tony Belpaeme
Centre for Robotics and Neural Systems, Cognition Institute
Plymouth University, U.K.
{emmanuel.senft, paul.baxter, james.kennedy, tony.belpaeme}@plymouth.ac.uk

ABSTRACT

Robot Assisted Therapy (RAT) for children with ASD has found promising applications. In this paper, we outline an autonomous action selection mechanism to extend current RAT approaches. This will include the ability to revert control of the therapeutic intervention to the supervising therapist. We suggest that in order to maintain the goals of therapy, sometimes it is better if the robot gives up.

Categories and Subject Descriptors: H.1.2 [Models and Principles]: User/Machine System

Keywords: Action selection, ASD, Cognitive Robotics, RAT, Social Robotics.

1. INTRODUCTION

Recent studies estimate that around 1.1% of the population in the UK and also in other European countries have Autism Spectrum Disorders (ASD). These people typically lack social skills normally expected in human interactions. Consequently, therapies have been designed to help children with ASD to improve their social abilities; these therapies can be enhanced by using robots [5].

However, due to the complexity of social interactions involving children, the majority of existing studies use the Wizard of Oz (WoZ) technique, where the robot is not autonomous but controlled by a human. Despite the clear advantages of this method, there are a number of reasons for researchers to move away from it, such as reducing the personnel required to use the robots, or improve the consistency of therapy [3, 6].

The present work is conducted within the DREAM project: a European project which aims to develop new Robot-Enhanced Therapy. We seek to develop the therapy robot's control system to enable supervised autonomous operation. A clinician will set the therapeutic goal for the session, from which the robot should be able to decide by itself which actions to execute, under explicit supervision. Rather than maintaining autonomy, we argue that allowing the robot to revert control to the therapist when appropriate would improve both the interaction and the therapeutic outcome.

HRI'15 Extended Abstracts, March 2–5, 2015, Portland, OR, USA.
ACM 978-1-4503-3318-4/15/03.
http://dx.doi.org/10.1145/2701973.2702715.

Figure 1: The Aldebaran Nao was selected as the common robot platform, to facilitate consistency and reproducibility.

2. BACKGROUND

Different approaches of Robot Assisted Therapy (RAT) have been explored by researchers in the last two decades. In previous studies robot control was typically achieved in one of two ways: either fully tele-operated using the WoZ method, e.g. [4, 7] or fully autonomous, e.g. [1, 2]. For WoZ control, a hidden, manual manipulation of the robot allows the therapist to obtain exactly the desired behaviour and to adapt to unpredicted events. On the other hand, an autonomous robot requires lower load on a human operator and allows greater repeatability of behaviour, but requires to design a complex controller. As such, only reactive control schemes are used in prior work.

Several attempts have already been made to combine the flexibility offered by the WoZ method and the autonomy and the consistency provided by autonomous operation. However, working with children with ASD presents additional challenges: the infrastructure required to perform the experiments is more extensive, it is hard to gather a population large enough to obtain statistically valid results, therapies take place over long periods of time and, as ASD is a spectrum, the children's behaviour can be more difficult to predict than neurotypical children.

To be able to use a robot as a therapeutic tool, we use a set of interaction scripts that determine the interaction between the child and the robot. These scripts are defined and selected by a therapist according to the goals of the current session and describe a clear, serial interaction where both the robot's and the expected child's actions are specified.

However, as we are working with children with ASD, it is unlikely that the script will be completely adhered to. The robot needs to be able to react to unpredicted actions to either return to the script or find alternate means of continuing the interaction. An autonomous action selection mechanism must therefore be able to cope with unplanned events whilst maintaining the therapeutic goals for the current session. A general description of the context in which the action selection mechanism should operate is presented in Figure 2.

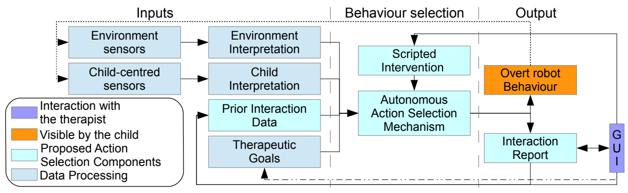


Figure 2: Context for the action selection mechanism.

Despite this, in some cases, the best action to select may be to stop the interaction and request help from the therapist. Contrary to the current approach, where the clinician must detect a problem and stop the interaction themselves, we aim to allow the robot to autonomously decide to refer to a human when this is appropriate.

3. APPROACH AND METHODOLOGY

The requirement for help for the robot may arise for a number of reasons. Firstly, there could be physical danger for the robot or the child. Some movements required by the script could harm the child if he is too close, requiring intervention if automated attempts to prevent collision fail.

Secondly, children could also react strongly to some specific actions, and force the therapist to stop the interaction. The robot could identify these actions and require help if one of them is requested for execution, thereby adjusting its level of autonomy. As the robot will be use in therapies, every action has to be carefully therapy-oriented and some actions could be defined that require approval before each execution.

Finally, the interaction could also fail (e.g. child no longer engages with the robot), where the robot does not have the competencies to pursue the interaction. In this case, therapist intervention would be required. If the therapist and the robot behave consistently in this context, the interaction may be more effective in terms of therapy.

3.1 Action Selection Mechanism

Based on previous studies about action selection in robotics, we have identified two broad approaches which could enable control to revert to the therapist. The first one is using a rule-based mechanism: as soon as a specific state is reached the therapist is consulted. An example is using the child's engagement in the interaction as a homeostatic variable: as soon as the implication goes outside a certain region, the interaction is stopped.

Another possibility is to use a predictive mechanism. Based on its previous interactions with this specific child or also with other children, the robot could have a model of what reaction is expected from its actions, and use it to predict the consequences of stopping the interaction versus continuing with its behaviour.

3.2 Evaluation methodology

To test our approach, we will use our algorithm in real session both with neurotypical children and ones with ASD in three scenarios: turn taking, imitation, and joint attention. If the robot detects a case where it has to revert the control to the therapist, it will broadcast a message describing what action should have been executed and why it stopped. The therapist will have the opportunity to execute the action, select another one, intervene in the interaction, or stop the session. We will use the therapist's action after control reverting and the number of times a therapist has to interrupt the interaction without a robot's prompt to evaluate the efficiency of the action selection mechanism.

4. DISCUSSION

Even when triggered by the robot, an unplanned human intervention in the interaction may have consequences on the child, the robot, and the therapist. For example, allowing autonomous failure detection, the robot can learn about it, and find itself a way to avoid the same state in the future.

Concerning the child, even if the session stops before an important problem, the emotional impact of interrupting the current interaction need to be taken into account. As children are sensitive, it is important to think carefully about the way to communicate the robot's failure to the child. Should the information about the interruption come from the robot? Should the therapist explain to the child what happened to the robot? We have no general solution yet, and the solutions may depend on individual characteristics. These questions have to be addressed based on data from empirical studies and collaboration with therapists.

5. CONCLUSIONS

In this paper we propose an approach to RAT for children with ASD: allowing a robot to voluntarily interrupt the interaction with a child and request help from a therapist. We outlined our motivations for this behaviour and presented possible consequences and questions to be resolved. The proposal is that autonomous action selection supports RAT because it reduces the workload on therapists, and improves its consistency.

6. ACKNOWLEDGMENTS

This work is supported by the EU FP7 DREAM project (grant 611391).

7. REFERENCES

[1] K. Dautenhahn. Robots as social actors: Aurora and the case of autism. *Proc. CT99*, (3), 1999.

[2] D. Feil-Seifer and M. Mataric. B3IA: A control architecture for autonomous robot-assisted behavior intervention for children with Autism Spectrum Disorders. *RO-MAN 2008.*, 2008.

[3] L. Riek. Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. *Journal of Human-Robot Interaction*, 1(1):119–136, Aug. 2012.

[4] B. Robins, K. Dautenhahn, R. T. Boekhorst, and A. Billard. Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills? *Universal Access in the Information Society*, 4(2):105–120, July 2005.

[5] B. Robins, K. Dautenhahn, and P. Dickerson. From Isolation to Communication: A Case Study Evaluation of Robot Assisted Play for Children with Autism with a Minimally Expressive Humanoid Robot. *Conferences on Advances in Computer-Human Interactions*, pages 205–211, Feb. 2009.

[6] S. Thill, C. A. Pop, T. Belpaeme, T. Ziemke, and B. Vanderborght. Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. *Paladyn*, 3(4):209–217, Apr. 2013.

[7] B. Vanderborght, R. Simut, and J. Saldien. Using the social robot probo as a social story telling agent for children with ASD. *Interaction Studies*, 110(Tager 2001), 2012.

Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Project No. 611391

DREAM
Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

TECHNICAL REPORT
Organisation of Cognitive Control and Robot Behaviour

Date: **02/02/2015**

Technical report lead partner: **Plymouth University**

Primary Author: **P. Baxter**

Revision: **2.2**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Service)	**PP**
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Summary

The purpose of this technical report is to summarise the motivations and constraints underlying the cognitive control structures, and to outline an organisation of these sub-systems. This is a proposal only; this document is intended to be a working one, to be updated as required during development. This version of the report is based primarily on the discussions that took place in Brussels (23/01/15).

Principal Contributors

The main authors of this document are as follows (in alphabetical order).

Paul Baxter, Plymouth University
Tony Belpaeme, Plymouth University
Hoang-Long Cao, VUB
Albert De Beir, VUB
Pablo Gomez, VUB
Emmanuel Senft, Plymouth University
Greet Van de Perre, VUB
Bram Vanderborght, VUB

Revision History

Version 1.0 (P.B. 21-01-2015)
Initial outline of ideas for the DREAM supervised autonomous robot control.

Version 2.0 (P.B. 26-01-2015)
Updated after discussions during Brussels meeting 23rd Jan 2015.

Version 2.1 (P.B. 02-02-2015)
Clarifications and updates following further discussion.

Version 2.2 (P.G. 27-02-2015)
Some modifications regarding priority system and Expression and Actuation subsystem functionality.

1 Aims and Constraints

An attempt is made in this section to formulate what the ideal (semi) autonomous system should conform to in terms of both clinical outcomes (i.e. the requests from the psychologists to improve the outcomes of individual children through robot-assisted therapy) and potential research (where this does not conflict with the clinical objectives).

The primary goal of the work in WP6 is to provide robot behaviour to facilitate the Robot-Assisted Therapy, see [1]. The main visible outcome of this should be the ability of the robot to execute the evaluation and therapeutic scripts as defined by the therapists. Whilst this must be achieved to fulfil the aims of the project, there are a number of areas in which there would be a role for behavioural adaptation, learning, and autonomous decision making. These should not however conflict in any way with the therapeutic goals for any given interaction session - indeed, it is necessary to vary the degree of shared control between the autonomous behaviour and the wizard supervisory control if this is more appropriate for a given child and/or circumstance.

Primary among these is the high probability that the interaction (due to the behaviour of the child for example) will deviate from the script. This must be handled in a manner consistent with the therapy, to not upset the child, and possibly (depending on the context) trying to re-engage the child with the script. A range of strategies will be required to deal with these situations, depending on the individual child (his/her characteristics) and the actual context for the departure from the script. This behaviour is likely to require flexible action selection, and will therefore require substantial research effort.

A second reason is that the robot is to demonstrate social behaviour in a supervised autonomous manner (with the requirement that the supervisor may over-rule this autonomous social behaviour if required). Social behaviour requires behaviour that is adaptive to the interaction partner in a range of interaction modalities (e.g. movement and speech). The autonomous behaviour of the robot must therefore be responsive to this, in a manner that is not, and indeed can not, be predetermined in the script.

Thirdly, given the range of intervention scripts that have been defined, there is also a possible need to modify the relative difficulty of the task (and/or interaction) given the specific characteristics and performance of the interacting child. This would, for example, involve varying the number and type of social behavioural cues used, the complexity of the required motor behaviours to complete the task, and/or the number of steps in the task.

The interfaces of the cognitive controller (WP6) with the rest of the DREAM integrated system (WP's 4 and 5) have already been defined. The intention in providing this overview document is to show how the subsystems of WP6 fit together to determine the behaviour of the robot in therapy interactions: the context in which each subsystem must operate is thereby defined. Initially, the skeleton of this system will be implemented in the most straightforward manner possible (with simplified code implementations of full component functionality for example) to check that the system fulfils all the requirements. This skeleton can then be filled in with more appropriate functionality over the course of the project.

2 Overall Organisation

A general high level description of the robot control system is shown is figure 1. This basically describes how the autonomous controller is informed by three external sources: the child behaviour description, sensory information, current intervention script state, and input from a therapist (e.g.

emergency stop, not shown in diagram). Combining these sources, the autonomous controller should trigger as an output the appropriate sequence of action primitives to be performed (as well as some feedback via the WoZ GUI), which then gets executed on the robot.

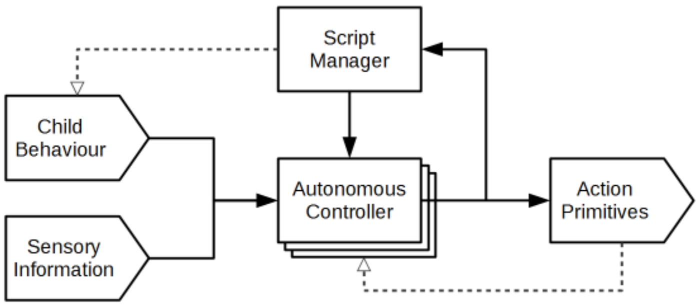


Figure 1: High level description of the robot control system. Child behaviour interpretation (WP5) and sensory information (WP4) provide the context for the autonomous action selection (as well as feedback from motor command execution), in combination with the particular intervention script being applied. The intervention script provides context for child behaviour interpretation.

The autonomous controller is composed of a number of sub-systems, as described in the DoW: Reactive, Attention, Deliberative, Self-Monitor and Expression and Actuation. These sub-systems interact, and must combine their suggested courses of actions to produce a coherent robot behaviour, in the context of constraints laid down by the therapist (for example, the script to be followed, types of behaviour not permissible for this particular child because of individual sensitivities, etc). An additional challenge is to ensure that the resulting system is independent of specific robot platform. As a result, we have formulated the following architecture describing how cognitive control informed by the therapy scripts is to be achieved (figure 2), an outcome of the WP6 meeting in Brussels (23/01/14). The following sections provide some further outline details of the main subsystems.

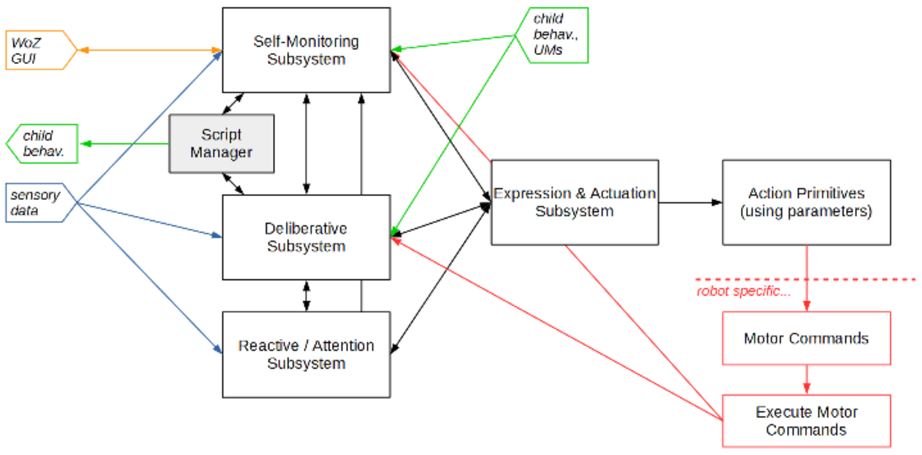


Figure 2: Description of the cognitive controller subsystems. The script manager is separate from, but tightly interacts with, the deliberative subsystem to enable the robot control system to generate appropriate social/interaction behaviour even in the absence of an explicit interaction script. UMs: User Models.

3 Reactive/Attention Subsystem

In the DoW, these are separated into two distinct subsystems. The reactive subsystem provides the general life-like behaviour of the robot (small motions, eye blinking, balancing, recovering from falls, 'pain' reactions, etc) in an as appropriate manner as possible (possibly requiring pilot studies to verify this). However, it should be possible to turn off these behaviours should the therapist deem it necessary for a particular child. This functionality is not envisaged to involve learning or adaptation. The attention subsystem is a "*combination of perceptual attention ... and attention emulation*". Making eventual use of saliency maps and habituation filters, this functionality will be guided by the deliberative subsystem.

We instead propose that these two subsystems be combined into a single component, due to the significantly overlapping technical systems required to fulfil the functions required. Both subsystems require access to features of the environment and interacting person(s) to respond appropriately (e.g. looking at a face or diverting attention to a loud noise somewhere in the environment). Managing this in a single component therefore seems a sensible choice so that functionality is not replicated. As planned in the DoW, it will be possible for the supervising therapist to switch off these functionalities if required for interaction with a particular child.

4 Deliberative Subsystem

A central aspect of the cognitive controller is the ability to follow intervention scripts as defined by the clinicians for both diagnosis and therapy. These scripts describe the high-level desired behaviour of the robot[1], and the expected reactions and behaviours of the child, in a defined order.

The decision was made to separate the script manager from the deliberative subsystem itself (fig 3). This decision was taken for a number of reasons. Firstly, it enables the cognitive control of the robot to be independent of the precise application domain - with the intention that the developments made would be more generally applicable within the field of social robotics, although the script-based behaviours remain a central part of the behaviour generation of the system. Secondly, it ensures that it would be possible to change the scripts in the future to alter their relative difficulty, by for example including further steps in the intervention, changing the type of intervention, or creating different activities, due to a modular design[2]. As a consequence of this, the deliberative subsystem is now primarily focussed on action selection considerations, making use of a range of algorithms and methodologies as will be explored in the coming years. Thirdly, this division of the script manager from the deliberative subsystem enables the system to generate coherent behaviour even if there is not a script active at a given moment. This could be useful for periods between the explicit intervention sessions for example, where the robot would then still be able to respond appropriately to environmental stimuli, if so desired by the therapists. These are consistent with the aims expressed within the WP6 DoW.

The script manager itself separates the logic necessary to manage progression through the script (by taking into account the available sensory feedback after actions for example) from the script itself. This makes it straightforward to add new scripts or modify existing scripts as required. This logic management could in the first instance be achieved using a Finite State Machine (FSM).

[1]These predefined robot behaviours differ from the the low-level motor control of the robot, as these may be mixed with other aspects of behaviour not specified explicitly in the high-level intervention script; e.g. the addition of attention to unexpected events in the environment.

[2]As noted above, these high-level scripts do not necessarily completely define the behaviour of the robot, and are distinct from any predefined robot motor control sequences that may be used, such as waving or nodding.

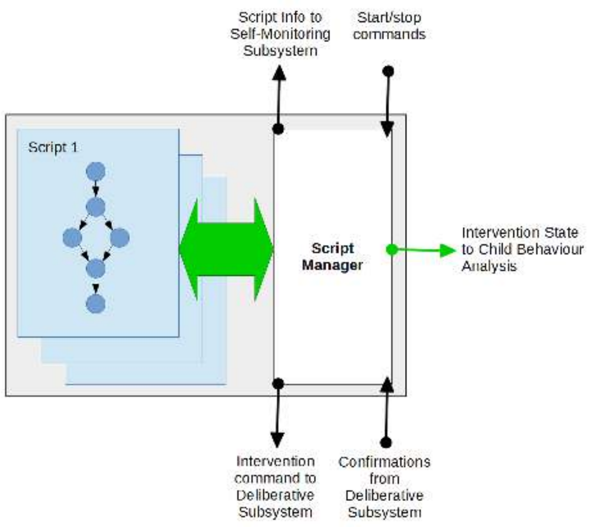

Script Info to Self-Monitoring Subsystem
Start/stop commands
Script 1
Script Manager
Intervention State to Child Behaviour Analysis
Intervention command to Deliberative Subsystem
Confirmations from Deliberative Subsystem

Figure 3: Overview of the script manager subsystem. The scripts are defined independently of the script manager, which is responsible for stepping through the script as appropriate and communicating with the other subsystems as required.

One possibility for the scripts is that each step in the script be defined as a 3-tuple of the form: *[existing_state, proposed_action, consequent_state]*. In this context, *existing_state* could be defined by default to be the *consequent_state* of the previous step. The *proposed_action* defines what action should be taken by the robot, and be one of the actions (or unique identifier thereof) defined in D1.2. The *consequent_state* defines what robot state should be expected (in terms of sensed state) if the *proposed_action* were successfully completed. This may be used by the script manager to determine if and when it is appropriate to move onto the next script step. These 3-tuples may initially be held in a plain text file to facilitate examination and modification by the clinical staff as required. This can be changed later to ease the process (for example by providing a drag-and-drop script construction GUI).

The deliberative subsystem is the primary locus of autonomous action selection in the cognitive controller (fig 2). This subsystem takes as input sensory data, child behaviour information, information on what step should be next executed from the therapy script, and higher-level direction from the wizard/self-monitoring subsystem. It then proposes what action should be taken next by the robot (this proposal is sent to the expression and actuation subsystem). In a normal script execution context, the deliberative subsystem is the primary driver of behaviour, which would typically propose the next script step.

There are however a number of circumstances in which this is not the most appropriate action to perform. For example, if the child is detected to have very low engagement with the task (as determined from the WP5 component/s, and/or information from WP4 sensory system saying the child is looking away for example), then it would be appropriate to attempt to re-engage the child with the robot/task prior to executing the next stage in the therapy script. In this case, the deliberative subsystem can choose to depart from the behaviour defined in the script, and instead propose a different behaviour.

5 Expression and Actuation Subsystem

The main functionality of this subsystem is to determine which combination of low-level actions the robot should execute next, and how these actions are to be performed. Suggestions for actions to take will come from three other subsystems: deliberative, reactive/attention, and self-monitoring, and the affective state generated by the deliberative subsystem, see left side of figure 4. Along with this, it is assumed that the supervising therapist, through the GUI, will determine (either beforehand or in real time) the aspects of robot behaviour that should be executed, from which relative priorities will be determined for the three subsystems. This covers for example whether external disturbances (a loud noise in the background, or the appearance of a new face) should be reacted to by the robot (by leaving the script for a while for example), or ignored (with the script rigidly adhered to). The Expression and Actuation subsystem will combine these sources of information in an appropriate manner, see Motion Mixer in figure 4, ensuring that the stability of the robot is maintained. For example, if a greeting wave is requested by the deliberative subsystem, and the reactive/attention subsystem wants to look at a face that has been detected, then the expression and actuation subsystem can combine the two by executing both (if the robot can remain stable by doing so). For a basic first step switches based on priority level could be used: i.e. if the script requests an action, execute it (and only it), but if there is no script action requested, then do what the reactive/attention subsystem proposes. However, the intention is to provide full behaviour mixing capabilities based on derived priorities from the therapists.

All this should be complemented by affective information, if this is available and appropriate to use. For example, the speed of motor execution could be related to arousal levels, or the choice of action sequence could be based on valence levels (if appropriate alternative sequences exist). This functionality will need to be switched on or off as required by the therapist based on child-specific considerations, and the relation to the therapy script (it may not appropriate to add emotional colouring to actions during the diagnosis procedure for example).

To approach such challenges, the first task should be to design a platform-independent representation of expressions. Different robots use the Facial Action Coding System (FACS) by Ekman and Friesen [2] to abstract away from the physical implementation of the robot face. FACS decomposes different human facial expressions in the activation of a series of Action Units (UA), which are the contraction or relaxation of one or more muscles. In a similar way, Body Action Units (BAU) will be defined together with a Body Action Coding System, where the different gestures are decomposed in the activation of BAUs. The BACS will point out the Action Units that need to be actuated for the generation of a desired gesture or body pose. This system avoids pre-programming of robot-dependent body poses and actions, which is relevant since humans are able to recognize actions and emotions from point light displays (so without body shape) [3].

The physical actuation of Action Units will depend on the morphology of the robot: a mapping will be needed between Action Units and physical actuators, this mapping will be specific to a robot platform and we will explore the possibility of learning this mapping. To translate this to the morphology of the robot, the Action Units need to be mapped to the degrees of freedom, and thus to the joints of the robot, see right side of figure 4.

A second task will be the categorisation of actions, comprised of temporal series of FACS and BACS, and the organisation in libraries that are accessible from the behaviour subsystems (Reactive, Attention and Deliberative). All actions for the different behaviours should be stored and expanded upon without the need to reprogram other subsystems.

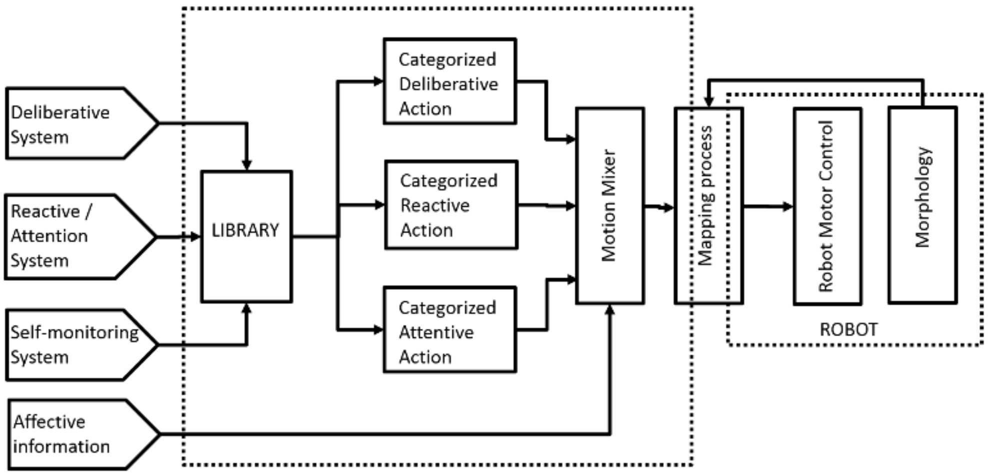


Figure 4: Overview of the Expression and Actuation subsystem. This subsystem receives inputs from several sources, categorizes them using the Library module and mixes them up to create a unique behavior. Such behavior is mapped into the joint configuration of the corresponding robot. This last process is done collaboratively between the subsystem and the robot.

6 Self-Monitoring Subsystem

The self-monitoring subsystem provides an oversight mechanism (or set of mechanisms) of the robot behaviour. It is intended to provide a check to prevent technical limits being exceeded (of the robot[3]), and to prevent any ethical boundaries being crossed. This subsystem should have some degree of autonomous behaviour, with the intention being that these checks be implemented in a set of predefined rules, with no role for learning within this subsystem.

During the discussions, it was proposed that the self-monitoring subsystem should also be integrated explicitly with the therapist GUI. In line with the principle of supervised autonomy established in the project, the therapist ("wizard") should be able to monitor the behaviour of the robot, and be able to intervene if necessary, either stopping the behaviour, modifying a behaviour, or setting an alternative behaviour. Having this oversight function go through the self-monitoring subsystem seems to be a reasonable solution. By specifying the required priorities for each subsystem depending on the needs of the therapy, and using the "alarm signals", the supervising therapist can stop the robot or modify its behaviour as desired.

Regarding both the autonomous oversight functions and the supervised actions, there are a number of issues that require exploration and further definition over the course of the project. One thing is how the robot should behave, and what feedback it should give to the child, should something go wrong. Possible alternatives are described in the DoW.

7 Action Primitives and Motor Execution

The behavioural functions of the action primitives required for completion of the therapy scripts have been defined. The execution of these is handled in a number of steps, as outlined in the "Robot

[3]This is mentioned here as it is listed in the DoW as a competence of the self-monitoring subsystem, however, this functionality is at least partially implemented in the low-level motor control system of the robot: see section 7.

Low-Level Motor Control" technical report. This provides an interface between the control system (handled in a Yarp-based system) and the API of the robot hardware (Naoqi in the case of the Nao). The purpose is both to provide a bridge between the two systems, and to provide information to behaviour planning and supervisory oversight regarding the progress of motor command execution, including why a fail occurs if it does. This can be used to inform future action selection for example (by providing feedback for learning).

In addition to this low-level control system, there is the possibility that hardware abstraction can be handled automatically: i.e. that motor commands at the joint level can be determined automatically for different robot embodiments, without having to manually encode each specific action.

8 Other aspects of the Cognitive Control System

8.1 User Models

One functionality that was not explicitly defined in the proposed architecture, WP6, or indeed elsewhere in the project, is some source of information on the child. This information could encompass personal identification and preference information that could be used in conversations (e.g. name, age, favourite colour, etc), and possibly also ASD diagnosis information (perhaps as emerging from the diagnosis interaction scripts).

These user models would enable, for example, inform learning mechanisms (within the deliberative subsystem for example) to link behaviours and outcomes with specific characteristics of individuals (indicated in figure 2). This information need only be uniquely identifiable rather than linked to a specific child - although the extent to which this can be done needs to be assessed in light of ethics considerations (cf. WP7 ethics manual draft, December 2014). Technically, in the first instance, a unique impersonal identifier may be used to represent an individual child. Where this information should reside, how it should be stored, etc, has not been decided. It would probably be useful however to coordinate this system with WP5, as the child behaviour interpretation methods may find such information useful too to be able to provide more personalised characterisations of engagement and performance for example.

References

[1] S. Thill, C. A. Pop, T. Belpaeme, T. Ziemke, and B.ram Vanderborght. Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. *Paladyn*, 3(4):209–217, 2012.

[2] Ekman P and Friesen W. *Facial Action Coding System*. Consulting Psychologists Press, 1978.

[3] A. P. Atkinson, W. H. Dittrich, A. J. Gemmell, A. W. Young, et al. Emotion perception from dynamic and static body expressions in point-light and full-light displays. *Perception-London*, 33(6):717–746, 2004.

Development of Robot-enhanced Therapy for
Children with Autism Spectrum Disorders

Project No. 611391

DREAM
Development of Robot-enhanced Therapy for
Children with Autism Spectrum Disorders

TECHNICAL REPORT
Sandtray Wizard-of-Oz System for Turn-taking Intervention

Date: **23/03/2015**

Technical report lead partner: **Plymouth University**

Primary Author: **P. Baxter**

Revision: **1.0**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Service)	**PP**
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Summary

In this technical report we describe the software organisation of the Sandtray system created for the turn-taking diagnosis/intervention interactions. This system is based on the organisation defined by the WP3 software engineering standards, although at the moment does not fit into the rest of the DREAM system:this was to facilitate ease of setup and launch for the end-user (i.e. minimal installation, and no compilation required). The WoZ system provides a GUI from which the therapist can control the robot behaviour in the turn-taking task, and logs of the interaction are automatically stored for retrospective analysis.

Principal Contributors

The main authors of this document are as follows (in alphabetical order).

Paul Baxter, Plymouth University
Emmanuel Senft, Plymouth University

Revision History

Version 1.0 (P.B. 23-03-2015)
First version

1 Turn-taking Diagnosis/Intervention

The turn-taking diagnosis and intervention tasks are specified in D1.1. In the context of the Sandtray device, this is a task in which the child and the robot take turns in moving an image displayed on the touchscreen to one of two target locations. The robot plays the game with the child, and provides verbal indications of whose turn it is. Figure 1 shows the constructed Sandtray device. The task may be varied by changing the images to be sorted (e.g. characters, emotions, etc).

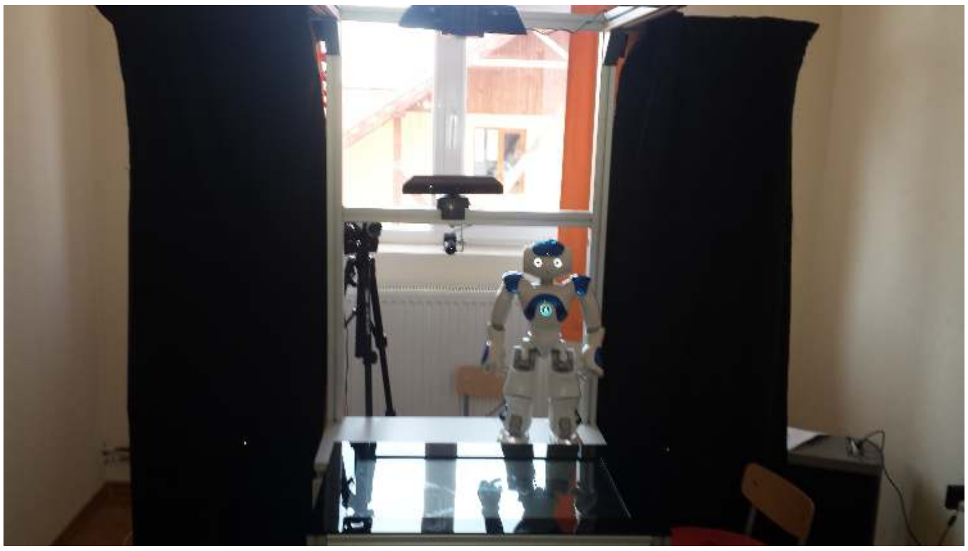


Figure 1: The final setup of the Sandtray with robot shown with the intervention table. The robot should ideally be centred in front of the screen to facilitate the interaction, and to ensure that the pointing behaviours are accurate.

In this first version of the task, a therapist provides all of the decisions regarding the behaviour of the robot: it is a full "Wizard-of-Oz" (WoZ) interaction. This enables experiments to take place before a full autonomous system has been implemented. Despite this, all of the components of the WoZ system are implemented using the software engineering standards established in WP3.

However, being designed as a standalone system with the intention of minimising the learning curve necessary for deployment and use, these components are not yet implemented in the context of the complete DREAM system architecture. The purpose of this technical report is to describe the system, its components and its use.

2 Wizard-of-Oz System

2.1 Dependencies

The intention is that this Sandtray WoZ system can be deployed easily and without need of recompilation on the target PC. There is only one dependency that requires installation before use: Yarp v2.3.63. This is needed in order to start the system. Instructions for installing this can be found on the project wiki ("Software Installation Guide").

Additional setup instructions are as follows:

1. Copy folder "dreamSandtray-release" into C: drive

2. Add ”C:/dreamSandtray-release/working/bin” to PATH (user) environment variable: this allows the precompiled executables to be found
3. Change the DREAM_ROOT environment variable to point to ”C:/dreamSandtray-release/”: if the full DREAM system is to be installed afterwards, be sure to change this back!
4. In file ”/working/config/config.ini”, change the IP address to that of the robot
5. In this same file, set the relative position of the robot to the touchscreen: this will be different depending on whether the robot is standing or crouching when playing the game; point of origin on robot is the neck joint
6. On sandtray machine, change the IP address to that of the Host PC (in ”settings.ini”)

2.2 Deployment

To launch the system, the following two steps are required:

1. On host PC: in directory ”/run” double click on ”run-sandtray.bat”: this will launch the system: if successful, the robot will readjust the position of its arms up and a GUI will appear; before the first launch, it may be useful to disable any firewalls, or else enable firewall access permissions for the four components that are launched; if launch fails, then close all windows and try again (if naoInterface.exe persists in failing to launch, then try restarting the robot)
2. After the robot has set its position and the GUI has appeared, then start the sandtray GameEngine: if successful, then will see confirmation on the terminal; if fails then check that the sandtray is on the same wireless network as the robot and the hot PC, and that the correct IP addressed have been set in the config files
3. Full interaction logs are automatically created: see section 2.4 below

2.3 Behaviour Control

Assuming a successful system launch, three terminal windows are spawned, and the control GUI (figure 2). This minimalistic GUI provides some feedback from the robot (success, or otherwise, of the desired motor commands - see low-level robot motor control technical report) and the Sandtray GameEngine (such as child successful or unsuccessful move, and the same for the robot, library change confirmation, etc).

The GUI provides two types of command. The first is presented in the middle column of buttons. These provide the verbal feedback that the robot can provide the child, such as a brief introduction, indication of whose turn it is, and feedback on the outcome of the categorisation. Additionally, a “rest” button is provided that moves the robot to a neutral position and switches off the motors. This provides a first level safety feature: if there is any chance of injury to child or damage to robot during movements, then this button will disable the robot. This is a basic feature at the moment, which will be extended in due course (see below).

The second type of command allows manipulation of the Sandtray GameEngine. “Good Move” and “Bad Move” make the robot perform the relevant classification of the image on screen (the image moves across the screen with the robot hand tracking it, thus providing the illusion of robot control). The “New lib” button presents a new image library on the touchscreen, and the “Reset lib” button presents the current image library on the screen again. “Shutdown” closes the GameEngine remotely.

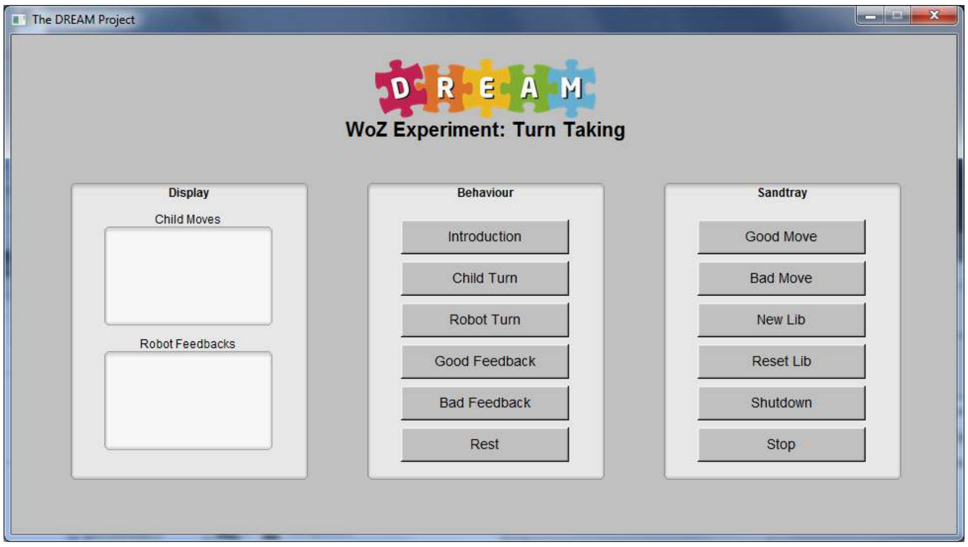


Figure 2: The Sandtray Turn-taking task control GUI. The windows on the left provide text feedback from the Sandtray and robot; the column of buttons in the middle provides robot verbal feedback; and the right-hand column of buttons provides robot behaviour control and GameEngine controls.

The “Stop” button is intended to provide a second level of safety control for the controlling therapist by enabling the interruption of robot movements that are currently under way. This functionality is under development, as part of the continued work on the low-level motor control (please refer to the relevant technical report).

2.4 Automated Interaction Logging

In addition to providing control, the Sandtray WoZ system also provides automated logging of all of the important events in the interaction that are detected by the Sandtray, or that are initiated by the therapist. These files are created automatically when the system is run: a filename with a timestamp is created in the directory where the batch file is located (where the system is launched from) - see section 2.2 for details.

This log file contains all events initiated through the GUI, events detected by the Sandtray, and motor command execution feedback. All events are logged with a timestamp (one second resolution), an identifier, and a description in comma-separated lines of a plain-text file. This facilitates parsing at a later time for retrospective analysis, replaying the interaction for diagnosis purposes, etc. This method of interaction logging will be extended as more data becomes available to the system.

3 Component Descriptions

The Sandtray Turn-taking WoZ system is comprised of four main components (in addition to the two external devices): sandtrayController, sandtrayServer, naoInterface and a GUI. sandtrayServer provides a communication interface between the yarp-based system and the Sandtray GameEngine. sandtrayController provides the primary coordinating role between the different components, containing some functionality that will soon be split between the Deliberative and Expression & Actuation sub-systems. The GUI provides the therapist-facing controls (described above). The naoInterface component

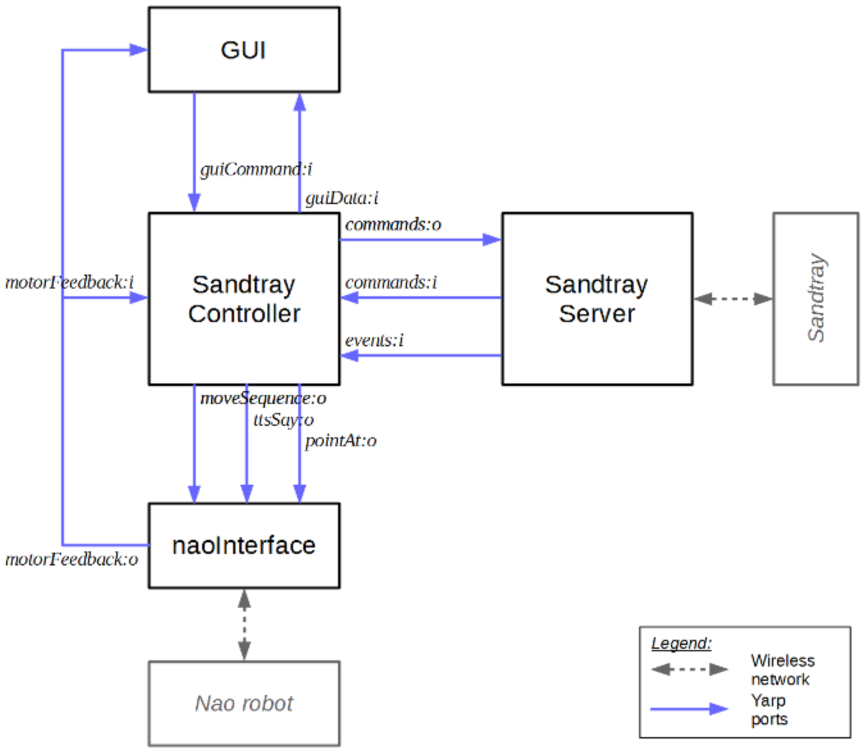


Figure 3: Components in the Sandtray WoZ control system. Communication between components is handled by yarp: shown are the component port names for the sandtrayController component (not all port names shown for clarity). Communication with the Sandtray device and Nao robot is handled over wireless networks outside of yarp.

3.1 gui component

This component is the interface used by the therapist to send request to the system. The current design is shown in figure 2. The description of the buttons may be found above.

The FLTK library is used to generate the GUI, the code is adapted from David Vernon's protoGUI, so the GUIutilities library is needed to generate it. The update is performed with a while(isStopping), could be improved with a rated thread or something else. The organisation of files is the same as described in YarpGenerator tech report: the 4 files needed for yarp, a yarpInterface class, a controller class where the main code is, and the display class containing all the information required for designing and running the GUI. Currently each button in display is linked to a static callback redirecting to a callback in display.cpp, which calls the appropriate function in controller.cpp, this could probably be simpler.

This component also uses the childName variable in the main config.ini to generate the string send to load the introduction behaviour to make it personal to each child. So one introduction behaviour need to be exported for each different child and should use the following convention: introduction-childName , e.g. "introductionGeorges".

3.2 sandtrayController component

The sandtrayController component is the main and central component. This is the one managing the interaction between the other components. It has input and output ports from/to all the other components. It gets the command from the gui and relay them to the server or the naoInterface and act in accord with the feedback obtained.

As this is the central component, the logging is done from here. The file structure used is the classical YarpGenerated one, with an additional logger class (.cpp and .h) to be able to generate logging files. Currently the logging is disabled but the file is still created, this could be changed.

This file does some string processing, so new methods have been added to handle that. It also integrate a Link class allowing to Link an action to a previous if a certain state is reached. The linking process is described in depth below.

This component is also responsible of the bezier calculation and the transformation from pixel coordinates to robot's ones. All the variables are defined in sandtrayControllerController.h except the horizontal and vertical distances between the robot and the screen which are defined in the global config.ini file in centimetre.

3.3 sandtrayServer component

This component is responsible of the interface between Yarp and the Sandtray, its main functionality is to transform Yarp message into sockets one and vice versa.

The global structure is the same as the others components. In addition, two class are define in the controller file: SandtrayControl and SandtrayEvent. The first one is a classical class which handle the communication along the *command socket port* with the Sandtray, which is used to transmit information from the controller to the Sandtray, and receive the answers. The communication is only triggered by the SandtrayController, and handled by Yarp callbacks, so no while loop is required.

The other one manges the communication along the *event socket port* and in that case, the discussion is triggered by the Sandtray. In the current implementation, this class wait using a while(isStopping) to get information from the Sandtray, so yarp::os::Thread is inherited from.

A limit of the current implementation is that the Sandtray needs to be started after the server component.

3.4 naoInterface component

This component implements the action primitives as defined in the deliverables. The idea is to process commands to the robot, check if the command is possible, avoid conflicts and provide a way to stack commands.

Currently the commands allowed are: pointAt, say, execute a behaviour. In addition to the classical files, an action class has been added, which is used to serialise the different type of actions: movement, text to speech or behaviour. All the tests performed are implemented here. The other new class is naoInterfaceModule, this provide the interface with Nao, with all the functions robot specifics. In principle the code should be easily modified to adapt to another robot, only this class and the refFeature map used to avoid conflicts should be modified. More information is available in the Low Level Control tech report.

Comparing to the previous implementation, multiple features have been added, and few changes have been implemented in the flow, no completion check is performed when an action is received. Multiple action cannot be successfully check anymore, like tts or behaviour and the movements can be composed of multiple points, so a previous completion check makes no sense. The ISTARTED message has been also overloaded, now it can be followed by a " sub id" with id a negative number used to synchronyse the controller and the naoInterface.

The component now integrate a pointAt function which does a simple inverse kinematic to transform x, y, z coordinate in joint space. It selects first an arm depending on how many points are more on the right side or the left one, then compute the joint angles. This function can take multiple points with the parameter x, y, z and t, and only a subset of the movement can be executed.

3.5 Detailed description

3.5.1 Action linking and synchronisation between the naoInterface and sandtrayController

In the previous implementation, each action required by a component and executed by the naoInterface was identified by a unique id. However this id was provided by the naoInterface and when a component required an action, it had no way to know the id of this exact action, multiple components can require action at the same time. We wanted a way to serialise two or more action. To do so, when a component request an action it needs to know what is this action's id to be able to command a new action when this one is finished depending on its result.

To allow this synchronisation, a system of subId was implemented. When a action is requested by a port, it can be overloaded by adding a new parameter at the end of the parameter list to assign a subId to an action. The subId is a negative int allowing to know what is the id (in naoInterface) assigned to a defined action. In naoInterface, this subId is extracted and added to the action and when the action is started, the subId is added at the end of the ISSTARTED message to allow the other component to know what is the id of the action with a specific subId. Currently, this subId is displayed and used only with the ISSTARTED message. In the future it should be important to use it also in case of direct failure.

In our setup, only the SandtrayController requires the use of synchronised actions. This is done via the Link class. Defined in sandtrayControllerController.h, this class define an action: id and subId, a linking condition ($\_status$), a consequence and a pointer to another link if needed. This allow us to perform a specific action when a defined action is finished, and cascade it if required. Currently, after the completion of and action, we can request a new tts, movement or behaviour, the type of action is stored in $\_consequence$, the parameters of this action are stored in the two strings $\_param1$ and $\_param2$ or $\_move$.

To create a Link, we need to know what is the subId the causal action, the one which will trigger the Link, later, this subId will be associated to a proper id (the same as used by naoInterface). If we want to chain multiple action, we need first to define all our actions with the proper subId (defined in the port parameters), and then create the different Links with the subId assigned to their causal action and giving them the parameters for the consequence action. Then, each consequent Link can be added to the previous Link in $\_nextLink$, and the first Link can be added to the $\_currentLink$ vector in the controller class.

A subId variable is stored in the controller, and is decremented once for each Link, and this is used to assign subId to an action.

Currently, when an action is a success, the $\_nextLink$ is added to the $\_currentLink$ vector, and if it is a faillure, it is destroyed. However there is a memory leak, if we have a chain of links, and the first one is failed, only the next one is deleted, not all the chain...

3.5.2 Flow when move requested

The flow is as defined in the previous the design. The added part is the following: the sandtray returns coordinates for the bezier move in pixel, this is relayed to the controller via the server. Then the controller extracts the bezier points and transform the coordinate in robot space.

A first pointAt is prepared with all the bezier points plus a one at the end of the vector to signify that only the first movement should be perform. The current subId is added also at the end of the vector. Then a link is create with this subId and the success status to execute the full movement and the boolean $\_synchronizedSandtray$ is set to 1 meaning that the command need to be sent to the Sandtray also. The subId variable is decremented and added to the movement after a 5, meaning that we want to perform the full movement with the new subId. A last link is created with this subId and the behaviour *init* in parameter, to reset the robot. This last link (behaviour) is added in the $\_nextLink$ pointer of the movement Link, which is added in $\_currentLink$.

Finally the first movement (initial pointAt) is sent to naoInterface. There the subId is extracted, and sent with the ISSTARTED message to the controller, which assign this id to the first link. When the pointing is completed, the first link is triggered: we send to the naoInterface and the Sandtray the request to move and we add the second link to the $\_currentLinks$. Once the full movement is received by the naoInterface, it send to the controller a new ISSTARTED message with the new id and subId. Similarly this is handled by the controller to assign the right id to the second link. And finally when the action is completed, the last link is triggered to reset the robot.

4 Sandtray GameEngine Management

4.1 Sandtray GameEngine

The Sandtray sorting task software is already loaded onto the touchscreen, with some sample image libraries (see next section for details of these). There is an executable in a folder on the desktop, and a shortcut on the desktop itself. Double-clicking this starts the programme in full-screen mode. This automatically loads all available image libraries, and cycles through them in numerical/alphabetical order.

There is only one keyboard shortcut that is required for operating the sorting game: the ESC key exits the programme and returns to the desktop. We advise not to do this in front of the child unless absolutely required - we recommend having the sorting game on the screen when the child enters the room.

Further to this there are two on-screen buttons that can be used to control the sorting game. The circular arrow leads to a reset of the current library (i.e. the same images are displayed again). The other icon (a sun with two + symbols...) indicates a switch to the next library. With further integration with the rest of the system, switching to specific image libraries will be possible at a later date. These on-screen buttons can be disabled if required in the config file (see below).

The software has a number of options that can be modified in a configuration file, located in "settings.ini". There should not be any need to change the paths in this file. The "robotiP" field should be set to the IP address of the host PC being used. The "game" options control aspects of the GUI, as described below (table 1).

Table 1: Sandtray sorting game options, can be found in settings.ini

Setting	Description	Default
LadderRungs	The number of fields visible for classified images	5
LadderWidth	Width of fields (in pixels)	50
ReserveTests	(do not modify) deprecated functionality	false
TestLibStart	(do not modify) deprecated functionality	-1
UseButtons	Whether to display control buttons on screen	true
ShowFeedback	Whether to display feedback on classifications	true
OneAtATime	Display images one at a time on screen	true
CentreImages	Display images in the centre of screen (random otherwise)	true

4.2 Sandtray Image Library Management

The Sandtray sorting game is based on *image libraries*. Each image library has a unique identifier, and contains a set of images (in .png format). Each library is in a separate folder in /images/libraries/. The name of each library folder follows a specific format:

libxx_name_opt

Where *xx* is an integer (e.g. *01*), *name* is a string identifier (e.g. *carbohydrates*), and *opt* is an optional extension that provides additional information.

For the purposes of the present experiments (turn-taking), a two category sorting task is assumed (one, two of four category sorting tasks are possible). Each library is a folder containing a number of images. There are no explicit limitations on the number of images per library, although resource issues (all images are pre-loaded at run-time) may mean that splitting up large libraries into parts is necessary. Two category images must be defined. These can be any image of the same type and size as standard images, but with the following naming convention:

catx_name.png

Where *x* is either *A* or *B*, and *name* is a string identifier of the category. For example, the following two filenames may be used to define two categories: *catA_high.png* or *catB_low.png*.

Standard images (in .png format) have to be assigned to one of the two categories (A or B), and their filenames should follow a similar but slightly different naming convention:

xNN_name.png

Where *x* is the category to which the image belongs (as defined for the category images above), *NN* is a unique image integer identifier (e.g. *01*), and *name* is a string identifier that provides some descriptive information. Examples of suitable image file names are *A01_chicken.png* or *C11_lasagne.png*.

Please see the example image libraries supplied with the Sandtray for indications of appropriate image size and resolution (as tested with hundreds of UK primary school children in the approximate age range of six to nine): please ensure that the images are not too large, as this will cause problems.

The string identifiers (and optional extra information) attached to the names of the libraries and the images are not used at the moment. However, they will be used to enable the robot to refer to specific objects, and specific properties of those objects as the autonomous system is developed. It is therefore worth adding this information to the folder and file names during the creation process.

Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Project No. 611391

DREAM
Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

TECHNICAL REPORT Manual for the use of Choregraphe boxes in Wizard of Oz experiments

Date: **11/02/2015**

Technical report lead partner: **Vrije Universiteit Brussel**

Primary Author: **Pablo Gómez**

Revision: **1.0**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Service)	**PP**
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Summary

The purpose of this manual is to help UBB in the development of the Wizard of Oz experiments within Work Package 2. Both PLYM and VUB have collaborated to develop the corresponding modules in Choregraphe. This manual aims at being a reference point to ease the habituation of the therapists to the software.

Principal Contributors

The main authors of this document are as follows (in alphabetical order).

Pablo Gómez, Vrije Universiteit Brussel

Revision History

Version 1.0 (P.G. 11-02-2015)
First draft

1 How to use this manual

In order to make the software the more understandable as possible we have divided each of the scenarios in Deliverable D1.1 into different project files. So that, in between breaks, the forthcoming scenario has to be loaded. The procedure to follow to load a project will be explained in Section 2. Once opened, each project file includes a set of boxes to make the robot perform an action or say something. Each of these boxes is identified by a self-explaining name. To make easier the management of the boxes during the experiment we have not grouped them chronologically but into families of similar behaviors, i.e. all text boxes are close to each other, actions regarding pointing are together, and so on. In addition, a representative image identifies each family of behaviors.

At the end of the manual some notes about the limitations of this software are included. We must not forget that it is following a Wizard of Oz methodology so most of the work to follow the script is required from the therapists.

2 How the software is organized

As mentioned above each of the scenarios in the deliverable D1.1 is one project file named after the name included in D1.1.

2.1 How to load a project file

In order to load a project file, once Choregraphe has been opened, click on *File*, *Open project...* and select the one you prefer from the list of projects, and click *Open*. Then, you will see something like in Figure 1.

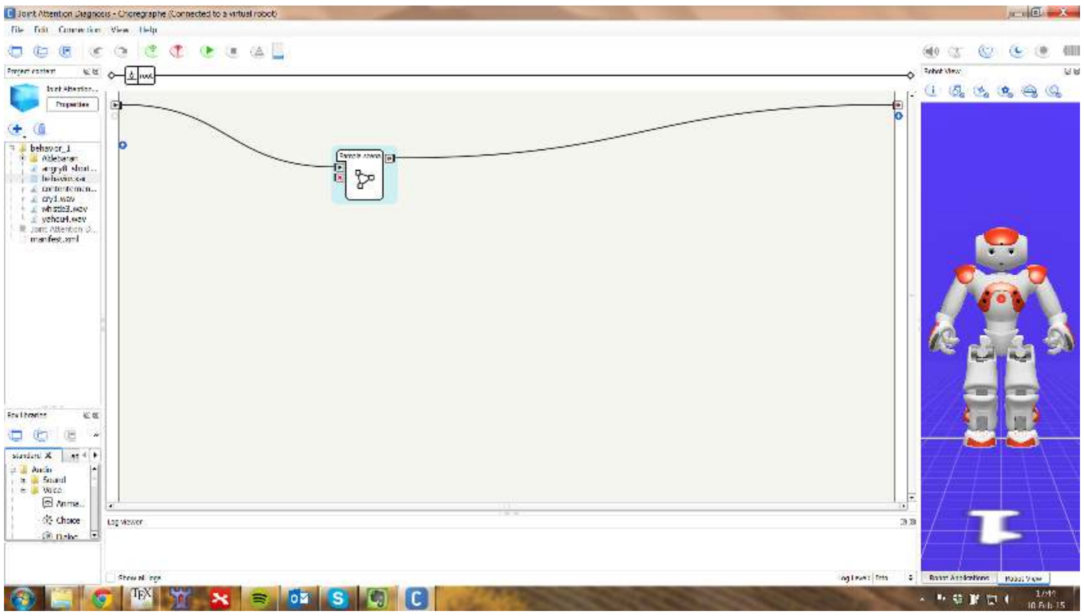

Joint Attention Diagnosis - Choregraphe (Connected to a virtual robot)
File Edit Connection View Help
Project content
Properties
behavior_1
Box libraries
Log viewer
Robot View

Figure 1: Opening a project file.

By double-clicking on the box that appears, Figure 2 will show up. You need to click on the *Interaction* block shown in blue. All the required behaviors for the corresponding scenario will be shown.

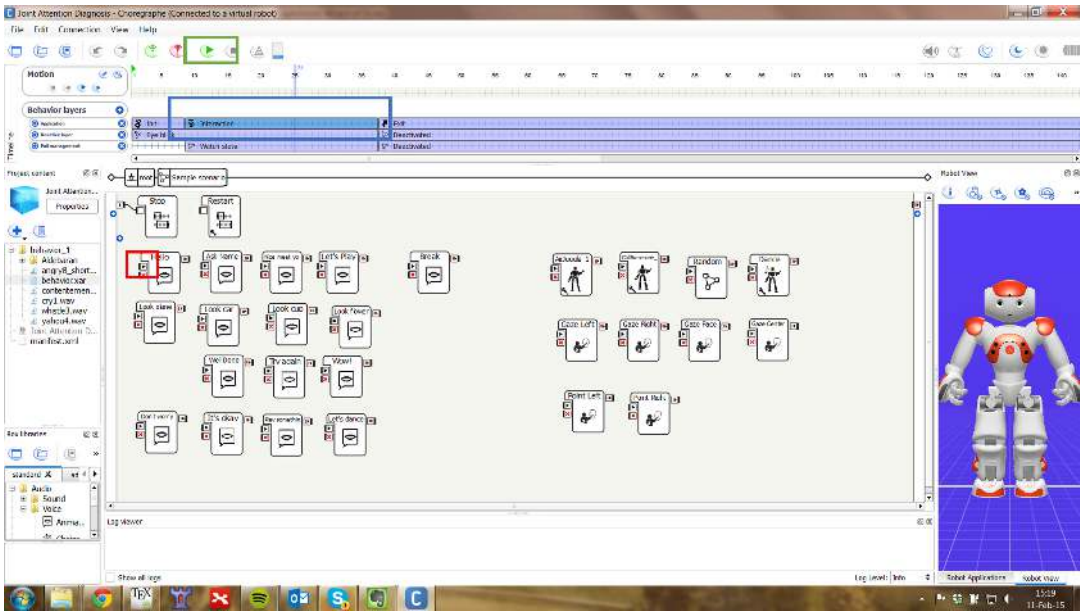


Figure 2: Accessing the behaviors within a project file. To activate each behavior click twice on the small *Play* button on the left of each of them.

All behaviors include a name to briefly identify what they do. They are grouped according to their functionality. There is no chronological order.

We have included a *Restart* button to use in case something went wrong to initialize all the variables and start the intervention over. There is a *Stop* button which is required by Choregraphe, you don't need to pay attention to it.

2.2 How to run a project file and stop it

In order to run a project file you just need to click on the *Play* button, in green in Figure 2. Once it is running, you can activate each behavior double-clicking on the small *Play* button each behavior includes, in red in Figure 2. When such behavior is activated the button changes its color to green for a few seconds. If it didn't happen the behavior was not correctly activated.

In order to stop a project file, is as easy as clicking on the *Stop* button next to the *Play* button within the upper menu.

3 Description of the scenarios

In this Section, the scenarios developed will be briefly described. All scenarios include the required behaviors in Sections 3.1 and 3.9 in Deliverable D1.1. Those behaviors are described here in Subsection 3.1.

3.1 Actions at the start of all RET tasks

At the beginning the robot stays standing up in a relax pose waiting to start. Two motivating motions are provided to try to engage the child, see *AirJuggle* and *CallSomeone* boxes in Figure 3 as well as a box to make the robot dance, see blue frame in Figure 3. There is also a box called *Random* which creates 6 different behaviors with the purpose of engaging the child.

There are several text boxes to start the session with the child and one to suggest to have a break. Click on each of them to make the robot say something. These boxes are those under the red frame in Figure 3. There are also text boxes to re-engage the child if something was unexpected, see yellow frame in Figure 3

3.2 Joint Attention Diagnosis ADOS

Within this project file there are text boxes to be used to make the robot say *Look at that <object>* where object could be a plane, a car, a cup or a flower, see purple frame in Figure 3; text boxes to provide feedback to the child after each interaction; and there are also boxes to direct the gaze of the robot (left, right and center) and boxes for pointing to left and right see green frame in Figure 3.

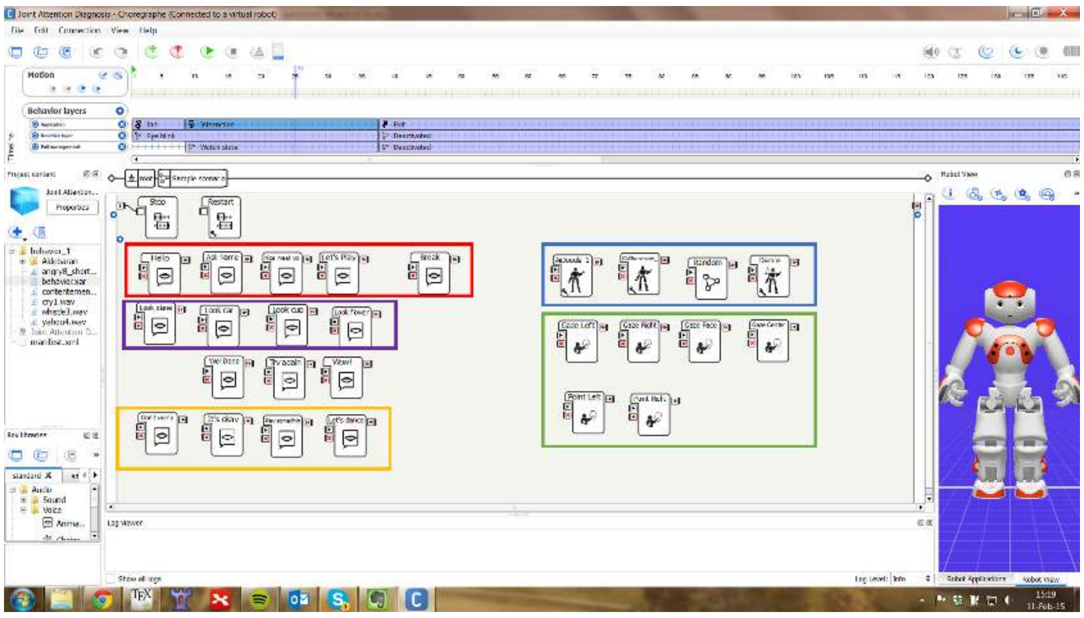


Figure 3: Joint Attention Diagnosis project file.

3.3 Joint Attention Intervention

In addition to what was included in the previous scenario, within this project file there are text boxes to ask the child to choose an emotion, purple frame in Figure 4; and boxes for expressing different emotions, see green frame in Figure 4.

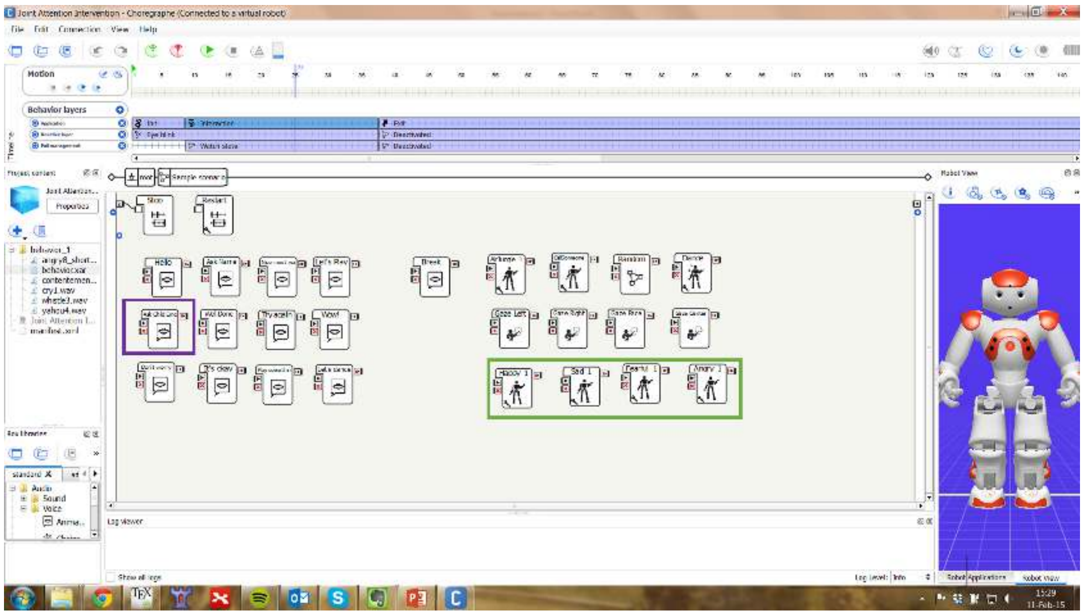


Figure 4: Joint Attention Intervention project file.

3.4 Imitation Diagnosis with Objects

Additionally to what there are in other project files, within this one there are text boxes to ask the therapist to give the robot certain object, see purple frame in Figure 5; a box to ask the child to replicate the motion, in yellow in Figure 5; and boxes for making different gestures with and without the objects, see green frame in Figure 5.

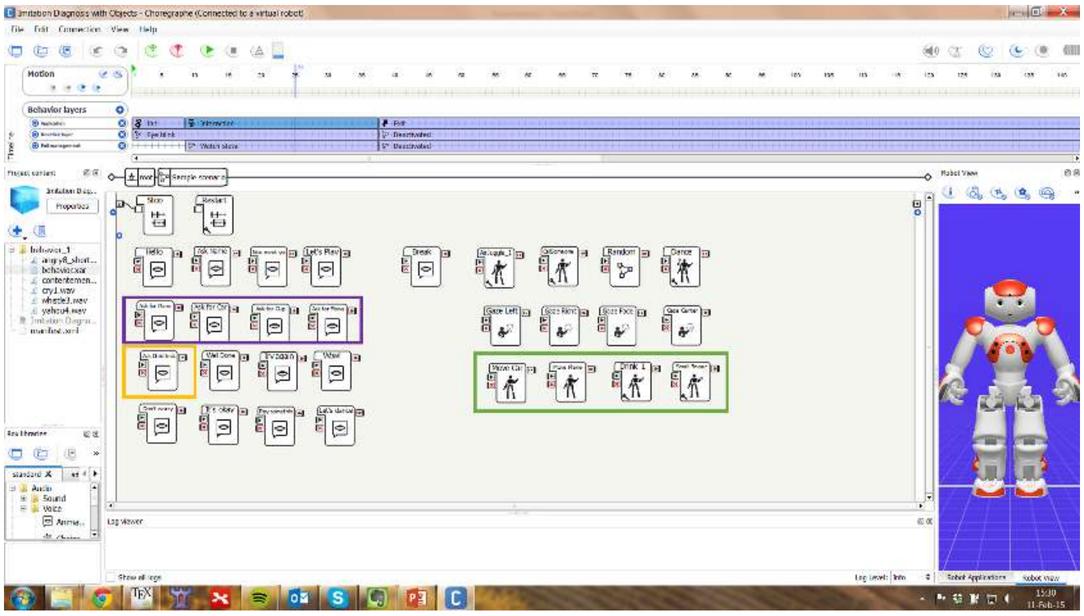


Figure 5: Imitation Diagnosis with Objects project file.

In order to make the robot pick up an object, the therapist should approach the object to the robot and touch its head, then the robot will open its hands. With a second touch in the head, the robot will close the hands grabbing the object. To make the robot drop the object, the same procedure should be followed.

3.5 Imitation Diagnosis without Objects

For this project file there are specific boxes to make the robot make 4 different motions (cover its eyes, touch its head, airplane arms and wave with one hand), see green frame in Figure 6.

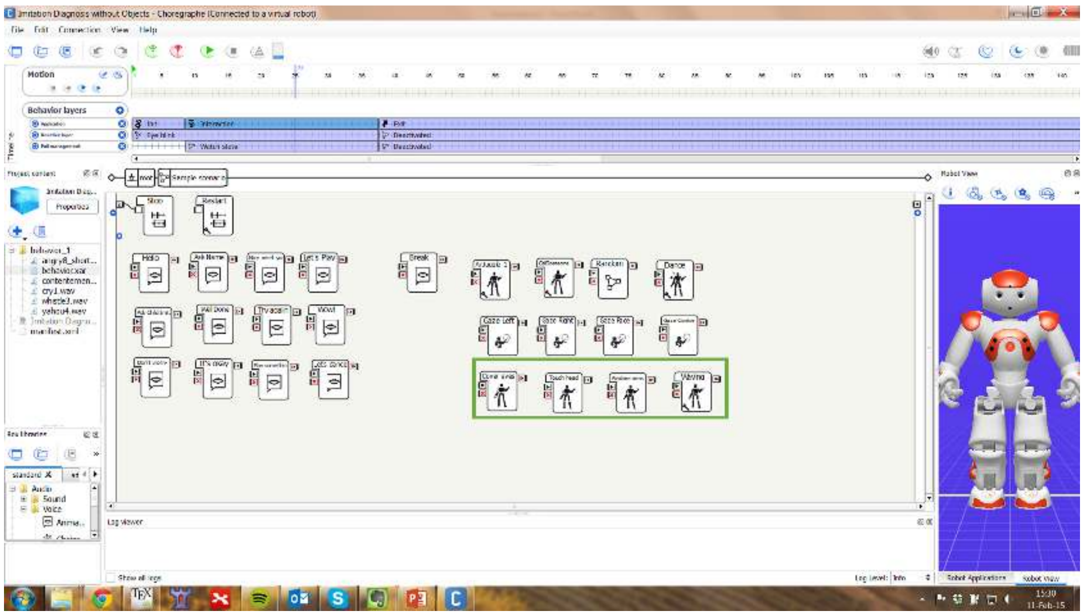


Figure 6: Imitation Diagnosis without Objects project file.

3.6 Imitation Intervention without Objects

This project file includes text boxes already described in previous scenarios.

3.7 Turn-taking diagnosis

The turn-taking diagnosis behaviour for the WoZ experiments is handled by a separate system based on the software engineering framework established in WP3. Please refer to the technical report "*Sandtray Wizard-of-Oz System for Turn-taking Intervention*" for more details.

3.8 Turn-taking intervention

The turn-taking intervention behaviour for the WoZ experiments is handled by a separate system based on the software engineering framework established in WP3. Please refer to the technical report "*Sandtray Wizard-of-Oz System for Turn-taking Intervention*" for more details.

4 Limitations of the software

As the experiments will follow the Wizard of Oz methodology, the therapists are responsible for following the script of each scenario. There is no autonomous behavior within this software at all. Moreover, in case it is needed more boxes can be added.

If you need further information about Choregraphe, you may find it in Aldebaran documentation (`http://doc.aldebaran.com/2-1/news/2.0/choregraphe_rn2.0.html`).

5 Miscelanea

The software has been developed using Naoqi version 2.1.2.17. If you need to upgrade it, please follow the instructions from the wiki (`https://dreamproject.aldebaran.com/projects/dream/wiki/Nao_software`).

Period 2

Touchscreen-Mediated Child-Robot Interactions Applied to ASD Therapy*

Paul Baxter[1], Silviu Matu[2], Emmanuel Senft[1], Cristina Costescu[2], James Kennedy[1], Daniel David[2] and Tony Belpaeme[1]
[1] *Centre for Robotics and Neural Systems, The Cognition Institute, Plymouth University, U.K.*
[2] *Department of Clinical Psychology and Psychotherapy, Babes-Bolyai University, Romania*

Abstract. Robots are finding increasing application in the domain of ASD therapy as they provide a number of advantageous properties such as replicability and controllable expressivity. In this abstract we introduce a role for touchscreens that act as mediating devices in therapeutic robot-child interactions. Informed by extensive work with neurotypical children in educational contexts, an initial study using a touchscreen mediator in support of robot-assisted ASD therapy was conducted to examine the feasibility of this approach, in so doing demonstrating how this application provides a number of technical and potentially therapeutic advantages.

Keywords: ASD, Robot-Assisted Therapy, Sandtray

INTRODUCTION

The application of robots to aid in the therapy of children with Autistic Spectrum Disorders (ASD) has become increasingly established [1], [2], with evidence suggesting that it can provide beneficial outcomes for the children [3]. In addition to this, recent efforts have emphasised providing an increasing degree of autonomy for the robot [4].

Providing such autonomous behaviour in interaction contexts is a challenging task, with sensory and motor limitations imposing a number of constraints. In our previous work, we have developed a methodology that makes use of a touchscreen mediator between children and robots to overcome a number of these difficulties: the Sandtray [5]. In this setup, a child and a robot engage in a collaborative task that is provided on the touchscreen (e.g. sorting of images into categories). The Sandtray has been successfully applied to a range of neurotypical child-robot interaction studies in various contexts, for example behavioural alignment [6], education [7], and others. As the Sandtray was inspired by the therapeutic intervention of sandplay (with this having proposed advantages for children with ASD [8]), we now seek to apply this same methodology to robot-assisted ASD therapy.

Touchscreens (without the robot) have found previous applications to this domain [9]. For example, a touchscreen has been used to enforce collaborative activity between pairs of children with ASD, resulting in an increase in coordination and negotiation behaviours [10], a finding supported elsewhere [11]. Furthermore, there have been attempts to enable sandplay therapy-like interactions with touchscreens [12], although our approach differs in both application context and involvement of the robot. These studies indicate the suitability of using touchscreens for children with ASD.

There are a number of advantages afforded by the use of such a mediating touchscreen in HRI. Firstly, it provides a shared space for collaboration that does not require complex manual dexterity for either the child or the robot; indeed it provides the same affordances for both interactants (pointing and dragging). Secondly, it reduces the sensory processing load (vision processing) on the robot since information on screen-oriented activity by the child can be obtained directly from the touchscreen. Thirdly, it provides a straightforward means of changing the task (or more broadly the interaction context) by just changing the images displayed on the screen: for instance, a sorting task can be appropriate for domains as diverse as mathematics and nutrition just by changing the pictures displayed.

The aim of this contribution is to motivate and illustrate how such touchscreen mediators can specifically serve as useful tools in the domain of robot-assisted therapy by first describing an application currently in progress, and then discussing the opportunities and challenges for the future.

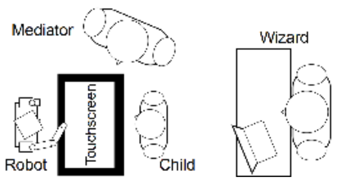


Fig. 1. Indicative setup and use of touchscreen for child-robot therapeutic interaction - robot is controlled by a wizard, and the mediator provides input to the interaction if needed (*not to scale; positions are indicative only*).

APPLICATION CASE STUDY: TURN-TAKING

An initial application to ASD therapy has been implemented and evaluated. Turn-taking is an important social skill that is used as part of therapeutic interventions [13]. We have created an emotion image categorisation task (using *sad* and *happy* faces) on the Sandtray for a child and Nao robot to play, with robot verbal behaviour used to encourage turn-taking behaviours. For this study, the robot was explicitly remote controlled (*wizarded*) by a remote operator (fig. 1).

With a four year-old girl with ASD, six interaction sessions with the Robot-Sandtray turn-taking task were con-

*This work was supported by the EU FP7 project DREAM (grant number 611391, http://dream2020.eu/).

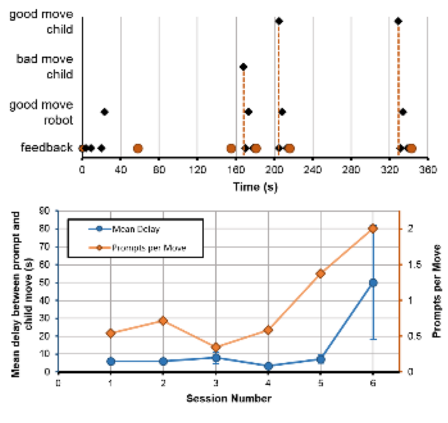


Fig. 2. (*Top*) Sample data from the sixth child-robot Sandtray turn-taking interaction session. The *feedback* was employed to encourage the child to move and to give them feedback. Orange circles indicate robot encouragements for the child to take a turn. (*Bottom*) Trends over six sessions, showing change in delay between robot prompt and the child moving, and the mean number of prompts per child move (*with 95% CI*).

ducted over a period of four weeks. Other robot-based therapy activities were conducted at a separate time. Each interaction had a mean length of 11:06 mins (*sd 5:03 mins*).

Since interaction data can be captured through the touchscreen, it is possible to retrospectively examine the events that occurred and their timing. Considering the relationship between robot encouragement and child moves in a single interaction (e.g. fig. 2, top), the data suggest that both the number of robot encouragement instances required before the child made a move, and the delay between suggestions and actual moves increases over time (fig. 2, bottom). A clinical explanation for this relationship is not proposed here, although the ideal behaviour in this context is a turn-taking interaction with the robot, without necessarily requiring explicit prompting. What can be noted though is that data such as these provide some insight into the interaction between the child and the robot over time.

DISCUSSION AND OPEN QUESTIONS

The examination and use of touchscreen-derived information has two benefits. Firstly, it may come to constitute an additional source of information for the therapist to aid in diagnosis or inform future therapy, with additional processing making aspects of emotion available for example [14]. The extent to which this is clinically useful is an open question that requires investigation. It should however be noted that we do not suggest that such data can replace traditional diagnosis information, rather that it can provide supplemental information. It should be further noted that the touchscreen-derived information alone is likely to be insufficient to provide a complete characterisation of the child's behaviour.

Secondly, since the information captured by the touchscreen is directly accessible to the robot system, it can be used by the robot to adapt its behaviour to the specific circumstances of an individual child in individual interactions, e.g. [6]. In the case of autonomous robot behaviour, such a source of information that does not require the overhead of complex visual or audio processing is a significant benefit.

Extensive previous work has been conducted with this touchscreen mediated interaction between (neurotypical) children, and robots. While this has shown that the touchscreen effectively constrains the content of the interaction (thus facilitating robot autonomous behaviour) [15], it is an open question as to whether a similar effect (such as helping to maintain focus on the interaction) is observable for children with ASD, or over what time scales such an effect may be manifested.

To conclude, we have presented data from an example set of interactions between a child with ASD and a robot in the context of the Sandtray. This provides an illustration of the type of data that is readily available through the use of the touchscreen mediation technology. While further development and data collection is required (and is ongoing), we suggest that the use of touchscreens as mediators for child-robot interactions in the context of ASD therapy provides benefits in terms of behaviour characterisation and technical feasibility that should be further taken advantage of.

REFERENCES

1. B. Robins et al, "Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills?" *Universal Access in the Information Society*, 4(2): 105–120, 2005.
2. B. Scassellati et al, "Robots for use in autism research," *Annual review of biomedical engineering*, 14: 275–94, 2012.
3. C. A. Costescu et al, "The Effects of Robot-Enhanced Psychotherapy: A Meta-Analysis," *Review of General Psychology*, 18(2): 127–136, 2014.
4. S. Thill et al, "Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook," *Paladyn*, 3(4): 209–217, 2013.
5. P. Baxter et al, "A Touchscreen-Based "Sandtray" to Facilitate, Mediate and Contextualise Human-Robot Social Interaction," in *7th HRI Conference*. Boston, MA, U.S.A.: IEEE Press, 2012, pp. 105–106.
6. P. Baxter et al, "Cognitive architecture for human-robot interaction: Towards behavioural alignment," *Biologically Inspired Cognitive Architectures*, 6: 30–39, 2013.
7. J. Kennedy et al, "The Robot Who Tried Too Hard: Social Behaviour of a Robot Tutor Can Negatively Affect Child Learning," in *10th HRI Conference*. Portland, Oregon, USA: ACM Press, 2015, pp. 67–74.
8. L. Lu et al, "Stimulating creative play in children with autism through sandplay," *Arts in Psychotherapy*, 37: 56–64, 2010.
9. W. Chen, "Multitouch Tabletop Technology for People with Autism Spectrum Disorder: A Review of the Literature," *Procedia Computer Science*, 14(1877): 198–207, 2012.
10. A. Battocchi et al, "Collaborative puzzle game: a tabletop interface for fostering collaborative skills in children with autism spectrum disorders," *Journal of Assistive Technologies*, 4(1): 4–13, 2010.
11. G. F. Mireya Silva et al, "Exploring collaboration patterns in a multitouch game to encourage social interaction and collaboration among users with autism spectrum disorder," *Computer Supported Cooperative Work (CSCW)*, 24(2-3): 149–175, 2015.
12. M. Hancock et al, "Supporting sandtray therapy on an interactive tabletop," *28th CHI Conference*, pp. 21–33, 2010.
13. C. A. Pop et al, "Enhancing play skills, engagement and social skills in a play task in ASD children by using robot-based interventions. a pilot study," *Interaction Studies*, 15(2): 292–320, 2014.
14. Y. Gao et al, "What Does Touch Tell Us about Emotions in Touchscreen-Based Gameplay?" *ACM Transactions on Computer-Human Interaction*, 19(4): 1–30, 2012.
15. J. Kennedy et al, "Constraining Content in Mediated Unstructured Social Interactions: Studies in the Wild," in *5th AFFINE Workshop at ACII 2013*. Geneva, Switzerland: IEEE Press, 2013, pp. 728–733.

Human-Guided Learning of Social Action Selection for Robot-Assisted Therapy

Emmanuel Senft, Paul Baxter, Tony Belpaeme
Centre for Robotics and Neural Systems,
Cognition Institute Plymouth University, U.K.
{emmanuel.senft, paul.baxter, tony.belpaeme}@plymouth.ac.uk

Abstract

This paper presents a method for progressively increasing autonomous action selection capabilities in sensitive environments, where random exploration-based learning is not desirable, using guidance provided by a human supervisor. We describe the global framework and a simulation case study based on a scenario in Robot Assisted Therapy for children with Autism Spectrum Disorder. This simulation illustrates the functional features of our proposed approach, and demonstrates how a system following these principles adapts to different interaction contexts while maintaining an appropriate behaviour for the system at all times.

1 Introduction

Humans are interacting increasingly with machines, and robots will be progressively more important partners in the coming years. Human-human interactions involve high dimensionality signals and require complex processing: this results in a large quantity of data that ideally needs to be processed by an autonomous robot. One potential solution is the application of machine learning techniques. Specifically, online learning is desirable, however, some level of initial knowledge and competencies are required to avoid pitfalls in the early phases of the learning process, particularly in contexts where random exploration could lead to undesirable consequences.

In this paper, we propose an approach inspired by, but separate from, learning from demonstration to guide this early learning of an action selection mechanism for autonomous robot interaction with a human, by taking advantage of the expert knowledge of a third-party human supervisor to prevent the robot from exploring in an inappropriate manner. We first present the formal framework in which our action selection strategy learning takes place (section 2), then illustrate this with a case study from the domain of Robot Assisted Therapy for children with Autism Spectrum Disorder (ASD), where the incorrect selection of actions can lead to an unacceptable impact on the goals of the interaction (section 3).

Appearing in Proceedings of the 4th Workshop on Machine Learning for Interactive Systems (MLIS) 2015, Lille, France. JMLR: W&CP volume 40. Copyright 2015 by the authors.

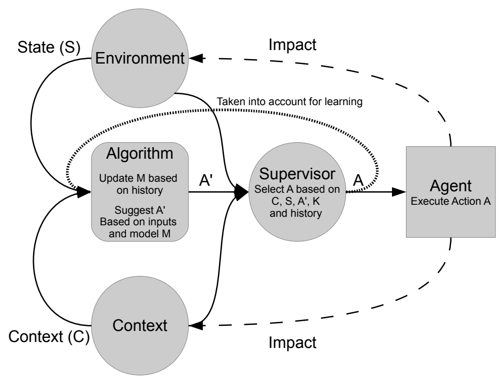

State (S)
Environment
Impact
Taken into account for learning
Algorithm
Update M based on history
Suggest A' Based on inputs and model M
A'
Supervisor
Select A based on C, S, A', K and history
A
Agent
Execute Action A
Context (C)
Context
Impact

Figure 1: The supervised online learning of autonomous action selection mechanism.

2 Supervised Emergent Autonomous Decision Making

2.1 Framework

The situation considered involves a robotic agent, a human supervisor of the agent, and a human with which the agent, but not the supervisor, should interact. The agent proposes actions that are accepted or rejected by the supervisor prior to executing them. The method proposed in this paper aims at enabling the agent to progressively and autonomously approximate the ideal behaviour as specified by the supervisor.

Our framework has five components: an agent and an environment interacting with each other, a supervisor, the algorithm controlling the agent and a context characterising the interaction between the agent and the environment. The agent has a defined set of available actions $\mathcal{A}$. The environment could be a human, a robot, or a computer for example and is characterised by a n-dimensional vector $\mathcal{S} \in \mathcal{R}^n$, which is time varying. The context $\mathcal{C} \in \mathcal{R}^m$ gives a set of parameters defining higher-level aspects, such as goals or the state, of the interaction, see figure 1. The supervisor and the agent have direct access to the context, but may ignore the real value of the state and see it only through observations.

The principal constraints are that the interaction has one or more high level goals, and some available actions can have a negative impact on these goals if executed in specific states. This should be avoided, so algorithms depending on randomness to explore the environment state space are inappropriate.

In order to simplify the system, we are making the following assumptions. Firstly, that the environment, while dynamic, is consistent: it follows a defined set of rules $\mathcal{E}$ which also describe how the context is updated. Secondly, the supervisor $\mathcal{T}$ is omniscient (complete knowledge of the environment), constant (does not adapt during the interaction), and coherent (will react with the same action if two sets of inputs are identical). Finally, the supervisor has some prior knowledge of the environment $\mathcal{K}$.

The algorithm has a model $\mathcal{M}$ of the supervisor and the environment and will update it through online learning following the learning method $\mathcal{L}$. $\mathcal{M}$ is iteratively updated based on supervisor feedback to approximate $\mathcal{T}$ and $\mathcal{E}$, in this way progressively approximating the action that the supervisor would have chosen, and what impact this would have on the environment. Equation 1 describes the update of each part of the framework from the step n to $n+1$.

$$\begin{aligned} M_n &: (C_{0\to n}, S_{0\to n}, A_{0\to n-1}, A'_{0\to n-1}) \longrightarrow A'_n \\ \mathcal{T} &: (C_{0\to n}, S_{0\to n}, A'_{0\to n}, A_{\to n-1}, \mathcal{K}) \longrightarrow A_n \\ \mathcal{E} &: (C_{0\to n}, S_{0\to n}, A_{0\to n}) \longrightarrow (S_{n+1}, C_{n+1}) \\ \mathcal{L} &: (C_{0\to n+1}, S_{0\to n+1}, A'_{0\to n}, A_{0\to n}) \longrightarrow M_{n+1} \end{aligned} \tag{1}$$

At the start of the interaction, the environment is in a state S_0 with the context C_0 and the algorithm has a model M_0. Applying M_0 to C_0 and S_0, the algorithm will select an action A'_0 and propose it to the supervisor. The supervisor can either accept this action or select a new one according to $\mathcal{T}$, and makes the agent execute the resulting action A_0. The environment will change to a new state S_1 and the context will be updated to C_1 according to $\mathcal{E}$. Based on S_1, S_0, C_1, C_0, A_0, and A'_0, the algorithm will update its model to M_1. The process can then be repeated based on the updated model.

2.2 Related Work

The approach we take here necessarily requires the application of machine learning, but we do not commit at this stage to a single algorithmic approach; the specific requirements for our application include online learning, deferring to an external supervisor, and being able to handle a dynamic environment.

A widely used method to transfer knowledge from a human to a robot is Learning from Demonstration (LfD), see [2] for a survey. In the case of policy learning, a teacher provides the learning algorithm with correct actions for the current state and repeats this state-action mapping for enough different states to give the algorithm a general policy. LfD is usually combined with supervised learning: trying directly to map outputs and inputs from a teacher, see [12] for a list of algorithms that can be used in supervised learning. The other important point is how the demonstrations are generated, a first approach is using batch learning: the teacher trains the algorithm during a training phase after which the robot is used in full autonomy [11]. Or there may be no explicit training phase; using online learning the demonstrations are given during the execution if required: the robot can request a demonstration for the uncertain states, e.g. when a confidence value about the action to perform is too low [6].

Another method is Reinforcement Learning: the algorithm tries to find a policy maximising the expected reward [3, 10]. However, this implies the presence of a reward function, which may not be trivial to describe in domains (such as social interaction) that do not lend themselves to characterisation. Consequently Abbeel and Ng proposed to use Inverse Reinforcement Learning by using an expert to generate the reward function [1], subsequently extended to use partially-observable MDPs [8], although expert-generated rewards also pose problems on the human side [17].

The goal of our proposed approach differs from these alternative existing methods. The intention is to provide a system that can take advantage of expert human knowledge to progressively improve its competencies without requiring manual intervention on every interaction cycle. This is achieved by only asking the human supervisor to intervene with corrective information when the proposed action of the robot agent is deemed inappropriate (e.g. dangerous) prior to actual execution. This allows the robot to learn from constrained exploration; a consequence of this is that the load on the supervisor should reduce over time as the robot learns. The supervisor nevertheless retains control of the robot, and as such we characterise the robot as having *supervised autonomy.* Contrary to the active learning approach used by, for example, Cakmak and Thomaz [5] the robot is not asking a question and requiring a supervisor response, it is proposing an action which may or may not be corrected by the supervisor.

3 Case Study: Application to Therapy

One potential application area is Robot Assisted Therapy (RAT) for children with Autism Spectrum Disorders (ASD). Children with ASD generally lack typical social skills, and RAT can help them to acquire these competencies, with a certain degree of success, e.g. [7, 14]. However, these experiments typically use the Wizard of Oz (WoZ) paradigm [13], which necessitates a heavy load on highly trained human operators.

We propose the use of supervised autonomy [15, 16], where the robot is primarily autonomous, but the therapeutic goals are set by a therapist who maintains oversight of the interaction. Having a supervised autonomous robot would reduce the workload on the therapist.Both the therapist and the robot would be present in the interaction, the robot interacting with the child and the therapist supervising the interaction and guiding the robot while it is learning its action selection policy.

The formalism described above (section 2.1) can be directly applied to this scenario. In this case, the context is the state of the task selected by the therapist to help the child develop certain social competencies, for example, a collaborative categorisation game [4] intended to allow the child to practice turn taking or emotion recognition. The state may be defined using multiple variables such as motivation, engagement, and performance exhibited by the child during the interaction, the time elapsed since the last child's action, and their last move (correct or incorrect). The robot could have a set of actions related to the game, such as proposing that the child categorises an image, or giving encouragement to the child.

In this scenario, the goal would be to allow the child to improve their performance on the categorisation task, and this would be done by selecting the appropriate difficulty level and finding a way to motivate the child to play the proposed game. We can expect the child to react to the robot action and that these reactions can be captured by the different variables that define the child's state (as provided by therapists for example). In principle, while precise determinations are likely to be problematic, we assume that some aspects of these variables can be estimated using a set of sensors (e.g. cameras and RGBD sensors to capture the child's gaze and position; the way the child interacts with the touch screen; etc). For the remainder of this paper, however, we assume that a direct estimation of internal child states are available to the system.

3.1 Proof of concept

A minimal simulation was constructed to illustrate the case study described above. The state $\mathcal{S}$ is defined using three variables: the child's performance, engagement and motivation in the interaction. The robot has the following set of actions $\mathcal{A}$: encouragement (give a motivating feedback to the child), waving (perform a gesture to catch the child's attention), and proposition (inviting the child to make a classification). In this minimal example, the environment $\mathcal{E}$ is the child model. A minimal model of the child was constructed that encompassed both processes that were dependent on the robot behaviour (e.g. responding to a request for action), and processes that were independent of the robot behaviour (e.g. a monotonic decrease of engagement and motivation over time independently of other events). The reaction of the model follows a rule-based system, but the amplitude of the response is randomly drawn from a normal distribution to represent the stochastic aspect of the child's reaction and the potential influence of non-defined variables in the state. A number of simplifications are necessary, such as the assumption of strict turn-taking and interactions in discrete time. The minimal child model is summarised in figure 2.

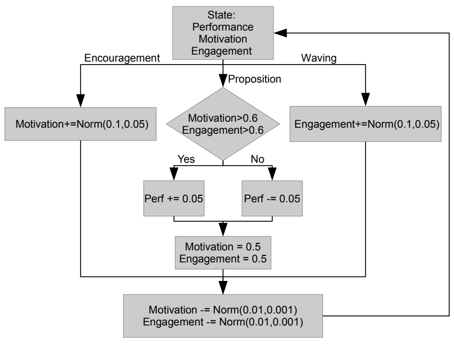


Figure 2: Model of the child used in the minimal simulation; random numbers are drawn from normal distributions.

Formally, the minimal simulation follows the framework established above (equation 1), with the simplification that a history of prior states, contexts, and actions is not used in the learning algorithm. This results in a setup where the system makes a suggestion of an action to take, which the supervisor can either accept or reject, in which case an alternative action is chosen (figure 3).

This allows the supervisor to take a more passive approach when the algorithm selects an acceptable action since they will only have to manually select a corrective action when this is needed. If the learning method is effective, the number of corrective actions should decrease over time, decreasing the workload on the therapist over the interaction.

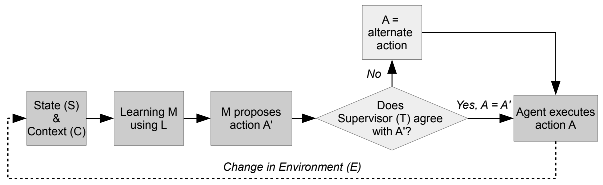


Figure 3: Description of agent's action selection process: the agent proposes actions that are validated by the supervisor prior to execution.

The learning model $\mathcal{M}$ is a MultiLayer Perceptron (MLP), with three input nodes for the input states, three output nodes for the three possible actions and nine nodes in the hidden layer. The model is trained using backpropagation (as $\mathcal{L}$), the true labels are given by the supervisor decision: output of 1 for the action selected by the supervisor (A) and -1 for the other ones. A Winner-Takes-All process is applied on the output of the MLP to select the action suggested by the robot (A').

Figure 4 shows a subset of a run from step 100 to 150. With this approach, there is no distinct learning and testing phases, but in the first part of the interaction (before step 100), the supervisor had to produce multiple corrective actions to train the network to express the desired output. The strategy used by the supervisor is the following: if the motivation is lower than 0.6 the supervisor enforces the action 'encouragement', else if the engagement is below 0.6 'waving' is enforced, and if both are above 0.6 then a proposition is made. The first graph presents the evolution of the state over time, and the second one the output of the MLP for each action. The vertical red lines represent an intervention from the supervisor, i.e. a case where the supervisor enforces a different action than the one suggested by the MLP. The action actually executed is represented by a cross with the same colour as the respective curves.

Figure 5 shows a comparison of the cumulative total of the different actions suggested and of the intervention as well as the child performance for three different models of a child and for a random action selection scheme. The difference in the child models in the three first graphs is the value of the thresholds required for a good classification action, high reactive child: 0.6 and 0.6, asymmetrically: 0.9 for encouragement and 0.6 for engagement, and low reactive: 0.9 and 0.9. Below these thresholds, a proposition would lead to a bad action decreasing the performance. It can be observed that the algorithm learns different strategies for each child and that there is more learning apparent at the start of the interaction than at the end (the rate of interventions is decreasing over time), indicating that the system is choosing the appropriate action at the appropriate time, and that the workload on the supervisor (necessity to provide these corrective actions) decrease over time. The last plot demonstrates a random action selection with a high reactive child. Contrary to the other cases, the child's performance decreases over time, and the number of interventions increases. Here, a *bad* action only decreases the performance, but in reality it may result in the termination of the interaction, which must be avoided.

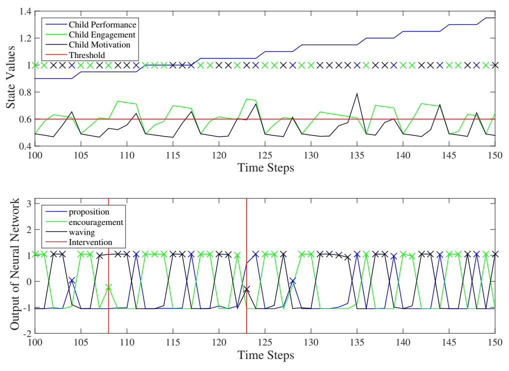


Figure 4: Subset of an interaction from step 100 to 150.

4 Discussion

While demonstrating promise, there are a number of limitations to the framework as presented. The assumptions described in section 2.1 are typically violated when working with humans. Firstly the children are all different, and a method learned for one child may often not be suited when working with another. Furthermore, the same child may have non-consistent behaviour between the sessions and even within a single session. There is no perfect solution to solve this problem, but we can expect that with enough training sessions and a more complex learning algorithm, the system would be able to capture patterns and react to the different behaviours appropriately. Since it is ex-

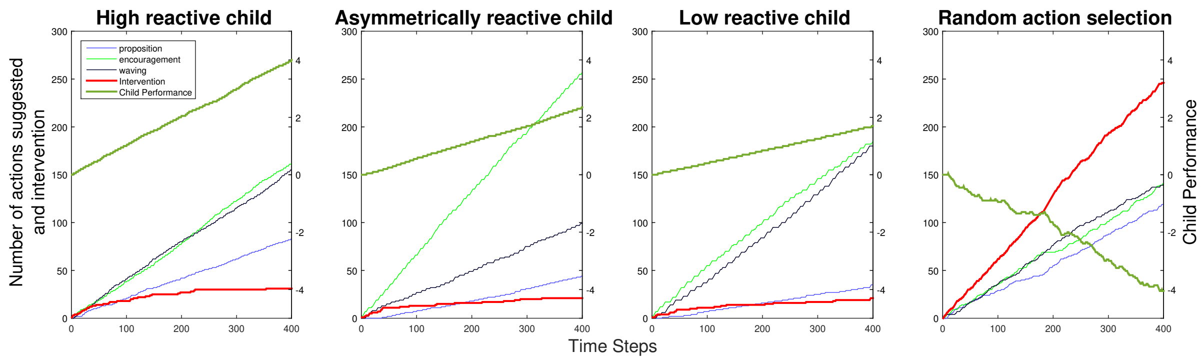


Figure 5: Comparison of the cumulative total of the different actions suggested, the supervisor interventions required, and child performance for three different models of a child (highly responsive; asymmetrically increased responsiveness to engagement than motivation; low responsiveness), and a random action selection scheme.

pected that in a real application of such an approach a therapist who knows the child will always be present, we propose that for a new child the algorithm will use a generic strategy based on previous interactions with other children, with subsequent fine-tuning under supervision.

Another assumption that is likely to be violated is that of a perfect supervisor. As explained in [6] humans are not always consistent nor omniscient, but methods presented in the literature can be used to cope with these inconsistencies if enough data is gathered. Further mitigating solutions could be employed, such as the robot warning the therapist if it is about to select an action which had negative consequences in a previous interaction (even if for a different child). Furthermore, it may not be possible to measure the true internal states of the child in the real world, with imperfect estimations of these states being more likely accessible. In this case, inspiration from [9] can be used to mixed the POMDP framework with the help of an exterior oracle. Another problem which will have to be addressed in the future is the difference in inputs between the robot and the therapist: the therapist will have access to language, more subtle visual features and their prior experience, whereas the robot may have direct and precise access to some aspects of the child's overt behaviour (such as timings of touchscreen interaction).

In the currently implemented case study, we assume that the supervisor responds to the action proposed by the robot within some predetermined fixed time, whether this response is accept or reject (figure 3). This, in principle, allows the supervisor to only actively respond if a proposed action is clearly inappropriate. In further developments, we will incorporate a measure of certainty (given prior experience) into the time allowed the supervisor to respond to the proposed action: for example increasing the time available if the confidence in the proposed action is low. This modulation of the load on the supervisor's attention according to confidence should result in the supervisor being able to increasingly pay attention to the child directly, rather than to the robot system, as training progresses.

5 Conclusion

We have presented a general framework to progressively increase the competence of an autonomous action selection mechanism that takes advantage of the expert knowledge of a human supervisor to prevent inappropriate behaviour during training. This method is particularly applicable to application contexts such as robot-assisted therapy, and our case study has provided preliminary support for the utility of the approach. While the simulation necessarily only provided a minimal setup, and thus omitted many of the complexities present in a real-world setup, we have nevertheless shown how the proposed method resulted in the learning of distinct action selection strategies given differing interaction contexts, although further refinement is required for real-world application. Indeed, given real-world supervisor knowledge limitations, we suggest it will furthermore be possible for a suitably trained action selection mechanism of this type to aid the supervisor in complex and highly dynamic scenarios.

Acknowledgements

This work is supported by the EU FP7 DREAM project (grant 611391).

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 1–8, 2004.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 1998.

[4] P. Baxter, R. Wood, and T. Belpaeme. A touchscreen-based sandtrayto facilitate, mediate and contextualise human-robot social interaction. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 105–106. IEEE, 2012.

[5] M. Cakmak and A. L. Thomaz. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 17–24. ACM, 2012.

[6] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34(1):1, 2009.

[7] K. Dautenhahn. Robots as social actors: Aurora and the case of autism. In *Proc. CT99, The Third International Cognitive Technology Conference, August, San Francisco*, volume 359, page 374, 1999.

[8] F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using bayes risk for active learning in pomdps. In *Proceedings of the 25th international conference on Machine learning*, pages 256–263. ACM, 2008.

[9] F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using bayes risk for active learning in pomdps. In *Proceedings of the 25th international conference on Machine learning*, pages 256–263. ACM, 2008.

[10] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, pages 237–285, 1996.

[11] W. B. Knox, S. Spaulding, and C. Breazeal. Learning social interaction from the wizard: A proposal. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[12] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[13] L. Riek. Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. *Journal of Human-Robot Interaction*, 1(1):119–136, Aug. 2012.

[14] B. Robins, K. Dautenhahn, R. T. Boekhorst, and A. Billard. Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills? *Universal Access in the Information Society*, 4(2):105–120, July 2005.

[15] E. Senft, P. Baxter, J. Kennedy, and T. Belpaeme. When is it better to give up?: Towards autonomous action selection for robot assisted asd therapy. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, HRI'15 Extended Abstracts, pages 197–198, New York, NY, USA, 2015. ACM.

[16] S. Thill, C. A. Pop, T. Belpaeme, T. Ziemke, and B. Vanderborght. Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. *Paladyn*, 3(4):209–217, Apr. 2013.

[17] A. L. Thomaz and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008.

SPARC: Supervised Progressively Autonomous Robot Competencies

Emmanuel Senft, Paul Baxter, James Kennedy, and Tony Belpaeme

Centre for Robotics and Neural Systems, Plymouth University, United Kingdom
{emmanuel.senft, paul.baxter, james.kennedy, tony.belpaeme}@plymouth.ac.uk

Abstract. The Wizard-of-Oz robot control methodology is widely used and typically places a high burden of effort and attention on the human supervisor to ensure appropriate robot behaviour, which may distract from other aspects of the task engaged in. We propose that this load can be reduced by enabling the robot to learn online from the guidance of the supervisor to become progressively more autonomous: Supervised Progressively Autonomous Robot Competencies (SPARC). Applying this concept to the domain of Robot Assisted Therapy (RAT) for children with Autistic Spectrum Disorder, a novel methodology is employed to assess the effect of a learning robot on the workload of the human supervisor. A user study shows that controlling a learning robot enables supervisors to achieve similar task performance as with a non-learning robot, but with both fewer interventions and a reduced perception of workload. These results demonstrate the utility of the SPARC concept and its potential effectiveness to reduce load on human WoZ supervisors.

1 Introduction

Over the last two decades, an increasing amount of research has been conducted to explore Robot Assisted Therapy (RAT). Using robots in therapies for children with Autism Spectrum Disorder (ASD) has revealed promising results [5, 10, 11]. The Wizard-of-Oz (WoZ) paradigm is typically used for this application, and others, where the robots are not autonomous but tele-operated. Many motivating factors for moving away from WoZ in RAT have been put forward [8, 13]. In particular, autonomous behaviour facilitates repetition of the robot behaviour and decreases the workload on therapists, freeing them to pay attention to other aspects of the interaction. It is the intention of our research to facilitate this shift to robot autonomy.

As the optimal robot behaviour is unlikely to be known in advance (be it in a therapeutic or indeed other domain), and with adaptability during and between the different interactions being generally desirable, it is necessary to provide the robot with learning capabilities. In the context of RAT, by using the knowledge of a therapist, the learning can be guided so that it is faster and safer, especially as the robot cannot use random exploration to acquire knowledge about its environment when interacting with children with ASD in case of negative therapeutic and/or clinical outcomes. We propose an approach taking inspiration from the Learning from Demonstration and online learning literature, and call it SPARC: Supervised Progressively Autonomous Robot Competencies. In SPARC, a therapist guides the robot in the early stages of the interaction, and progressively, the robot learns an action policy adapted to the particular therapeutic session [12]. Assuming the effective learning of the robot in this context, the therapist can allow the robot to behave increasingly autonomously, whilst maintaining oversight. Although not reducing the attentional requirements, this would reduce the physical interventions to direct the robot behaviour required by the therapist. Thus, by proposing and executing good actions, SPARC can reduce the therapists' workload.

A RAT scenario typically involves three parties: the patient, a robot, and the human therapist. In this context, the therapist does not interact with the patient directly, but rather through the robot. The therapist could therefore be described as playing the role of a robot supervisor. The focus of this paper is not on a new learning algorithm, but rather on the interaction between the robot and the therapist (supervisor), and the role that robot autonomy can play in this relationship. Specifically, as an initial validation of the principle, we seek to assess whether the SPARC concept can feasibly result in a reduction in workload for the supervisor, even given different strategies used by different individuals. A user study employing a novel methodology is conducted (section 3), demonstrating that progressive robot autonomy does indeed result in lower supervisor workload (section 4). This outcome provides support for the proposed approach and motivates further development efforts in the domain of RAT.

2 Related work

A number of research groups have studied the use of robot in therapy for children with ASD, which allowed children to express previously unseen social behaviour for example [9, 10]. Two primary methods have been used for these investigations: using an autonomous robot following preprogrammed rules [6, 14], or using the WoZ paradigm, allowing more flexibility in the robot's reaction. As noted in [8, 13], using WoZ allows testing and prototyping of interaction scenarios, but researchers should consider moving away from it to achieve more scalability, more repeatability, and to allow the use of robots without increasing the workload on therapists. Complex behaviour is required for a therapeutic robot, thereby making learning a desirable feature for future, more autonomous, RAT. As therapists possess the knowledge required to make appropriate decisions in different contexts, Learning from Demonstration [1] provides a useful starting point. Recently, Knox *et al.* proposed the Learning from Wizard paradigm in [7]. The robot is first controlled by a human operator as in a WoZ scenario, and after a number of interactions, batch learning is applied on the previous interaction data to obtain autonomous behaviour.

A fixed action policy of this type is however not desirable for RAT as children may not be consistent between interactions, and thus online learning is required

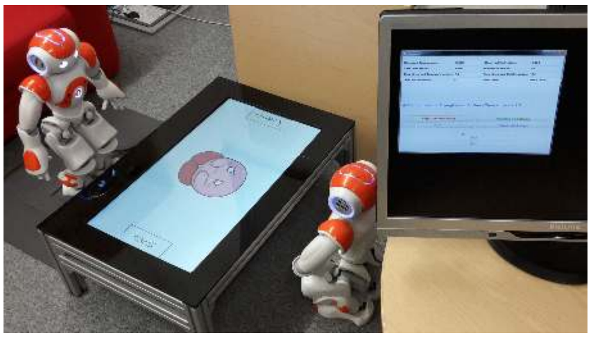


Fig. 1. Setup used for the user study from the perspective of the human supervisor. The *child-robot* (left) stands across the touchscreen (centre-left) from the *wizarded-robot* (centre-right). The supervisor can oversee the actions of the *wizarded-robot* through the GUI and intervene if necessary (right).

to provide the robot with the adaptability necessary to update its action policy depending on the current circumstances. Several experimenters in HRI have studied active learning: a robot actively questions a human teacher in order to request data points or demonstration for an uncertain scenario. A study exploring the type of questions that a robot could ask and the human reactions can be seen in [3], and Chernova and Veloso propose a progressive learning algorithm where a robot can estimate the confidence in its action decision in a fixed environment [4]: if the confidence is too low, a demonstration from a human teacher is required to complete the task.

However, an important element missing from the current literature is online learning for interaction. The robot needs to be able to progressively create an action policy, and update it later if necessary, to reach a more complex interaction behaviour. This paper explores how supervised progressive learning can be used in an interaction scenario and introduces a novel methodology to test this technique.

3 Assessing the effect of a progressively autonomous robot on supervisor workload

The focus of the present study is to assess whether the application of the SPARC concept to RAT results in a decrease in workload for the human supervisor. Two types of robot controller are employed to determine the presence and magnitude of this effect: a robot that learns from the actions of the supervisor to progressively improve its behaviour (*learning* controller), and a robot that only generates random actions (*non-learning* controller).

The methodology used in this paper is based on a real scenario for RAT for children with ASD based on the Applied Behaviour Analysis therapy framework. The aim of the therapy is to help the child to develop/practice their social skills: the task we focus on here is emotion recognition. This scenario involves a child playing a categorisation game with a robot on a mediating touchscreen device [2]. Images of faces or drawings are shown to the child, and she has to categorise them by moving the image to one side or the other depending on whether the picture shown denotes happiness or sadness (e.g. fig. 1). The human supervisor is physically present and guides the robot using the Wizard of Oz paradigm, but does not interact with the child directly.

In our proposed system, the basic interaction structure following the SPARC concept is as follows: the robot suggests an action to the supervisor, the supervisor agrees or disagrees with this suggestion (providing an alternative if disagreeing), the robot executes the action, and then both robot and supervisor observe the outcome. Over time, it is possible for the robot to learn an appropriate strategy based on observations of the child and oversight from the supervisor, with the supervisor still maintaining overall control if necessary.

Given the focus on human supervisor workload, it is necessary to provide a consistent experimental environment across both conditions in which the task, setup, and interaction partner is kept constant. A minimal model of child behaviour is therefore used to stand in for a real child. A second robot is employed in the interaction to embody this child model: we term this the *child-robot*. The robot being directly guided by the human supervisor is termed the *wizarded-robot* (fig. 1).

3.1 Child model

The purpose of the child model is not to realistically model a child (with or without autism), but to provide a means of expressing some of the behaviours we observed in our interactions with children in a repeatable manner. The child-robot possesses an internal model encompassing an *engagement* level and a *motivation* level, together forming the *state* of the child. The engagement represents how often the child-robot will make categorisation moves and the motivation gives the probability of success of the categorisation moves. Bound to the range $[-1, 1]$, these states are influenced by the behaviour of the wizarded-robot, and will asymptotically decay to zero without any actions from the wizarded-robot. These two states are not directly accessed by either the supervisor or the wizarded-robot, but can be observed through behaviour expressed by the child-robot: low engagement will make the robot look away from the touchscreen, and the speed of the categorisation moves is related to the motivation (to which gaussian noise was added). There is thus incomplete/unreliable information available to both the wizarded-robot and the supervisor, making the task non-trivial.

The influence of the wizarded-robot behaviour on the levels of engagement and motivation are described below (section 3.2). In addition to this, if a state is already high and an action from the wizarded-robot further increases it, then there is a chance that this level will sharply decrease, as an analogue of child-robot *frustration*. When this happens, the child-robot will indicate this frustration verbally (uttering one of eight predefined strings). The reason this mechanism is required is that it prevents a straightforward engagement and motivation maximisation strategy, thus better approximating the real situation, and requiring a more complex strategy to be employed by the supervisor.

3.2 Wizarded-robot control

The wizarded-robot is controlled through a Graphical User Interface (GUI) and has access to multiple variables characterising the state of the interaction. The wizarded-robot has a set of four actions, which each have a button in the GUI:

- Prompt an Action: Encourage the child-robot to do an action.
- Positive Feedback: Congratulate the child-robot on making a good classification.
- Negative Feedback: Supportive feedback for an incorrect classification.
- Wait: Do nothing for this action opportunity, wait for the next one.

The impact of the action on the child-robot depends on the internal state and the type of the last child-robot move: good, bad, or done (meaning that feedback has already been given for the last move and supplementary feedback is not necessary). A prompt always increases the engagement, a wait has no effect on the child-robot's state, and the impact of positive and negative feedback depends on the previous child-robot move. Congruous feedback (positive feedback for correct moves; negative feedback for incorrect moves) results in an increase in motivation, but incongruous feedback can decrease both the motivation and the engagement of the child-robot. The supervisor therefore has to use congruous feedback and prompts, whilst being careful not to use them too often, to prevent the child-robot becoming frustrated. A 'good' strategy would keep the engagement and motivation high, leading to an increase in performance of the child-robot in the categorisation task.

Through the GUI, the supervisor has access to *observed states* (noisy estimations of the child-robot state), and information about the interaction history: number of moves, child-robot performance, time since last child-robot and wizarded-robot actions, type of the last child-robot move, and elapsed time. However the supervisor can not control the wizarded-robot directly, actions can only be executed only at specific times triggered by the wizarded-robot. Two seconds after each child-robot action, or if nothing happens in the interaction for five seconds, the wizarded-robot proposes an action to the supervisor by displaying the action's name and a countdown before execution. Only after this proposition has been done can the supervisor provide feedback to the wizarded-robot. If the supervisor does nothing in the following three seconds, the action proposed by the wizarded-robot is executed. This mechanism allows the supervisor to passively accept a suggestion made by the wizarded-robot or actively make an *intervention* by selecting a different action and forcing the wizarded-robot to execute it.

3.3 Learning algorithm

The two robot controllers used for the study were a learning controller and a non-learning random action selection controller. The learning algorithm used was a Multi-Layer Perceptron, trained with back propagation (five input, six hidden and four output nodes): after each new decision from the supervisor, the network was fully retrained with all the previous state-action pairs and the new one.

3.4 Participants

In WoZ scenarios, the wizard is typically a technically competent person with previous experience controlling robots. As such, to maintain consistency with the target user group, the participants for this study (assuming the role of the supervisor) are taken from a robotics research group. Ten participants were used (7M/3F, age *M*=29.3, 21 to 44, *SD*=4.8 years).

3.5 Hypotheses

To evaluate the validity of our method and the influence of such an approach, four hypotheses were devised:

H1 A 'good' supervisor (i.e. keeping the motivation and engagement of the child-robot high) will lead to a better child-robot performance.
H2 When interacting with a new system, humans will progressively build a personal strategy that they will use in subsequent interactions.
H3 Reducing the number of interventions required from a supervisor will reduce their perceived workload.
H4 Using a learning wizarded-robot allows the supervisor to achieve similar performance with fewer interventions when compared to the same scenario with a non-learning wizarded-robot.

3.6 Interaction Protocol

Each participant experienced both robot controllers, with the order changed between participants to control for any ordering effects. In *Condition LN* the participants first interact with the learning wizarded-robot, and then with the non-learning one; in *Condition NL* the participants first interact with the non-learning wizarded-robot, and then the learning robot. Participants were randomly assigned to one of the two conditions.

The interactions took place on a university campus in a dedicated experiment room. Two Aldebaran Nao robots were used; one robot had a label indicating that it was the *Child-Robot*. The robots face each other with a touchscreen between them, and participants assuming the role of the supervisor sit at a desk to the side of the wizarded-robot, with a screen and a mouse to interact with the wizarded-robot (fig. 1). The participants were able to see the screen and the child-robot.

A document explaining the interaction scenario was provided to participants. After the information had been read, a 30s video presenting the GUI in use was shown to familiarise them with it, without biasing them towards any particular intervention strategy. The participant then clicked a button to start the first interaction which lasted for 10 minutes. The experimenter was sat in the room outside of the participants' field of view. After the end of the first interaction, a post-interaction questionnaire was administered. The same protocol was applied in the second part of the experiment with another post-interaction questionnaire following. Finally, a questionnaire asking the participants to explicitly compare the two conditions was administered.

4 Results

4.1 Interaction data

The state of the child and the interaction values were logged at each step of the interaction (at 5Hz). All of the human actions were recorded: acceptance of the wizarded-robot's suggestion, selection of another action (intervention), and the states of the child-robot (motivation, engagement and performance) at this step. From this the intervention ratio was derived: the number of times a user chose a different action to the one proposed by the wizarded-robot, divided by the total number of executed actions. On average, after a first exploration phase, where the participant discovers the system, the learning robot robot has an intervention ratio lower than the non learning one (fig. 2, left)

The performance indicates the number of good categorisations executed by the child-robot minus the number of bad categorisations. A strong positive correlation (Pearson's r=0.79) was found between the average child-robot motivation and engagement and its performance.

In both conditions, the average performance in the second interaction (M_{LN-2} =38, 95% CI [36.2, 39.8], M_{NL-2}=34.8, 95% CI [30.8, 38.8]) was higher than in the first one (M_{LN-1}=29.4, 95% CI [25.3, 33.5], M_{NL-1}=24.3, 95% CI [19.4, 29.4]; Fig. 2 *left*). The 95% *Confidence Interval of the Difference of the Mean* (CIDM) for the L-NL condition is [4.1, 13.1] and for the NL-L condition is [4.0, 16.8]. However, the performance is similar when only the interaction order (first or second) is considered. The participants performed slightly better in the LN condition, but the CIDM includes zero in both cases (95% CIDM$_1$ [-1.5, 11.5], 95% CIDM$_2$ [-1.2, 7.6]). In the condition L-NL, the intervention ratio increased between the learning and non learning condition (M_{LN-1}=0.31, 95% CI [0.20, 0.42] to M_{LN-2}=0.68, 95% CI [0.66, 0.70], CIDM$_{LN}$=[0.26, 0.48]). But in the NL condition, the intervention ratio is almost identical between the two interactions but slightly lower for the learning case (M_{NL-1}=0.50, 95% CI [0.44, 0.57] to M_{NL-2}=0.46, 95% CI [0.40, 0.51], CIDM$_{NL}$ [-0.03, 0.13]). This shows that when the wizarded-robot learned, a similar performance is attained as without learning, but the number of interventions required to achieve this is lower.

4.2 Questionnaire data

The post-interaction questionnaires evaluated the participant's perception of the child-robot's learning and performance, the quality of suggestions made by the

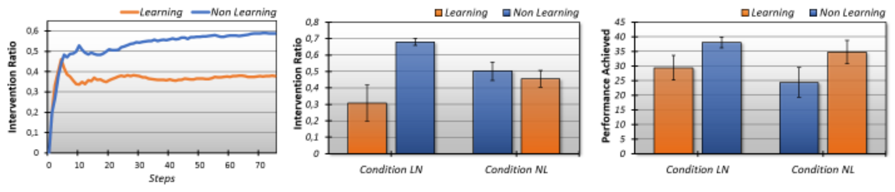


Fig. 2. (Left) evolution of intervention ratio over time for the learning and non learning cases. Intervention ratio (centre) and final performance (right) for the two conditions and the two interactions (*errors bars show 95% CI*). In condition LN participants started wizarding a robot which learns their interaction style, followed by a non-learning robot; in condition NL participants started with a non-learning robot, followed by a learning robot. Results show that a learning robot reduces the workload of the wizard, but performs equally well as a non-learning robot that needs wizarding at all times.

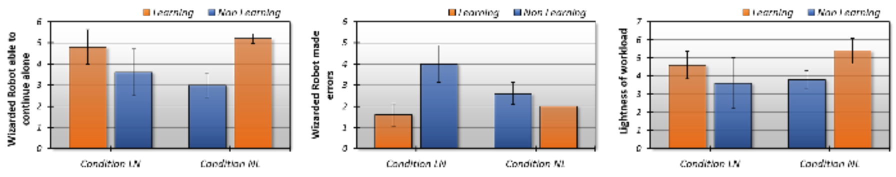


Fig. 3. Questionnaire responses (*mean and 95% CI*): increased confidence in the learning wizarded-robot over the non-learning version is apparent, as is a lower perceived workload.

wizarded-robot, and the experienced workload. All responses used seven point Likert scales.

Across the four possible interactions, the rating of the child-robot's learning was similar (M=5.25, 95% CI [4.8, 5.7]). The same effect was observed for the evaluation of the child performance (M=4.75, 95% CI [4.3, 5.2]). As the child-robot was using the same interaction model in all four conditions, this result is expected.

Participants report the wizarded-robot as more suited to operate unsupervised in the learning than in the non learning condition (M_{LN-1}=4.8, M_{LN-2}=3.6, M_{NL-1}=3, M_{NL-2}=5.2 ; CIDM for LN condition [-0.2, 2.6], CIDM for the NL condition [1.6, 2.8]).

Similarly, a trend was found showing that learning wizarded-robot is perceived as making fewer errors than the non-learning robot (M_{LN-1}=1.6, M_{LN-2}=4.0, M_{NL-1}=2.6, M_{NL-2}=2 ; CIDM for LN condition [1.3, 3.4], CIDM for the NL condition [0.1, 1.1]).

The participants tended to rate the workload as lighter when interacting with the learning robot, and this effect is much more prominent when the participants interacted with the non-learning robot first (M_{LN-1}=4.6, M_{LN-2}=3.6, M_{NL-1}=3.8, M_{NL-2}=5.4 ; CIDM for LN condition [-0.6, 2.6], CIDM for the NL condition [0.7, 2.5]).

5 Discussion

Strong support for H1 (a good supervisor leads to a better child performance) was found, a correlation between the average states (engagement and motivation) and the final performance for all of the 10 participants was observed (r=0.79). We could expect a similar effect when working with real children, but measuring these values would be a challenge.

The results also provide support for H2 (supervisors create personal strategies): all the participants performed better in the second interaction than in the first one. This suggests that participants developed a strategy when interacting with the system in the first interaction, and were able to use it to increase their performance in the second interaction. Looking in more detail at the interaction logs, it is possible to see that different people used different strategies.

H3 (reducing the number of interventions will reduce the perceived workload) is partially supported: the results show a trend for participants to rate the workload as lighter when interacting with the learning robot, and another trend between using a learning robot and the intervention ratio. However, when considering the difference of workload rating and intervention ratios between the two interactions, a positive correlation is only found for the LN condition, which could be accounted for by the initial steep learning curve for the study participants. Nevertheless, regardless of the order of the interactions, the learning robot consistently received higher ratings for lightness of workload (fig. 3).

Finally, H4 (using learning keeps similar performance, but decreases interventions) is supported: interacting with a learning robot results in a similar performance than interacting with a non-learning robot, whilst requiring fewer active interventions from the supervisor. This has real world utility, it frees some time for the supervisor, to allow her to focus on other aspects of the intervention, e.g. analysing the child's behaviour rather than focusing on the robot control.

It should be noted that the actual learning algorithm used in this study is only of incidental importance, and that certain features of the supervisor's strategies may be better approximated with alternative methods – of importance for the present work is the presence of learning at all. Future work will assess what the most appropriate machine learning approach is given the observed features of supervisor strategy from this study.

In conclusion, this paper proposed the SPARC concept (Supervised Progressively Autonomous Robot Competencies). Based on a suggestion/intervention system, this approach allows online learning for interactive scenarios, thus increasing autonomy and reducing the demands on the supervisor. Results showed that interacting with a learning robot allowed participants to achieve a similar performance as interacting with a non-learning robot, but requiring fewer interventions to attain this result. This suggests that while there is always adaptation in the interaction (leading to similar child-robot performance given the two wizarded-robot controllers), the presence of learning shifts this burden of adaptivity onto the wizarded-robot rather than on the human. This indicates that a learning robot could allow the therapist to focus more on the child than on the robot, with improved therapeutic outcomes as potential result.

Acknowledgements

This research was funded by the EU FP7 DREAM project (grant no. 611391).

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems 57(5), 469–483 (2009)
2. Baxter, P., Wood, R., Belpaeme, T.: A touchscreen-based sandtray to facilitate, mediate and contextualise human-robot social interaction. In: Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on. pp. 105–106. IEEE (2012)
3. Cakmak, M., Thomaz, A.L.: Designing robot learners that ask good questions. In: Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction. pp. 17–24. ACM (2012)
4. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. Journal of Artificial Intelligence Research 34(1), 1 (2009)
5. Dautenhahn, K.: Robots as social actors: Aurora and the case of autism. In: Proc. CT99, The Third International Cognitive Technology Conference, August, San Francisco. vol. 359, p. 374 (1999)
6. Feil-Seifer, D., Mataric, M.: B3IA: A control architecture for autonomous robot-assisted behavior intervention for children with Autism Spectrum Disorders. RO-MAN 2008. (2008), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4600687
7. Knox, W.B., Spaulding, S., Breazeal, C.: Learning social interaction from the wizard: A proposal. In: Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)
8. Riek, L.: Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. Journal of Human-Robot Interaction 1(1), 119–136 (Aug 2012)
9. Robins, B., Dautenhahn, K., Boekhorst, R.T., Billard, A.: Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills? Universal Access in the Information Society 4(2), 105–120 (Jul 2005)
10. Robins, B., Dautenhahn, K., Dickerson, P.: From Isolation to Communication: A Case Study Evaluation of Robot Assisted Play for Children with Autism with a Minimally Expressive Humanoid Robot. Conferences on Advances in Computer-Human Interactions pp. 205–211 (Feb 2009)
11. Scassellati, B., Admoni, H., Mataric, M.: Robots for use in autism research. Annual review of biomedical engineering 14, 275–294 (2012)
12. Senft, E., Baxter, P., Belpaeme, T.: Human-guided learning of social action selection for robot-assisted therapy (2015), in press for 4th Workshop on Machine Learning for Interactive Systems
13. Thill, S., Pop, C.A., Belpaeme, T., Ziemke, T., Vanderborght, B.: Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. Paladyn 3(4), 209–217 (2012)
14. Wainer, J., Dautenhahn, K., Robins, B., Amirabdollahian, F.: Collaborating with kaspar: Using an autonomous humanoid robot to foster cooperative dyadic play among children with autism. In: Humanoid Robots (Humanoids), 2010. pp. 631–638. IEEE (2010)

Providing a Robot with Learning Abilities Improves its Perception by Users

Emmanuel Senft, Paul Baxter, James Kennedy, Séverin Lemaignan and Tony Belpaeme
Centre for Robotics and Neural Systems
Plymouth University, U.K.
{emmanuel.senft, paul.baxter, james.kennedy, severin.lemaignan, tony.belpaeme}@plymouth.ac.uk

***Abstract*—Subjective appreciation and performance evaluation of a robot by users are two important dimensions for Human-Robot Interaction, especially as increasing numbers of people become involved with robots. As roboticists we have to carefully design robots to make the interaction as smooth and enjoyable as possible for the users, while maintaining good performance in the task assigned to the robot. In this paper, we examine the impact of providing a robot with learning capabilities on how users report the quality of the interaction in relation to objective performance. We show that humans tend to prefer interacting with a learning robot and will rate its capabilities higher even if the actual performance in the task was lower. We suggest that adding learning to a robot could reduce the apparent load felt by a user for a new task and improve the user's evaluation of the system, thus facilitating the integration of such robots into existing work flows.**

I. Introduction

This paper presents a study exploring the impact of providing a robot with learning capabilities on the interaction preferences and robot performance evaluations by users.

Two main approaches are reported in the literature to study human preferences about robots. The first one involves the administration of surveys where participants are asked robot-related questions with or without priming. For example, in [1] 240 subjects are asked questions about tasks that could be replaced by robots and about general attitudes toward robots, without trying to influence the participants *a priori*. Priming can also be a useful means of educating the participants before administering a questionnaire, allowing them to imagine a more constrained and plausible scenario than they otherwise would. This approach has been followed by Coeckelbergh et al. [2], who surveyed the attitudes of participants toward Robot Assisted Therapy (RAT) for children with autism spectrum disorder. The participants answered more positively, in contrast to previous studies conducted without priming, when they were first exposed to a one minute video presenting the state of the art of robotics in RAT.

The second main approach is administering a questionnaire to participants after an actual interaction with a robot. Using this method, the responses are grounded in the context of their interaction: this can diminish the generalisability of the results, but makes them more reliable. This method has been applied to explore how elderly people react to a robot with learning abilities [3].

This paper follows the real robot interaction approach, and presents additional results gathered in the experiment presented in [4]. In this study, participants interacted with a robot both with and without learning capabilities, and this manuscript reports their interaction preference and their relative performance evaluation of the two robots.

II. Methodology

The study (and therefore the methodology) is the same as in [4] where we introduced Supervised Progressively Autonomous Robot Competencies (SPARC), a means for the robot to learn from the interaction to improve its capabilities. This previous paper also reported the impact of SPARC on the performance and workload of a robot's human supervisor in a scenario inspired by RAT for children with autism spectrum disorder. In classical RAT, the robot is interacting with the child and is often controlled using the Wizard of Oz (WoZ) paradigm, i.e. fully tele-operated. This often implies a high workload on the therapist, which could be reduced by providing the robot with a supervised autonomy. As this study focuses on the interaction between the *wizarded-robot* and its supervisor, we replace the child with a robot interacting in his place (the *child-robot*) to produce consistent experimental conditions (fig. 1).

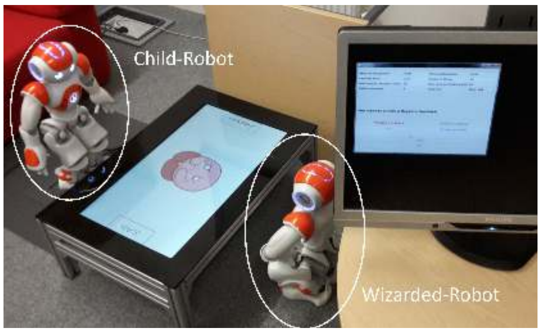


Fig. 1. Installation used for the study. The *child-robot* stands on the left, performing the task on the touchsceen, and facing the *wizarded-robot* on the centre-right. The human supervisor can control the action about to be executed by the *wizarded-robot* using the GUI on the right.

The child-robot is interacting with a touchscreen, and performs a categorisation task where it has to classify images of face as either happy or sad, with the aim of improving its performance. The wizarded-robot can execute actions (e.g. giving positive or negative feedback, waiting, or prompting the child to act), aiming to help the child-robot in its task.

The participants have to control the wizarded-robot to make it execute the correct actions to help the child-robot. This is however context dependent: actions can either improve or worsen the performance of the child-robot based on its current state. A Graphical User Interface (GUI) allows the users to control the wizarded-robot in a WoZ inspired scenario involving supervised autonomy. At specific times, the wizarded-robot makes suggestions to the supervisor who can either not react and let the action execute, or use a button to force the wizarded-robot to execute another action. A habituation phase allows the participants to become familiar with the interface and action set. If the suggestion of the robot is correct, the supervisor does not need to act to have this action executed.

The participants interacted with two systems. In the first system, the actions proposed by the wizarded-robot are random, so we expect the user to correct them most of the time. This system simulates a classical WoZ setup, which we denote the *non-learning robot*. The second system uses SPARC and includes a learning algorithm based on a Multi-Layer Perceptron using noisy observation of states as inputs and a winner-take-all on the actions as output. This system is referred to as the *learning-robot*. It is important to note that in both systems, these terms relate to the capabilities of the wizarded-robot, not the child-robot (which had constant behaviour in both systems).

The study involved ten participants (7M/3F, age M=29.3, 21 to 44, SD=4.8 years) taken from a robotic research group, as typical WoZ users are technical. Each participant interacted for 10 minutes with both systems, with the order counterbalanced. In the LN condition, participants interacted first with the learning robot then with the non-learning one, and the order is reversed in the NL condition. This paper presents and analyses the responses from the participants to the questions:

– Which *wizarded-robot* was better able to perform the task?
– Which *wizarded-robot* did you prefer supervising?

III. Results and Discussion

Overall, the participants preferred supervising the learning-robot (6 out of 10) and found it better able to perform the task (8 out of 10). Despite the limitations of the small sample size, these results suggest that providing a robot with learning capabilities can improve its perception by users and also make the users prefer supervising it. These results are consistent with previous results [4], which showed that providing a robot with learning capabilities can decrease the number of interventions required to achieve a similar performance compared to a robot without learning. The reduction in the number of interventions needed might explain the results observed here.

Breaking the results down by ordering condition (LN vs. NL) provides a more detailed perspective (fig. 2). From these separated results, the learning capability is not the only effect influencing the preferences of, and the evaluation by, the participants: the order of interaction also plays an important role. On average, the second robot is the preferred one to supervise (7 of 10) and rated as better able to perform the task (7 of 10). This ordering effect was probably due to the complexity of the system that the participants interacted with. The participants had to get used to a GUI displaying a large volume of information, and to the time constraints.

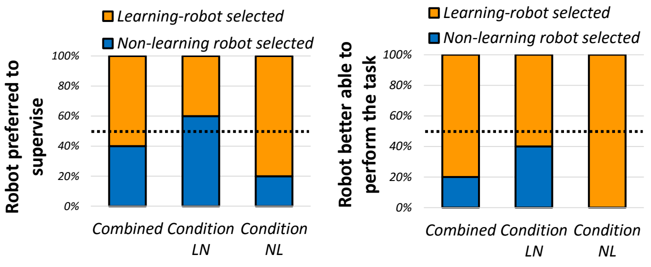


Fig. 2. Results for supervisory preference and rating of ‘preferred to supervise’ and ‘better to perform the task’ for the two conditions. The vertical bars represent the number of times that the learning robot was selected and the horizontal dotted line denotes chance (i.e. 50%).

Additionally, in 4 of the 5 cases when participants interacted with the learning robot first, they achieved a better performance during the second interaction than during the first one. Three of these participants also rated the learning robot as better able to perform the task even when it had a lower performance. This could indicate that participants can distinguish between the robot’s abilities and the performance achieved (depending also on their abilities). It could also be a reflection of the natural propensity of humans to adapt and learn through interaction. Viewed in this way, the results could be interpreted as showing that interaction with the learning robot first better equips the human to interact with the non-learning robot than vice-versa, leading to higher performance, and hence preference ratings, for the non-learning robot in the LN condition. While another potential benefit of learning robots, this interpretation will require further empirical investigation.

In this paper we presented results showing a trend towards the addition of learning capabilities to a robot helping users to cope with a new or complex task, and improving the rating of their performance by their supervisor. This is an important point for design, especially when there is a heavy workload on users such as in RAT when therapists have to use WoZ to continuously control the robot.

IV. Acknowledgements

This work is funded by the EU FP7 project DREAM (grant 611391).

References

[1] C. Ray *et al.*, “What do people expect from robots?” in *Int. Conf. on Intelligent Robots and Systems*. IEEE/RSJ, 2008.

[2] M. Coeckelbergh *et al.*, “A Survey of Expectations About the Role of Robots in Robot-Assisted Therapy for Children with ASD: Ethical Acceptability, Trust, Sociability, Appearance, and Attachment,” *Science and Engineering Ethics*, pp. 1–19, 2015.

[3] J. Hoefinghoff *et al.*, “”Yes Dear, that Belongs into the Shelf!”-Exploratory Studies with Elderly People Who Learn to Train an Adaptive Robot Companion,” in *Proc. of the 7th Int. Conf. on Social Robotics*, 2015.

[4] E. Senft *et al.*, “SPARC: Supervised Progressively Autonomous Robot Competencies,” in *Proc. of the 7th Int. Conf. on Social Robotics*, 2015.

Cognitive Architectures for Social Human-Robot Interaction

Paul Baxter, Séverin Lemaignan
Centre for Robotics and Neural Systems
Plymouth University, Plymouth, U.K.
{paul.baxter, severin.lemaignan}@plymouth.ac.uk

J. Gregory Trafton
Naval Research Laboratory
Washington DC, USA
greg.trafton@nrl.navy.mil

***Abstract*—Social HRI requires robots able to use appropriate, adaptive and contingent behaviours to form and maintain engaging social interactions with people. Cognitive Architectures emphasise a generality of mechanism and application, making them an ideal basis for such technical developments. Following the successful first workshop on Cognitive Architectures for HRI at the 2014 HRI conference, this second edition of the workshop focusses specifically on applications to social interaction. The full-day workshop is centred on participant contributions, and structured around a set of questions to provide a common basis of comparison between different assumptions, approaches, mechanisms, and architectures. These contributions will be used to support extensive and structured discussions, with the aim of facilitating the development and application of cognitive architectures to social HRI systems. By attending, we envisage that participants will gain insight into how the consideration of cognitive architectures complements the development of autonomous social robots.**

***Index Terms*—Cognitive Architectures; Cognitive Robotics; Social Human-Robot Interaction**

I. Introduction

Achieving social interactions between humans and robots is a complex task that has yet to be attained, but which is necessary for the increasing range of real-world applications for social robots. It requires an understanding of human social behaviour, and it requires the robots to use appropriate, adaptive and contingent behaviours to form and maintain these social interactions. Given that pre-programmed approaches are clearly insufficient for this problem, Cognitive Architectures provide a good alternative as they propose general mechanisms of 'intelligence' and behaviour generation.

Following the successful First Workshop on Cognitive Architectures for HRI held at the HRI conference in 2014 (Bielefeld) [1], we we are running a second edition, in which we focus specifically on cognitive architectures for *social* human-robot interaction[1]. As previously, the intention is to have the workshop be as inclusive as possible, catering both for experienced researchers in the area, but also for those for whom this may be a new topic. For all, we intend the workshop to provide a forum for discussion and the exchange of ideas. To facilitate this discussion and to provide a basis for a concrete contribution to the research community, we request short position paper contributions, and will organise a special issue (based on extended version of the position papers) after the workshop to consolidate the progress made, and provide a reference point for the community.

[1]https://sites.google.com/site/cogarch4socialhri2016/

Fig. 1. Workshop logo: cogs are typically used to represent cognition in an individual agent, this has been adapted to acknowledge the central role that interaction must play in social human-robot interactions in addition to the 'internal' cognition of the individuals.

II. Background

Cognitive Architectures are constructs (encompassing both theory and models) that seek to account for cognition (over multiple timescales) using a set of domain-general structures, mechanisms and/or processes [2]. Typically (but not necessarily) inspired by human cognition [3], the emphasis is on deriving a set of general principles of operation not constrained to a specific task or context. Despite the multitude of implementations used [4], they encourage the system designer to initially take a broader perspective than the computational mechanisms to be used and consider what sort of functionality needs to be present for the type of application, and how this relates to other cognitive competencies that are required.

For HRI, such an approach to building autonomous systems based on Cognitive Architecture would emphasise first those aspects of behaviour that are common across domains, before applying these to specific interaction contexts for evaluation. In the case of social interaction, the problems are numerous, encompassing the coordination of multiple sensory and motor modalities for the robot, the timing of proactive and reactive actions, and the recognition of interacting human states (cognitive, affective, physical, etc). Indeed, recent theoretical developments have emphasised the complex temporal coordination dynamics of human social behaviour, rather than the internal state of any individual agent [5]. This leads to questions regarding how the human should be taken into account in the action preparation/selection for the robot: explicit and individual models of performance, theory-of-mind, and/or generalised statistical models of human behaviour? It also gives rise to the question of whether and how the robot 'cognition' and actions should be directly informed by (or indeed constrained by) human psychology and physiology, with the complexity and 'non-optimal' behaviours that may result, e.g. [6]. Should our cognitive architectures for social robots be based directly on models of human behaviour, or is there no need for this? These, and related, questions are outstanding in the field and require addressing if the utility and efficacy of social robots in the real world is to be realised.

Up to now, there have only been limited and relatively isolated attempts to addressing these questions, particularly within the HRI community, with few examples of direct applications, e.g. [7]. Building on the first iteration of this workshop [1], we seek to bring together researchers who are attempting to formalise knowledge of appropriate robot behaviours for naturalistic interaction with people, typically emphasising generally applicable, holistic perspectives (i.e. striving to consider the full gamut of socially interactive behaviour rather than only individual aspects).

III. Outline of the Workshop

This workshop is aimed at researchers from a wide range of backgrounds who may be interested in applying concepts from Cognitive Architectures to their work, specifically Social HRI. Participation in this workshop is open to all interested researchers.

Prospective participants are requested to submit a 2-4 page position paper on (preferably) their work involving cognitive architectures (including the development and/or application thereof). In order to facilitate interactions and discussions at the workshop (by providing a basis for comparison), we ask that all authors additionally use their position papers to provide an answer to six guiding questions. These are as follows:

1) Why should you use cognitive architectures - how would they benefit your research as a theoretical framework, a tool and/or a methodology?
2) Should cognitive architectures for social interaction be inspired by and/or limited by models of human cognition?
3) What are the basic requirements for social interaction for a cognitive architecture?
4) How the requirements for social interaction would inform your choice of the fundamental computational structures of the architecture (e.g. symbolic, subsymbolic, hybrid, ...)?
5) What is the primary outstanding challenge in developing and/or applying cognitive architectures to social HRI systems?
6) Can you devise a social interaction scenario that current cognitive architectures would likely fail, and why would this be the case?

Submission of a position paper is not a pre-requisite for attendance, and we encourage researchers to attend the workshop even if not willing/able to submit a position paper, in order to maximise community engagement and the uptake of these concepts within the field of HRI. By attending, we envisage that participants will gain insight into how the consideration of cognitive architectures complements the development of autonomous social robots.

IV. Organisers

Paul Baxter is a researcher at Plymouth University (UK) in the Centre for Robotics and Neural Systems, and the Cognition Institute. After obtaining a PhD in Developmental Cognitive Robotics (University of Reading, UK), he worked on the EU FP7 ALIZ-E project to apply and evaluate a memory-centred perspective on cognition to social child-robot interaction. His current research work involves the development of supervised autonomous therapy robots for children with ASD (EU FP7 DREAM project), with a specific focus on cognitive robot control.

Greg Trafton is a Cognitive Scientist at the Naval Research Laboratory in Washington, DC, USA. He has degrees in both Computer Science (Trinity University) and Psychology (Princeton University) and works on Human-Robot Interaction from a cognitive modeling / architectures perspective.

Séverin Lemaignan is a researcher at Plymouth University (UK) in the Centre for Robotics and Neural Systems, and the Cognition Institute, focusing on the cognitive pre-requisites of social interaction between humans and robots. He conducts both basic work on mechanisms like the Theory of Mind, and technical realisations on interactive robots.

Acknowledgement

This work was partially supported by the EU FP7 project DREAM (grant number 611391, http://dream2020.eu/), and the Marie Sklodowska-Curie Actions of the EU H2020 project DoRoThy (grant number 657227).

References

[1] P. Baxter and J. G. Trafton, "Cognitive Architectures for Human-Robot Interaction," in *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction - HRI '14*. Bielefeld, Germany: ACM Press, 2014, pp. 504–505.

[2] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," *Cognitive Systems Research*, vol. 10, no. 2, pp. 141–160, jun 2009.

[3] R. Sun, L. A. Coward, and M. J. Zenzen, "On levels of cognitive modeling," *Philosophical Psychology*, vol. 18, no. 5, pp. 613–637, oct 2005.

[4] D. Vernon, *Artificial Cognitive Systems - A Primer*. MIT Press, 2014.

[5] E. Di Paolo and H. De Jaegher, "The Interactive Brain Hypothesis," *Frontiers in Human Neuroscience*, vol. 6, no. June, pp. 1–16, 2012.

[6] J. L. McClelland, "The Place of Modeling in Cognitive Science," *Topics in Cognitive Science*, vol. 1, no. 1, pp. 11–38, jan 2009.

[7] J. G. Trafton, L. M. Hiatt, A. M. Harrison, P. Tamborello, S. S. Khemlani, and A. C. Schultz, "ACT-R/E : An Embodied Cognitive Architecture for Human-Robot Interaction," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 30–54, 2013.

Memory-Centred Cognitive Architectures for Robots Interacting Socially with Humans

Paul Baxter
Centre for Robotics and Neural Systems
The Cognition Institute
Plymouth University, U.K.
paul.baxter@plymouth.ac.uk

Abstract—**The Memory-Centred Cognition perspective places an active association substrate at the heart of cognition, rather than as a passive adjunct. Consequently, it places prediction and priming on the basis of prior experience to be inherent and fundamental aspects of processing. Social interaction is taken here to minimally require contingent and co-adaptive behaviours from the interacting parties. In this contribution, I seek to show how the memory-centred cognition approach to cognitive architectures can provide an means of addressing these functions. A number of example implementations are briefly reviewed, particularly focusing on multi-modal alignment as a function of experience-based priming. While there is further refinement required to the theory, and implementations based thereon, this approach provides an interesting alternative perspective on the foundations of cognitive architectures to support robots engage in social interactions with humans.**

I. Introduction

The representation and handling of memory is an important feature of cognitive architectures, with a variety of symbolic and sub-symbolic representation schemes used (generally as passive storage), typically based on assumptions of modularity [1]. As such, memory is generally considered to be structurally separable from the cognitive processing mechanisms, and functions to provide these 'cognitions' with the required data.

In the memory-centred cognition perspective, memory is instead considered to be a fundamentally active process that underlies cognitive processing itself rather than being a passive adjunct [2], [3]. Based on evidence and models in neuropsychology, e.g. [4], this approach necessitates a re-examination of the organisation and functions of cognitive architectures, as outlined below (section III).

Previously, I put forward the case for the greater consideration of memory in HRI developments [5]. I argued that memory is pervasive: fundamentally involved in all aspects of social behaviour, beyond mere passive storage of information in data structures. In this brief (and relatively introspective) contribution, I expand on this point, exploring specifically the requirements of social interaction for robots, and consequently what cognitive architectures need to encompass.

II. Facets of Social Interaction

Social interaction is a complex phenomena that entails a range of abilities on the part of the interactants; indeed, there are facets of human-human social interaction that are as yet not fully understood, with the neural substrates supporting these in the individual yet to be characterised. One aspect that is commonly emphasised is the requirement for social signal processing for the individual, where behavioural cues (such as gaze, intonation, gesture, etc) should be interpreted to inform the behaviour of the observer.

One central idea emerging in the behavioural sciences is the notion of 'social contingency': the coupling and co-dependency of behaviours between interacting individuals [6]. This explicitly acknowledges the necessary role that the 'other' plays to set up the contingent behaviours, and moves away from the emphasis on social signal processing (though not discounting it). Minimal interaction paradigms provide intriguing illustrations of this: even given a low bandwidth interaction environment, there are non-trivial dynamics set up that cannot be explained by observations of an individual [7].

For social interaction generally, and in particular for this latter interacting systems perspective, there is an important role for prediction [8]. When interacting, there is an expectation that the interaction partner is also a social agent, and thus predicable in that context. Infants, for example, can use the gaze behaviour of a robot to infer that the robot is a psychological agent with which they can interact [9]. A previous study has further lent support to the idea that the imposition of expectations of social behaviour (and therefore the arising of socially contingent behaviours, in this case turn-taking) will come about if the interactants view each other as (potentially) social agents [10].

If the interaction partner (whether it is human or robot) is attributed with social agency, initially as a result of anthropomorphism for example [11], then one fundamental characteristic of social interaction between humans that will be seen is the 'chameleon effect' [12], or imitation/alignment, e.g. [13], [14], [15]. The presence of this within an interaction, as a type of contingency between the interactants (see above), could be seen as an indicator of sociality.

These phenomena, from attribution of social agency to alignment, illustrate a necessity for social robots (to a certain extent at least) to conform to human cognitive and behavioural features, as well as to their constraints, to enable predictability, consistency and contingency of robot behaviour with respect to the human(s) in the interaction.

III. Memory-Centred Cognitive Architecture

From neurospychology, the Network Memory framework [4] emphasises the central role that distributed associative cortical networks play in the organisation and implementation of cognitive processing in humans. The role of associative networks serves not only as a learning system (through Hebbian-like learning), but also as a substrate for activation dynamics. The reactivation and adaptation of existing networks combine to generate behaviour that is inherently based on prior experience.

The Memory-Centred Cognition perspective, as applied to the domain of cognitive robotics [2], seeks to extend these principles of operation: associative networks supporting activation dynamics that bring prior experience to bear on the current situation. A developmental perspective is necessary in order to do so [16]: the creation (and subsequent updating) of the associative networks must be done through the process of experience in order to form the appropriate associations between information in the present sensory and motor modalities of the robot (or system, in the case of a simulation).

Once an associative structure has been acquired, the principle mechanism at play is *priming* [2]. Priming in a memory-centred system occurs when some sub-set of the system is stimulated (from incoming sensory information for example), which causes activation to flow around the network, in turn causing parts of the network with no external stimulation to become active. Priming in this way fulfils a number of important functions. Firstly, it sets up cross-modal expectations, or the prediction of currently absent stimuli. Secondly, the priming process facilitates an integration of information across different modalities in a way that is explicitly based on prior experience (biased by the weights of the associative network).

A computational implementation of this has been applied to an account of the developmental acquisition of concepts [17]: not only was the system able to complete the task with a high success rate, but also the errors it made were consistent with those made by humans. A similar computational implementation has also been used to demonstrate how word labels for real-world objects can facilitate further cognitive processing [18]. These examples provide a glimpse of the range of cognitive processing (relevant to human cognitive processing) that can be accounted for using the memory-centred perspective.

Regarding social human-robot interaction, and in particular the notion that alignment is a fundamental feature of it (section II), the memory-centred perspective provides an intuitive, and indeed effective, account. Using exactly the same mechanism as for the concept learning study, the structure of an associative network was learned based on human behaviour (across a number of different modalities), which could then be directly used to determine the characteristics of the robot behaviour [14]. Alignment is achieved as a by-product of the way the memory-centred cognitive system operated: the associations were learned through experience, and behaviour was generated from priming (i.e. recall).

IV. Addressing Questions

From the context outlined above, I now attempt to provide answers to a set of six questions relevant to the notion of social cognitive architectures. I particularly seek to emphasise a principled-basis (as opposed to computational mechanism-basis) for cognitive architectures and for the application to social interaction.

A. Why should you use cognitive architectures - how would they benefit your research as a theoretical framework, a tool and/or a methodology?

The benefit would be in considering cognitive architectures as a set of principles (a theoretical framework), a methodology for assessing these principles, and as a tool for providing robots with autonomous intelligent behaviour.

There are in my view three specific contributions related to scientific development (as opposed to technical implementation) that cognitive architectures can make to HRI research and development, which are centred around the idea of a cognitive architecture being made up of a set of formalised hypotheses.

Firstly, in a principled manner, they allow data and theory from empirical human studies to be integrated into artificial systems. For example, if data from a psychology experiment is to be integrated, a framework for doing so is required (i.e. the architecture enables an interpretation of the data). This first point promotes the idea of a directly human-inspired/constrained architecture. Secondly, treating cognitive architectures as a set of formalised (through implementation) principles, they facilitate a comparison of different architectures at a level abstracted away from the computational systems/algorithms used, enabling a focus on the assumptions. In the presently considered case of social interaction, this is a useful facet given the as yet uncertain nature of what exactly constitutes social interaction (section II). Thirdly, the application of cognitive architectures (in robotic systems for instance) provides a means of evaluating its constituent assumptions and principles. This is related to the first point, but is focused more on the integration of empirical evidence obtained from application/experimentation with the architecture itself.

B. Should cognitive architectures for social interaction be inspired and/or limited by models of human cognition?

Following from the principles of social interaction outlined above, essentially, yes.

Taking the view that social interaction between humans is founded on the intrinsic tendency of humans to expect certain types of behaviour from their interaction partners (see section II), it becomes important to ensure that the robot will not violate expectations. In order not to violate expectation, there must necessarily be some understanding (either on the part of the system designer or learned by the system itself) of what expected human behaviour would be.

In the memory-centred cognition perspective, prior interaction history of the robot with humans would constrain its future behaviour by this experienced behaviour.

C. What are the functional requirements for a cognitive architecture to support social interaction?

The discussion of social interaction (section II) emphasised the importance of contingent behaviour, anticipation/prediction to support this, and adaptation/personalisation. In addition, it is necessary to specify appropriate timing, and embodiment-appropriate responses.

If socially-appropriate behaviour is in the eye of the (human) beholder, then the Keepon robot for example demonstrates the importance of coherence of behaviour and timing [19]. The minimally complex embodiment is convincingly responsive in a social manner, to the extent that it is seen as a communicative partner [20]. Even though it doesn't use language, only uses few degrees of freedom (in contrast to many other robots used in HRI), and is only minimally humanoid in appearance, the effect of apparent sociality is strong.

Integration of sensory and motor modalities in a temporally consistent and responsive manner (i.e. contingency), based on principles of prediction from prior experience (i.e. memory), and coherency with the robot embodiment used (c.f. Keepon example) are therefore fundamental functional requirements for a social cognitive architecture.

D. How would the requirements for social interaction inform your choice of the fundamental computational structures of the architecture (e.g. symbolic, sub-symbolic, hybrid, ...)?

Given the commitment to the memory-centred cognition perspective in this work, there is a natural fit with sub-symbolic computational structures. This provides a number of inherent advantages (section III), such as the integration of predictive behaviour from prior experience, and priming effects (within and between modalities).

However, the nature of applications in human-robot interaction (relying on language for example) means that it is not yet possible to dispense with symbol-processing systems. Nevertheless, there is in principle an effort to push the limits of sub-symbolic processing mechanisms up the processing and representation hierarchy, as revisited below (section V).

E. What is the primary outstanding challenge in developing and/or applying cognitive architectures to social HRI systems?

One of the primary challenges in the application of cognitive architectures to social interaction lies in the general lack of understanding of what is precisely involved in human-human social interaction. To a certain extent it is an attempt to find a solution to a problem that is as yet not fully characterised. This reflects on the requirements for the cognitive architectures that should engage in social interaction: if a commitment to human-like cognition/behaviour is made (see section IV-B), then what precisely are the constraints that need to be incorporated?

A more practical concern that requires further development is the provision of sensory systems for robots that can provide sufficiently complex characterisations of the (social) environment for effective decision making. There is however, in my opinion, no clear distinction between sensory systems and cognitive processing, given the necessity for interpretation of raw sensory signals (e.g. camera images) at various levels of abstraction.

F. Can you devise a social interaction scenario that current cognitive architectures would likely fail, and why?

The question is whether the application to a single domain can be generalised to other domains, which is where the benefits of cognitive architectures should come (section IV-A). As such, rather than a specific interaction scenario, I would suggest instead that autonomous sociality over variable time-scales poses challenges to current approaches and implementations.

In the short term, the challenge for social robots is to produce behaviour appropriate to the interaction context, informed by prior interaction experience, in a manner consistent with the expectations of the interacting humans. Furthermore, this socially interactive behaviour should adapt to the interaction partner over time, in terms of verbal and non-verbal behaviours for example. The technical challenges to support this in terms of sensory processing are outstanding, but there are also clear challenges in terms of the mechanisms of adaptation required (i.e. the 'cognitive' aspect). The memory-centred approach has ventured an implementation towards this problem, although the account is as yet incomplete.

Over extended periods of time, the challenges are compounded by requirements for stability. This is not just stability in terms of ensuring the system doesn't fail, but also in resolving the apparent trade-off between adaptability to new situations and robustness of the cognitive system. From the perspective of the memory-centred cognition account, the resolution to this question lies in how the formation, maintenance and manipulation of memory is handled in the system in terms of parameters and structures.

V. Outlook

The nature of the discussion above is primarily principled and theoretical rather than focused on specific computational mechanisms. Naturally I believe memory-centred cognition perspective to have a consistency and coherence that merits consideration and further development. However, it is not in its current state able to practically support all aspects of real social interactions with real people.

This is a limitation shared with many 'emergent' cognitive architecture approaches [21]: theoretically interesting and coherent perhaps, but practically limited in terms of what can be done on real systems (use of language and dialogue being good examples of this). This is partly due to an implication of the theoretical perspective: by committing to a holistic approach that emphasises the integration and interplay of many different factors (including, for example, cognition, embodiment, culture, etc), the problem is made more difficult before a computational implementation is even begun. On a practical level, the types of dynamical system (be they neural network-based or other) used are typically not fully understood, or are at least highly complex [22], e.g. in terms of conditions for stability (particularly when adaptation/learning is incorporated), which does not bode well for social robots that have to be reliable in real interactions with real people.

For these reasons, I do not believe that symbol-based approaches should (or can) be discarded, at least not for the foreseeable future. They provide the means of getting closer to actually achieving the desired behaviours in reality. Having said this, and as noted above (sec. IV-D), I remain intent on pushing the boundary between symbolic and sub-symbolic implementations 'up' the abstraction hierarchy, in a manner common with a range of other developmentally-oriented researchers [23], [24].

So, what does a memory-centred cognitive architecture look like if it is to be effectively applied to social interaction? And what does the memory-centred cognitive architecture enable in terms of social robots that would be difficult to achieve with an alternative approach? The functionality of developmental learning of cross-modal associations for prediction and action generation outlined above (section III) provides a technically difficult but in principle effective solution to the issue of learning from a vast array of potential multi-modal information in a way that is useful for action generation. This is not to say that this is the only approach (theoretical or computational) that would be capable of a similar functionality. However, this is where the second aspect, the requirement to fulfill social interaction with humans through conformity with human cognition (section II), becomes a distinguishing characteristic of the memory-centred approach.

In developing the theory, I have applied it to a range of practical systems and applications, as reviewed above (section III). For example using the same mechanism, accounts have been made of concept acquisition [17] and multi-modal robot behaviour alignment to an interaction partner [14]. Other systems using the same principles have been used to demonstrate the development of low-level sensory-motor coordination through experience [16], and the role of words in supporting new cognitive capabilities [18].

Whereas my commitment to the memory-centred cognition perspective for robotics is strong, my commitment to the specific mechanisms used is weak. I must acknowledge that there are a number of weaknesses with the various systems used, notably related to hierarchical structure/representation, and an incomplete account of temporal processing. However, in my view, this does not invalidate the theoretical approach, and merely serves to provide motivation to either find or develop a more appropriate computational implementation that fulfils all of the principles and constraints of the memory-centred cognition perspective.

Acknowledgement

This work was supported by the EU FP7 project DREAM (grant number 611391, http://dream2020.eu/).

References

[1] R. Sun, "Desiderata for Cognitive Architectures," *Philosophical Psychology*, vol. 17, no. 3, pp. 341–373, sep 2004.

[2] P. Baxter, R. Wood, A. Morse, and T. Belpaeme, "Memory-Centred Architectures: Perspectives on Human-level Cognitive Competencies," in *Proceedings of the AAAI Fall 2011 symposium on Advances in Cognitive Systems*, Arlington, Virginia, U.S.A.: AAAI Press, 2011, pp. 26–33.

[3] R. Wood, P. Baxter, and T. Belpaeme, "A Review of long-term memory in natural and synthetic systems," *Adaptive Behavior*, vol. 20, no. 2, pp. 81–103, 2012.

[4] J. M. Fuster, "Network Memory," *Trends in Neurosciences*, vol. 20, no. 10, pp. 451–9, 1997.

[5] P. Baxter and T. Belpaeme, "Pervasive Memory: the Future of Long-Term Social HRI Lies in the Past," in *Third International Symposium on New Frontiers in Human-Robot Interaction at AISB 2014*, London, UK, 2014.

[6] E. Di Paolo and H. De Jaegher, "The Interactive Brain Hypothesis," *Frontiers in Human Neuroscience*, vol. 6, no. June, pp. 1–16, 2012.

[7] E. Di Paolo, M. Rohde, and H. Iizuka, "Sensitivity to social contingency or stability of interaction? Modelling the dynamics of perceptual crossing," *New Ideas in Psychology*, vol. 26, no. 2, pp. 278–294, 2008.

[8] E. C. Brown and M. Brüne, "The role of prediction in social neuroscience," *Frontiers in Human Neuroscience*, vol. 6, no. May, pp. 1–19, 2012.

[9] A. N. Meltzoff, R. Brooks, A. P. Shon, and R. P. N. Rao, ""Social" robots are psychological agents for infants: a test of gaze following." *Neural networks*, vol. 23, no. 8-9, pp. 966–72, 2010.

[10] P. Baxter, R. Wood, I. Baroni, J. Kennedy, M. Nalin, and T. Belpaeme, "Emergence of Turn-taking in Unstructured Child-Robot Social Interactions," in *HRI'13*, Tokyo, Japan: ACM Press, 2013, pp. 77–78.

[11] B. R. Duffy, "Anthropomorphism and the Social Robot," *Robotics and Autonomous Systems*, vol. 42, pp. 177–190, 2003.

[12] T. L. Chartrand and J. A. Bargh, "The Chameleon Effect: the perception-behavior link and social interaction," *Journal of Personality and Social Psychology*, vol. 76, no. 6, pp. 893–910, 1999.

[13] K. Dautenhahn and A. Billard, "Studying robot social cognition within a developmental psychology framework," in *Third European Workshop on Advanced Mobile Robots (Eurobot'99)*, Zurich, Switzerland, 1999, pp. 187–194.

[14] P. E. Baxter, J. de Greeff, and T. Belpaeme, "Cognitive architecture for humanrobot interaction: Towards behavioural alignment," *Biologically Inspired Cognitive Architectures*, vol. 6, pp. 30–39, 2013.

[15] A.-L. Vollmer, K. J. Rohlfing, B. Wrede, and A. Cangelosi, "Alignment to the Actions of a Robot," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 241–252, 2015.

[16] P. Baxter and W. Browne, "Memory as the substrate of cognition: a developmental cognitive robotics perspective," in *Proceedings of the Tenth International Conference on Epigenetic Robotics*, Örenäs Slott, Sweden, 2010, pp. 19–26.

[17] P. Baxter, J. D. Greeff, R. Wood, and T. Belpaeme, ""And what is a Seasnake?": Modelling the Acquisition of Concept Prototypes in a Developmental Framework," in *International Conference on Development and Learning and Epigenetic Robotics*. San Diego, USA: IEEE Press, 2012, pp. 1–6.

[18] A. F. Morse, P. Baxter, T. Belpaeme, L. B. Smith, and A. Cangelosi, "The Power of Words," in *Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*. Frankfurt am Main, Germany: IEEE Press, 2011, pp. 1–6.

[19] H. Kozima and C. Nakagawa, "Social Robots for Children: Practice in Communication-Care," in *AMC'06*. Istanbul, Turkey: IEEE Press, 2006, pp. 768–773.

[20] A. Peca, R. Simut, H.-L. Cao, and B. Vanderborght, "Do infants perceive the social robot Keepon as a communicative partner?" *Infant Behavior and Development*, vol. in press, 2015.

[21] D. Vernon, G. Metta, and G. Sandini, "A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 151–180, 2007.

[22] R. D. Beer, "On the Dynamics of Small Continuous-Time Recurrent Neural Networks," *Adaptive Behavior*, vol. 3, no. 4, pp. 469–509, 1995.

[23] L. B. Smith, "Cognition as a dynamic system: principles from embodiment," *Developmental Review*, vol. 25, pp. 278–298, 2005.

[24] A. Cangelosi, *et al*, "Integration of Action and Language Knowledge: A Roadmap for Developmental Robotics," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 167–195, 2010.

Period 3

Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Project No. 611391

DREAM
Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

TECHNICAL REPORT
A Guide to Using systemGUI

Date: **18/01/2017**

Technical report lead partner: **Plymouth University**

Primary Author: **James Kennedy**

Revision: **1.0**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Summary

The purpose of this technical report is to provide a guide for all technical partners and the end users of the systemGUI developed as part of WP6. Technical partners may need this information in order to perform tests with an expanded subset of the integrated DREAM system. Therapists will need to understand how the systemGUI interface is used in order to run planned experiments effectively. This report is based on the current systemGUI at the production date; details may change to align with the GUI changes in the future.

Principal Contributors

The main authors of this document are as follows (in alphabetical order).

Hoang-Long Cao, VUB
Pablo Gomez, VUB
James Kennedy, Plymouth University
Emmanuel Senft, Plymouth University

Revision History

Version 1.0 (J.K. 18-01-2017)
Outline of first draft.

Version 1.1 (J.K. 18-01-2017)
First version complete.

1 Overall Vision

The GUI is split into four sections:

1. Actions to be performed between scripts (Section 2)
2. Actions to be performed when running a script (Section 3)
3. Child history information (Section 4)
4. Developer console (not covered in this document)

To switch between these sections, use the tabs as highlighted in Figure 1. Some buttons will remain greyed-out until certain elements have been loaded - see the relevant section of this report for further details.

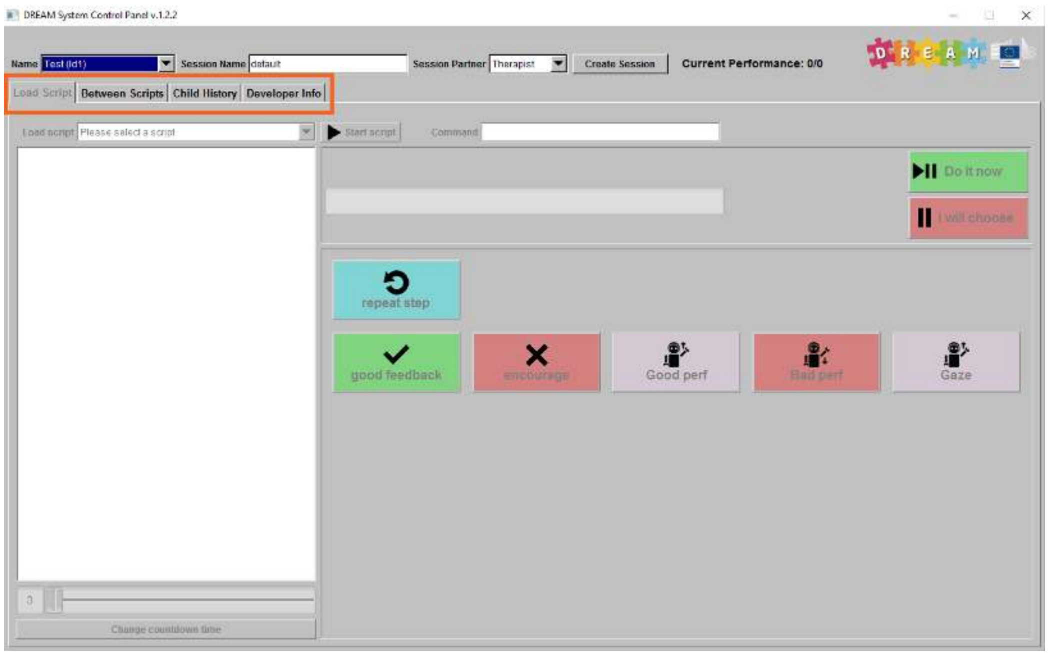


Figure 1: View of systemGUI on launch. Tabs to move between the main interaction views are highlighted by the orange box.

2 Off-Script Actions

Figure 2 shows the off-script actions pane. The robot must be connected correctly to the system in order to perform these actions. Clicking a button will trigger the corresponding robot behaviour. This tab is not available while a script is running, but can be used when a script is stopped, or when a script step is manually stopped by the therapist using the 'I will choose' button (more details in Section 3). These buttons are to be used by the therapist between scripts for maintaining the child's engagement, or in scripts to regain the child's focus if they are not engaging.

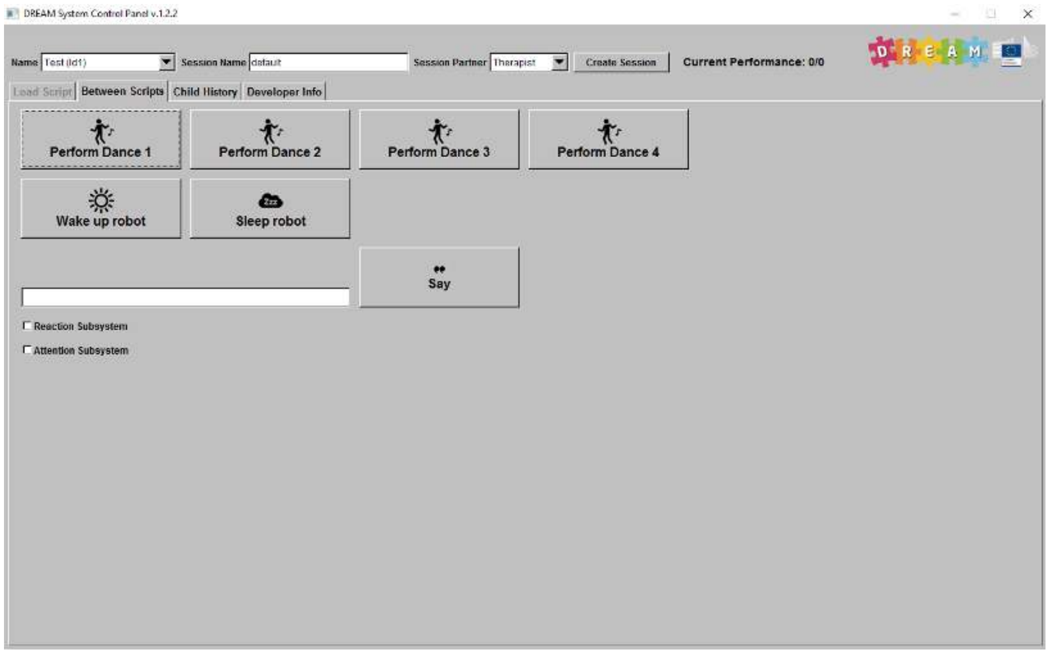


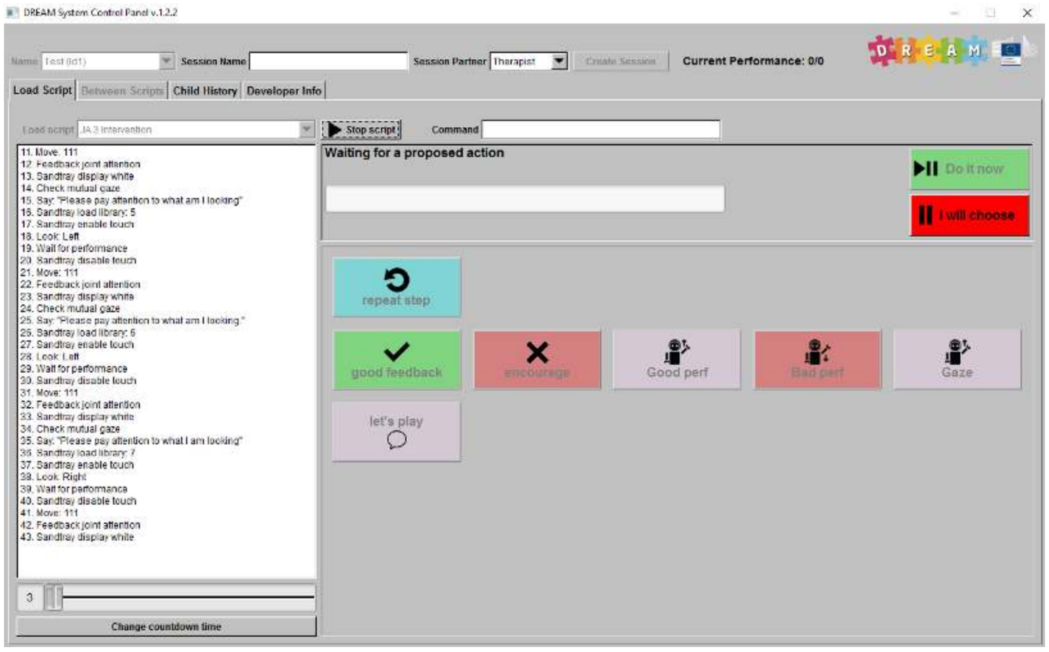
Figure 2: View of off-script actions tab.

3 Running Scripts

The script pane, as shown in Figure 3 is where most of the therapy will be conducted from. The interaction model is based on supervised autonomy, as described in DREAM deliverable D6.3.3. To start an intervention:

1. Select a child from the ’Name’ drop-down
2. Type a name for the session in the ’Session Name’ field
3. Select the ’Session Partner’: either *Therapist* or *Robot*
4. Click ’Create Session’

From this point, the child details are loaded - they can now be viewed in the ’Child History’ tab, and the ’Load script’ option is now available. After a script is selected in the ’Load script’ drop-down, the ’Start script’ button can be used to begin the script. It can be paused at any time using the ’I will choose’ red button at the right of the pane. It can be completely stopped to restart, or select a new script using the ’Stop script’ button (same location as the ’Start script’ button was). The robot will autonomously follow the script steps loaded on the left. If the proposed action is incorrect or sensory information is missed then the therapist can override the robot behaviour using the ’I will choose’ button on the right. This pauses the script and enables the ’Between Scripts’ tab and the lower part of the script buttons. These can now be used to execute robot behaviours until the ’Back to script’ button is pressed (same location as ’I will choose’). The buttons available will depend on the script loaded - this is so the right subset of the robot behaviours is displayed to keep things simple (choosing from 6 or 7 options is easier than from 40+). The ’Do it now’ button can be used to instantly execute the current proposed robot behaviour.



Waiting for a proposed action

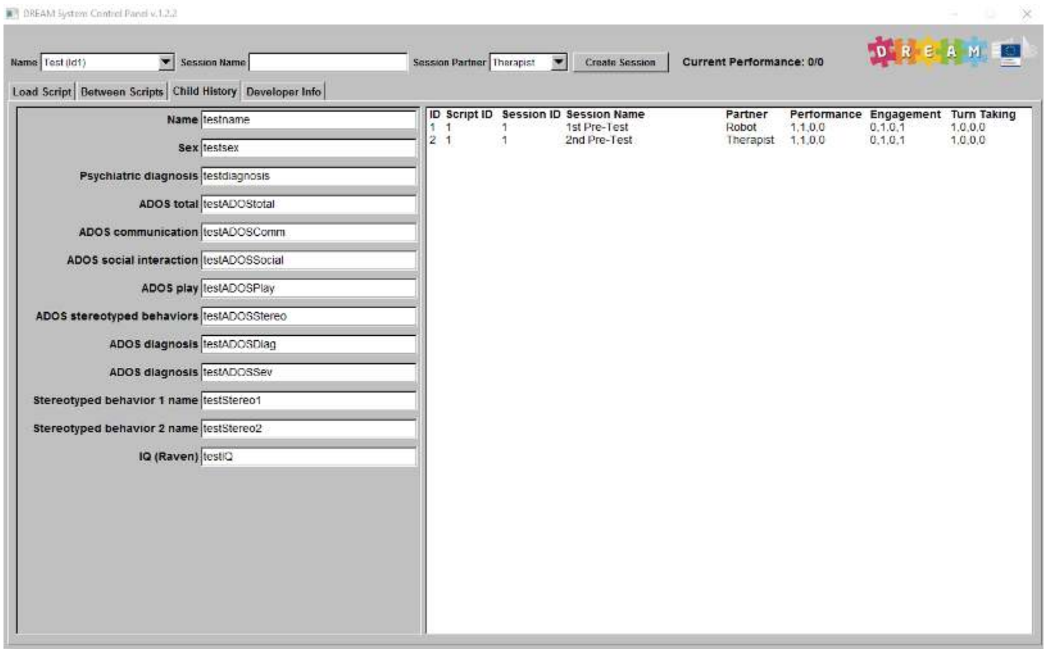
```
11. Move: 111
12. Feedback joint attention
13. Sandtray display white
14. Check mutual gaze
15. Say: "Please pay attention to what am I looking"
16. Sandtray load library: 5
17. Sandtray enable touch
18. Look: Left
19. Wait for performance
20. Sandtray disable touch
21. Move: 111
22. Feedback joint attention
23. Sandtray display white
24. Check mutual gaze
25. Say: "Please pay attention to what am I looking."
26. Sandtray load library: 6
27. Sandtray enable touch
28. Look: Left
29. Wait for performance
30. Sandtray disable touch
31. Move: 111
32. Feedback joint attention
33. Sandtray display white
34. Check mutual gaze
35. Say: "Please pay attention to what I am looking"
36. Sandtray load library: 7
37. Sandtray enable touch
38. Look: Right
39. Wait for performance
40. Sandtray disable touch
41. Move: 111
42. Feedback joint attention
43. Sandtray display white
```

Figure 3: View of on-script actions tab after a script has been loaded.

4 Child History/Data Storage

This view provides details for the child in the current session; see Figure 4. The details will not be loaded here until a child is selected from the 'Name' drop-down and 'Create Session' has been clicked. In the left pane are the details stored in the user model by the therapists for each child (these details are pre-entered by the therapists using the userModelCreator.exe tool available in the DREAM SVN /release/tools directory produced by J.K., PLYM). In the right pane are historical intervention details. All of these pieces of information are read from the user model file, stored in the DREAM SVN /release/components/userModel/config/userdata directory. Each user has their own .user file with the pre-entered and intervention data stored. These files therefore hold a primary means of evaluating the interventions and should never be manually modified while experiments are ongoing, they should be regularly backed up, and also be treated as confidential.

Before running an experiment, it is important that the intervention history is cleared for any users that have been used as part of testing (this should be done manually).



Name: Test (Id1)
Session Name:
Session Partner: Therapist
Current Performance: 0/0

Name	testname
Sex	testsex
Psychiatric diagnosis	testdiagnosis
ADOS total	testADOStotal
ADOS communication	testADOSComm
ADOS social interaction	testADOSSocial
ADOS play	testADOSPlay
ADOS stereotyped behaviors	testADOSStereo
ADOS diagnosis	testADOSDiag
ADOS diagnosis	testADOSSev
Stereotyped behavior 1 name	testStereo1
Stereotyped behavior 2 name	testStereo2
IQ (Raven)	testIQ

ID	Script ID	Session ID	Session Name	Partner	Performance	Engagement	Turn Taking
1	1	1	1st Pre-Test	Robot	1,1,0,0	0,1,0,1	1,0,0,0
2	1	1	2nd Pre-Test	Therapist	1,1,0,0	0,1,0,1	1,0,0,0

Figure 4: View of child history tab after a session has been loaded.

Towards "Machine-Learnable" Child-Robot Interactions: the PInSoRo Dataset

Séverin Lemaignan[1], James Kennedy[1], Paul Baxter[2] and Tony Belpaeme[1]

***Abstract*— Child-robot interactions are increasingly being explored in domains which require longer-term application, such as healthcare and education. In order for a robot to behave in an appropriate manner over longer timescales, its behaviours should be coterminous with that of the interacting children. Generating such sustained and engaging social behaviours is an on-going research challenge, and we argue here that the recent progress of deep machine learning opens new perspectives that the HRI community should embrace. As an initial step in that direction, we propose the creation of a large open dataset of child-robot social interactions. We detail our proposed methodology for data acquisition: children interact with a robot *puppeted* by an expert adult during a range of playful face-to-face social tasks. By doing so, we seek to capture a rich set of human-like behaviours occurring in natural social interactions, that are explicitly mapped to the robot's embodiment and affordances.**

I. Machine Learning: the Next Horizon for Social Robots?

While the family of *recurrent neural networks* have repeatedly made the headlines over the last few years with impressive results, notably in image classification, image labelling and automatic translation, they have been largely ignored in many other fields so far as they are perceived to require very large datasets (hundreds of thousands to millions of observations) to actually build up useful capabilities. Even though neural networks have demonstrated compelling results in open-ended, under-defined tasks like image labelling, they did not stand out as attractive approaches to problems involving high dimensions with relatively small datasets available – like human-robot social interactions.

Besides, if one considers "social interactions" to also entail joint behavioural dynamics, and therefore, some sort of temporal modeling, neural networks look even less enticing as time is notably absent from most of the tasks which neural networks have been successful at.

In 2015, the Google DeepMind team demonstrated how a convolutional recurrent neural network could learn to play the game Break-Out (amongst 48 other Atari games) by only *looking* at the gaming console screen [1]. This result represents a major milestone: they show that a relatively small sample size (about 500 games) is sufficient for an artificial agent to not only learn how to play (which requires an implicit model of time to adequately move the Break-Out paddle), but to also create gaming strategies that *look like* they would necessitate planning (the system first breaks bricks on one side to eventually get the ball to break-out and reach the area *above* the remaining bricks, therefore ensuring rapid progress in the game). We argue that the complexity of mechanisms that such a neural network has been able to quickly uncover and model should invite our community to question its applicability to human-robot interactions (HRI) in general, and sustained, natural child-robot interactions in particular.

However, the lack of a widespread HRI dataset suitable for the training of neural networks is a critical obstacle to this initial exploration. Therefore, as a first step, we propose a design for such a dataset, as well as a procedure to acquire it. We hope that discussions during the workshop may help in further refining this proposal.

II. Machine learning and social behaviour

Using interaction datasets to teach robots how to socially behave has been previously explored, and can be considered as an extension of the traditional learning from demonstration (LfD) paradigms to social interactions (for instance [2], [3]). Previous examples have generally focused on low-level recognition or generation of short, self-standing behaviours, including social gestures [4] and gazing behaviours [5].

Based on a human-human interaction dataset, Liu *et al.* [6] have investigated machine learning approaches to learn longer interaction sequences. Using unsupervised learning, they train a robot to act as a shop-keeper, generating both speech and socially acceptable motions. Their approach remains task-specific, and while they report only limited success, they emphasise the "life-likeness" of the generated behaviours.

Kim *et al.* [7] highlight that applying deep learning to visual scene information in an HRI scenario was successful, but that generating behaviours for the robot to be able to act in a dynamic and uncertain environment remains a challenge.

These examples show the burgeoning interest of our community for the automatic learning of social interactions, but also highlight the lack of structure of these research efforts, as further illustrated by the quasi-absence of public and large datasets of human-robot interactions. To our best knowledge, only the H^3R Explanation Corpus [8] and the Vernissage Corpus [9] have been published to date. The H^3R Explanation Corpus is a human-human and human-robot

This work has been partially supported by the EU H2020 Marie Sklodowska-Curie Actions project DoRoThy (grant 657227), the EU FP7 DREAM project (grant 611391), and the EU H2020 L2TOR project (grant 688014).

[1]Séverin Lemaignan, James Kennedy and Tony Belpaeme are with the Centre for Robotics and Neural Systems, Plymouth University, U.K. `firstname.surname@plymouth.ac.uk`

[2]Paul Baxter is with the Lincoln Centre for Autonomous Systems, School of Computer Science, University of Lincoln, U.K. `pbaxter@lincoln.ac.uk`

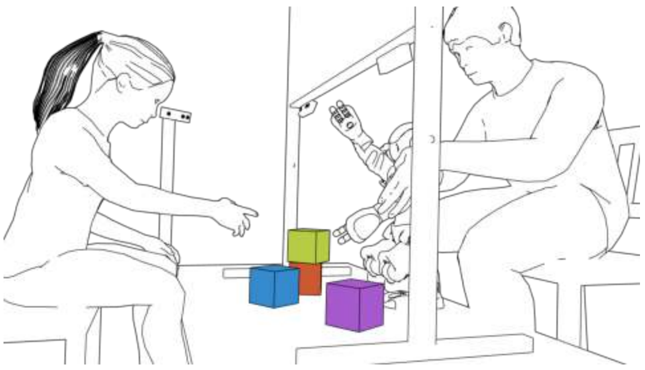


Fig. 1. The acquisition setup: a child interacts with a robot in a range of interactive tasks. The robot is physically guided by an adult expert. We record, in a synchronised manner, the full joint-states of the robots, the RGB and depth video stream from three perspectives (global scene and each of the participant faces), and the sounds (notably, the verbal interactions between the participants).

dataset focusing on a "assembly/disassembly explanation" task and includes physiological signals (22 human-robot interactions), but is not publicly available. the Vernissage Corpus includes one museum guide robot interacting with two people (13 interactions in total), with recordings and annotations of poses and speech audio (stated to be publicly available). Both these corpora are however too small for machine-learning applications.

III. The Plymouth Interacting Social Robots Dataset (PInSoRo)

A. *High-Level Aims*

The Plymouth Interacting Social Robots (PInSoRo) Dataset is intended to be a novel dataset of human-guided social interactions between children and robots. Once created, we plan to make it freely available to any interested researcher.

This dataset aims to provide a large record of social child-robot interactions that are *natural*: we aim to acquire robot behaviours through corresponding human social behaviour. To this end, we propose that an expert adult will *puppet* a passive robot (Fig. 1). As such, the gestures, expressions and dynamics of the interaction are defined and acted by a human, but as he/she uses the robot body to actually perform the actions, the motions are implicitly constrained by (and thus reflect) the robot embodiment and affordances.

The interactions are supported by a range of short social tasks (described in Section III-B). Critically we propose to limit these tasks to *face-to-face* social interactions, either dyadic or triadic. This constrains the dataset to a more tractable domain, and should ensure technical feasibility. The tasks have to fulfil several key requirements:

- be *fundamentally social*, *i.e.* these tasks would make little or no sense for an agent alone;
- foster rich *multi-modal interaction*: simultaneous speech, gesture, and gaze behaviours are to be observed;
- exhibit *non-trivial dynamics*, such as implicit turn-taking;
- should cover a *broad range of interaction contexts* and situations.

While the tasks will initially be short (in order to acquire a diverse enough dataset), we believe that the captured social behaviours could also be used to inform long-term child-robot interaction. Indeed, naturalistic, rich and socially-oriented multimodal behaviour (beyond simple stereotyped and reactive behaviour) sets the expectation in the human that long-term interactions and social presence [10] can be supported by the robot. Furthermore, we expect such a dataset to allow researchers to uncover several implicit and/or micro-behaviours that, while essential for long-term natural interactions, are difficult to explicitly characterise, and therefore difficult to implement.

B. *Tasks*

We suggest an initial set of four tasks, lasting about 10 minutes each. They involve collaborative manipulation of simple objects (such as toy cubes), (acted) storytelling, and dialogue-based social gaming. The tasks are intended to be sufficiently different from one another in order to collect a variety of different behaviours, and to minimise task-dependency of the behaviours eventually learnt from the dataset. Physical manipulation of objects across the tasks is limited by the Aldebaran Nao grasping capabilities; the tasks are designed with this in mind, *e.g.* pushing objects away or to the side is possible, whereas pulling them is more difficult.

The tasks are also designed to be playful and engaging, and are derived from classic childrens' games and activities (they are directly inspired by tasks used in other child-robot interaction work, such as [11]). They are thus expected to elicit social interactions that are particularly relevant to child-robot interaction.

a) Task 1: Spatial reasoning: In this task, one partner (child or robot) has a "completed" model made from shapes. Their role is to explain to the other partner how to arrange an identical set of shapes in order to re-create the completed model. The partner with the completed model is not allowed to directly touch the shapes. This task is intended to encourage verbal communication and deictic as well as iconic gestures. It is possible to tune the difficulty of the task through, for example, providing multiple pieces with the same colour, or shape. Similar spatial tasks have been used in other HRI experiments both with adults [12] and children [13].

b) Task 2: Storytelling: The second task revolves around storytelling. To provide a context and collaborative element to the storytelling, "Story Cubes" are incorporated into the task. These cubes are like dice, but with pictures in place of numbers; the pictures serve to guide the story. The two partners are asked to invent a story together, and they take turns in throwing one (large, custom-made) die, arranging the new picture into the story line, and proceeding to tell, and act out, the unfolding story. This task is expected to primarily generate verbal interaction, accompanied by iconic gestures.

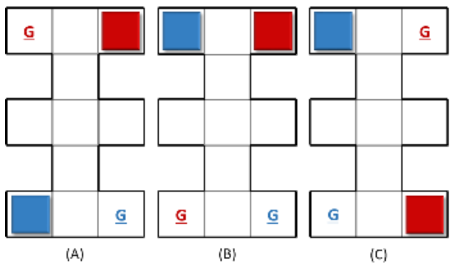


Fig. 2. A sokoban-inspired task requiring collaboration to complete given limitations in robot manual dexterity: the robots face each other across the long edge of each puzzle. Each object (red/blue square) must be pushed to its own goal (red/blue G), in three example levels of difficulty: (A) red and blue objects each simply pushed by one individual, both interactants required, but no explicit collaboration; (B) again a single object requires only a single interactant to manipulate, but some coordination is required due to shared path; (C) each object requires both interactants to manipulate, as well as coordination due to joint path.

c) Task 3: Collaborative strategising: The third proposed task is inspired by the *Sokoban* game (Fig. 2): the two partners must correctly move a set of cubes to locations within a 2D playground by only *pushing* the cubes. Due to the physical setup of the interaction (Fig. 1), the robots are essentially limited to pushing *away* the cubes, transforming the game into a necessarily collaborative activity.

d) Task 4: Party game "Taboo": The fourth proposed task involves triads in a social party game chosen not to require specific gesturing. One such game is "Taboo", a game where one must get others to guess a word without using the word itself. As the game relies only on verbal interaction, we expect all the gestures and gaze behaviour performed by the players to be social backchannel communication, and therefore of direct relevance for the dataset. Using triads is also expected to elicit a richer set of social situations. We expect it to prevent the overfitting of the model to the specific features of dyadic social interactions.

C. Methodology

The envisioned dataset would be comprised of a large number (> 50) of about 30 minutes long recordings of interactions between one child and one puppet-robot, guided by an experimenter (Fig. 1). The pair would be invited to play one or several of the proposed tasks (to be defined after initial pilots). The children would be between 8 and 14 years old. A possibly narrower age range is to be specified once the tasks are precisely defined to ensure the tasks are suitable and engaging for the target age group. Children would typically be recruited from local schools.

We propose to use a Nao robot, and to record the full joint-state of the robot over time. The robot is mostly passive: the feet are firmly fixed on the support table, and all other degrees of freedom, except for the head, are free. The head is externally controlled so that the robot gaze follows the gaze of its human puppeteer in real-time.

The choice of the Nao robot is guided by its small size, making it suitable for puppeting, and its prevalence in the HRI community, resulting in a dataset relevant for a broader academic audience. Also, since Nao is a relatively high degrees-of-freedom (DoF) robot (25 DoFs in total, 5 DoFs per arm), it mimics human kinematics reasonably well. As the motions are recorded in joint space, the dataset can be mapped to other robotic embodiments with similarly configured degrees-of-freedom.

D. Recorded Data

The dataset would comprise the following raw data:

- full 30Hz 25 DoF joint-state of the Nao robot,
- RGB + depth video stream of the scene (see Fig. 1),
- RGB + depth video stream from the child, as seen by the robot,
- speech recording.

Recorded in a fully synchronised manner, these data streams are intended to represent a useful input for many machine-learning techniques. They provide a rich dataset for a range of domains related to social child-robot interaction: from analysis of behavioural alignment between partners (via metrics like the recently proposed *Individual Motor Signature* [14]), to modeling of the dynamics of turn-taking, to the uncovering of implicit in-the-moment synchronisation mechanisms.

This would be complemented by higher-level, post-processed data:

- 68 face landmarks on the child's face, providing options for further facial analysis (like emotion recognition),
- child's skeleton extraction,
- the gaze localisation of each of the participants,
- the 3D localisation of all physical actors (child, all robot parts, cameras, table, manipulated objects),
- the verbal interaction transcripts (automatic transcript with manual verification and correction).

All these sources would be acquired via the ROS middleware (which provides the required mechanism for time synchronisation between the sources) and stored as *ROS bag* files, making it simple to replay the interactions.

As this dataset would contain sensitive data involving children, strict and specific guidelines to ensure the ethical handling of the dataset will be issued before effectively sharing any data.

IV. Discussion

A. Envisioned Applications

The recent advances in machine-learning described in the introduction raise the question of its applicability to the key challenges of artificial intelligence for robotics. Social HRI is a particularly difficult field as it encompasses a large range of cognitive skills in an intricate manner. Application domains of social HRI are typically under-defined, highly dynamic and difficult to predict.

From the data collected, a starting point for machine learning could entail a probabilistic model for reactive behaviours in a given task, *i.e.* finding for each "social cue" the possible set of responses and their probabilities. This could be made generative by using the probability distribution to seed a roulette-wheel action selection mechanism, effectively creating a probabilistic reactive controller. Whilst simplistic, this is an illustrative example of how the data may be used.

As suggested in the introduction, we also believe that such a dataset could be used to train deep neural networks. While the proposed dataset is very likely not comprehensive enough to train a neural network into an autonomous interactive system, it may be sufficiently rich to train interesting hidden units whose activations would be conditional on specific social situations. For instance, one could imagine that an adequately configured network would generate hidden units able to activate on joint gaze, or on deictic gestures. It must be emphasised that such findings are entirely hypothetical, and we only conjecture them here.

B. Possible Methodological Alternative

Several methodological issues that may impact on the quality of the interaction, the data collection, and the generalisability of results have been anticipated. As the puppeteer behaviours are bound to the embodiment of the robot, it may be that this manipulation inhibits the production of natural behaviours. A small-scale pilot will be used to explore whether or not the puppetted behaviours of the robot inhibit natural interactions with the children.

Besides, one drawback of the proposed acquisition methodology is that the puppeteer remains partially visible to the child (the hands, legs, torso are visible), which may impact the clarity of the interaction (is the child interacting with the robot or with the human behind it?). An alternative acquisition procedure is considered where the puppeteer would remotely control the robot from a different room, using Kinect-based skeleton tracking for the posture control, a head-mounted device for immersive remote vision, and a headset for remote audio. While this adds significant complexity to the acquisition procedure and increases the level of dexterity a task may require, it would provide a cleaner interaction context.

While the tasks have been designed to collect a variety of social behaviours and interaction dynamics, it may be that they are still too similar for any subsequent machine learning to acquire adequately general (*i.e.* not task-specific) behaviours for broader use. Similarly, the use of a single robot may prevent generalisation to other robotic platforms. However, it is not possible to know until algorithms have been applied and tested.

C. Long-Term Considerations

If *useful* social behaviours can be learnt from the initial dataset collected, then this would warrant further collection and exploration of the technique. Transfer to adult-adult pairs could be conducted (possibly with modification of the tasks). Child pairs performing the tasks without the robot could be used to further update behavioural models, as could human behaviours in response to learned robot models, thus providing longer-term adaptivity of behaviour.

Whilst we must acknowledge that the task-centred interactions we propose as part of the PInSoRo dataset are relatively short-term, we do argue that they are capable of simultaneously capturing a range of subtle and complex naturalistic behaviours across a range of different modalities. This type of rich behaviour (by going beyond simple stereotyped and reactive behaviour) supports the expectation in the human that they are interacting with a truly socially competent agent, thus providing the conditions in which long-term child-robot interactions could take place. The application of machine learning algorithms (particularly "deep" methods) provide an opportunity to automatically datamine the solutions to this vastly complex problem that may not be possible with hand-coded systems. Whilst this methodology may yet prove to not be *sufficient* for a complete solution, we propose that the PInSoRo dataset (and others that may follow) establishes a *necessary* foundation for the creation of socially-competent robots over long-term interactions.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] C. L. Nehaniv and K. Dautenhahn, *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions*. Cambridge University Press, 2007.

[3] Y. Mohammad and T. Nishida, "Interaction learning through imitation," in *Data Mining for Social Robotics*. Springer, 2015, pp. 255–273.

[4] Y. Nagai, "Learning to comprehend deictic gestures in robots and human infants," in *Proc. of the 14th IEEE Int. Symp. on Robot and Human Interactive Communication*. IEEE, 2005, pp. 217–222.

[5] S. Calinon and A. Billard, "Teaching a humanoid robot to recognize and reproduce social cues," in *Proc. of the 15th IEEE Int. Symp. on Robot and Human Interactive Communication*. IEEE, 2006, pp. 346–351.

[6] P. Liu, D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita, "How to train your robot - teaching service robots to reproduce human social behavior," in *Proc. of the 23rd IEEE Int. Symp. on Robot and Human Interactive Communication*, 2014, pp. 961–968.

[7] K.-M. Kim, C.-J. Nan, J.-W. Ha, Y.-J. Heo, and B.-T. Zhang, "Pororobot: A deep learning robot that plays video q&a games," in *Proc. of the AAAI 2015 Fall Symposium on AI for Human-Robot Interaction*, 2015.

[8] Y. Mohammad, Y. Xu, K. Matsumura, and T. Nishida, "The H^3R Explanation Corpus human-human and base human-robot interaction dataset," in *Proc. of the Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE, 2008, pp. 201–206.

[9] D. B. Jayagopi, S. Sheiki, D. Klotz, J. Wienke, J.-M. Odobez, S. Wrede *et al.*, "The vernissage corpus: A conversational human-robot-interaction dataset," in *Proc. of the 8th ACM/IEEE Int. Conf. on Human-Robot Interaction*. IEEE Press, 2013, pp. 149–150.

[10] I. Leite, C. Martinho, A. Pereira, and A. Paiva, "As time goes by: Long-term evaluation of social presence in robotic companions," in *Proc. of the 18th IEEE Int. Symp. on Robot and Human Interactive Communication*. IEEE, 2009, pp. 669–674.

[11] T. Belpaeme, J. Kennedy, P. Baxter, P. Vogt, E. J. Krahmer, S. Kopp *et al.*, "L2TOR - Second Language Learning Tutoring using Social Robots," in *Proc. of the First Int. Workshop on Educational Robots at the 2015 Int. Conf. on Social Robotics*, 2015.

[12] A. Sauppé and B. Mutlu, "Effective task training strategies for instructional robots," in *Proc. of the 10th Annual Robotics: Science and Systems Conference*, 2014.

[13] C. Zaga, M. Lohse, K. P. Truong, and V. Evers, "The effect of a robot's social character on children's task engagement: Peer versus tutor," in *Proc. of the 2015 Int. Conf. on Social Robotics*. Springer, 2015, pp. 704–713.

[14] P. Słowiński, C. Zhai, F. Alderisio, R. Salesse, M. Gueugnon, L. Marin *et al.*, "Dynamic similarity promotes interpersonal coordination in joint action," *Journal of The Royal Society Interface*, vol. 13, no. 116, 2016.

Development of Robot-enhanced Therapy for
Children with Autism Spectrum Disorders

Project No. 611391

DREAM
Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

TECHNICAL REPORT
WP6 Full Port Descriptions

Date: **27/01/2017**

Technical report lead partner: **Plymouth University**

Primary Author: **James Kennedy**

Revision: **1.0**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Summary

The purpose of this technical report is to provide a complete overview of the WP6 system. The system is broken down into components and all connections on ports are shown between these components. For the primitives, this document also defines the structure of the messages being passed on ports for use by other technical partners in WP4 and WP5.

Principal Contributors

The main authors of this document are as follows (in alphabetical order).

Hoang-Long Cao, Vrije Universiteit Brussel
Pablo Gomez, Vrije Universiteit Brussel
James Kennedy, Plymouth University
Emmanuel Senft, Plymouth University

Revision History

Version 1.0 (J.K. 27-01-2017)
    Outline of first draft.
Version 1.1 (J.K. 27-01-2017)
    Completed document.

1 Overview

This Technical Report outlines the design of WP6 components and their connections to one another and components from other work packages, down to the communication port level. It is intended to be used as a guide for the developers within WP6, and also the other technical partners to understand the information that WP6 components expose for their use. The components described are as follows (with the DREAM deliverable number detailing their purpose/implementation):

- attentionReactionSubsystem (D6.1, D6.2)
- scriptManager (D6.3.1+)
- deliberativeSubsystem (D6.3.1+)
- userModel (D6.3.3+)
- systemGUI (D6.3.3+)
- sandtrayServer (D6.3.3+)
- sandtrayEvent (D6.3.3+)
- actuationSubsystem (D6.4.1+)
- naoInterface (D6.4.3+)
- selfMonitoringSubsystem (D6.5; due M54)

2 Primitive Implementations

Below are definitions of the two previously undefined (at the port level) primitives that WP6 exposes to other components.

getInterventionStatus(VectorOf<int>) (Agreed in principle with WP5; ST). The complete set of new definitions are required to complete the script_id parameter. The structure may need to change if there are any situations in which we can expect two behaviours in the same script step; this is not the case in D1.1.).

getInterventionStatus(script_id, script_step, expected_behaviour_id, expected_behaviour_parameter, expected_behaviour_time_window, on_script, script_type_id)

Index	Description	Possible Values	Value Translation
0	script_id	0	between scripts/script not started
		1	joint attention 1
		2	joint attention 2
		3	joint attention 3
		..	more tbc
1	script_step	>=0	script step indicator
2	expected_behaviour_id	0	no child behaviour expected for script step
		1	child perform good sandtray move
		2	child touch sandtray image
		3	child does not touch sandtray
		4	child touch robot-owned sandtray image
		5	look right
		6	look left
		7	point left
		8	point right
		9	no movement
		10	child speaks
		11	hand wave
		12	hands covering eyes
		13	hands on head
		14	fly
		15	drive car
		16	drink/smell
		17	new complex traj 1
		18	new complex traj 2
		19	new complex traj 3
		20	new complex traj 4
		21	Knocking
3	expected_behaviour_parameter	-1	no parameter
		>=0	sandtray image id
4	expected_behaviour_time_window	>=0	time in ms for behaviour to occur within (-1 means infinity)
5	on_script	0	off script
		1	on script
6	script_type_id	0	turn taking
		1	imitation
		2	joint attention
		3	pattern
		4	sharing information

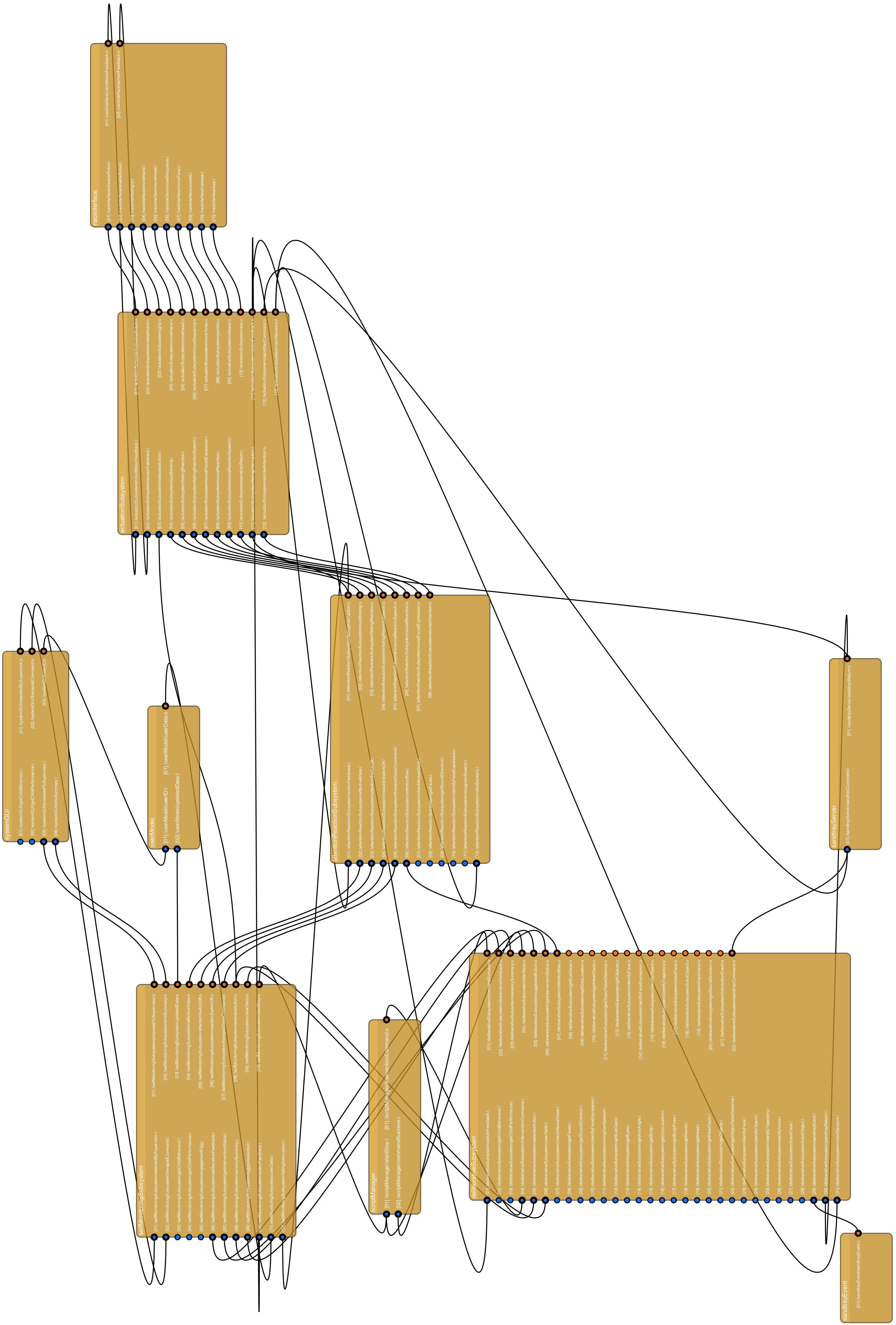
interactionEvent(VectorOf<int>) (Agreed in principle with WP5; ST). The primary intention of this primitive is to transmit relevant sandtray events for calculation of child performance from WP6 to WP5, however it has designed to be flexible for other purposes if required at a later date).

interactionEvent(type_of_event_id, event_parameter)

Index	**Description**	**Possible Values**	**Value Translation**
0	type_of_event_id	0	good sandtray move
		1	bad sandtray move
		2	touch on sandtray image
		3	touch off sandtray image
		4	child touch robot-owned sandtray image
1	event_parameter	-1	no parameter
		>=0	sandtray image id

3 Component and Port Descriptions

This section includes a graphical representation of all connections with WP6 specified to the port level. Below this are tables for each of the components in WP6, describing the component, the ports, the port directions, and the data formatting.


naoInterface
actuationSubsystem
attentionReactionSubsystem
systemGUI
userModel
sandtrayServer
selfMonitoringSubsystem
scriptManager
deliberativeSubsystem
sandtrayEvent

Component Name	/naoInterface			
Functionality	Translates action primitives into robot-specific commands			
Primitives implemented	grip()			
	moveHand(handDescriptor, x, y, z, roll)			
	moveHead (x, y, z)			
	moveSequence(sequenceDescriptor)			
	moveTorso (x, y, z)			
	release()			
	say(text, tone)			
	enableRobot();			
	disableRobot();			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	/actuationSubsystem/disableRobot:i	BufferedPort<VectorOf<int>>	Actuate	i
	/actuationSubsystem/enableRobot:i	BufferedPort<VectorOf<int>>	Actuate	i
	/actuationSubsystem/grip:i	BufferedPort<VectorOf<int>>	Actuate	i
	/actuationSubsystem/moveHand:i	BufferedPort<VectorOf<double>>	Actuate	i
	/actuationSubsystem/moveTorso:i	BufferedPort<VectorOf<double>>	Actuate	i
	/actuationSubsystem/release:i	BufferedPort<VectorOf<int>>	Actuate	i
	/actuationSubsystem/say:i	BufferedPort<Bottle>	Actuate	i
	/actuationSubsystem/moveSequence:i	BufferedPort<VectorOf<int>>	Actuate	i
	/actuationSubsystem/moveHead:i	BufferedPort<VectorOf<double>>	Actuate	i
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/naoInterface/pointAt:i	BufferedPort<VectorOf<float>>	Actuate	i
	/naoInterface/sensorFeedback:o	BufferedPort<Bottle>	Actuate	o
	/naoInterface/robotMotorFeedback:o	BufferedPort<Bottle>	Actuate	o
Date of submission core	**extended delivery only**			
Date of submission extended	**18/11/2016**			

ins	10
outs	2

Component Name	/selfMonitoringSubsystem			
Functionality	Monitors activity during the intervention and the intended behaviour of the robot. It checks ethical limitations. Communicates with the therapist through a GUI.			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	/selfMonitoringSubsystem/getChildBehaviour:i	BufferedPort<VectorOf<double>>	CBS (WP5)	i
	/selfMonitoringSubsystem/engagementFlag:i	BufferedPort<VectorOf<double>>	CBS (WP5)	i
	/selfMonitoringSubsystem/getChildPerformance:i	BufferedPort<VectorOf<double>>	CBS (WP5)	i
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/selfMonitoringSubsystem/actionFeedback:i	BufferedPort<Bottle>	Actuate	i
	/selfMonitoringSubsystem/fallingInterruption:i	BufferedPort<VectorOf<int>>	ARS	i
	/selfMonitoringSubsystem/getInterventionStatus:i	BufferedPort<VectorOf<int>>	CC (Delib)	i
	/selfMonitoringSubsystem/deliberativeFeedback:i	BufferedPort<Bottle>	Delib	i
	/selfMonitoringSubsystem/sensorySummary:i	BufferedPort<Bottle>	Delib	i
	/selfMonitoringSubsystem/suggestedAction:i	BufferedPort<Bottle>	Delib	i
	/selfMonitoringSubsystem/selectedBySupervisor:i	BufferedPort<Bottle>	GUI	i
	/selfMonitoringSubsystem/therapistCommand:i	BufferedPort<Bottle>	GUI	i
	/selfMonitoringSubsystem/userData:i	BufferedPort<Bottle>	UM	i
	/selfMonitoringSubsystem/affectiveState:o	BufferedPort<Bottle>	ARS	o
	/selfMonitoringSubsystem/attentionSwitchOff:o	BufferedPort<VectorOf<int>>	ARS	o
	/selfMonitoringSubsystem/reactionSwitchOff:o	BufferedPort<VectorOf<int>>	ARS	o
	/selfMonitoringSubsystem/therapistGazeCommand:o	BufferedPort<VectorOf<double>>	ARS	o
	/selfMonitoringSubsystem/selectedAction:o	BufferedPort<Bottle>	Delib, Actuate	o
	/selfMonitoringSubsystem/userDelib:o	BufferedPort<Bottle>	Delib	o
	/selfMonitoringSubsystem/proposedToSupervisor:o	BufferedPort<Bottle>	GUI	o
	/selfMonitoringSubsystem/smsSummary:o	BufferedPort<Bottle>	GUI	o
	/selfMonitoringSubsystem/startStop:o	BufferedPort<VectorOf<int>>	Script	o
	/selfMonitoringSubsystem/updatedData:o	BufferedPort<Bottle>	UM	o
Date of submission core	**18/11/2016**			
Date of submission extended	**31/01/2018**			

ins	12
outs	10

Component Name	/attentionReactionSubsystem			
Functionality	Given inputs from sensoryInterpretation component and other cognitive controller subsystems, it outputs attention data for the Actuation subsystem (where the moveHead( ) primitive is implemented). Additionally, it ensures that the robot can handle the real time challenges of its environment appropriately taking care of small motions, appropriate eye blinking, whole body motion during gesturing and head motion, recovering from falls, and appropriately reacting to affective displays by young users.			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	/attentionReactionSubsystem/checkMutualGaze:i	BufferedPort<VectorOf<int>>	SI (WP4)	i
	/attentionReactionSubsystem/getFaces:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/attentionReactionSubsystem/getSoundDirection:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/attentionReactionSubsystem/identifyFaceExpression:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/attentionReactionSubsystem/recognizeSpeech:i	BufferedPort<Bottle>	SI (WP4)	i
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/attentionReactionSubsystem/actionFeedback:i	BufferedPort<Bottle>	Actuate	i
	/attentionReactionSubsystem/robotSensors:i	BufferedPort<Bottle>	Actuate	i
	/attentionReactionSubsystem/attentionBias:i	BufferedPort<VectorOf<double>>	Delib	i
	/attentionReactionSubsystem/affectiveState:i	BufferedPort<Bottle>	SMS	i
	/attentionReactionSubsystem/attentionSwitchOff:i	BufferedPort<VectorOf<int>>	SMS	i
	/attentionReactionSubsystem/reactionSwitchOff:i	BufferedPort<VectorOf<int>>	SMS	i
	/attentionReactionSubsystem/therapistGazeCommand:i	BufferedPort<VectorOf<double>>	SMS	i
	/attentionReactionSubsystem/elicitedAttention:o	BufferedPort<VectorOf<double>>	Actuate	o
	/attentionReactionSubsystem/eyeBlinking:o	BufferedPort<Bottle>	Actuate	o
	/attentionReactionSubsystem/fallingReaction:o	BufferedPort<VectorOf<int>>	Actuate	o
	/attentionReactionSubsystem/fallingReactionSpeech:o	BufferedPort<VectorOf<int>>	Actuate	o
	/attentionReactionSubsystem/socialFacialExpression:o	BufferedPort<VectorOf<int>>	Actuate	o
	/attentionReactionSubsystem/socialReaction:o	BufferedPort<VectorOf<int>>	Actuate	o
	/attentionReactionSubsystem/socialReactionSpeech:o	BufferedPort<VectorOf<int>>	Actuate	o
	/attentionReactionSubsystem/fallingInterruption:o	BufferedPort<VectorOf<int>>	SMS, Actuate	o
Date of submission core	**extended delivery only**			
Date of submission extended	**18/11/2016**			

ins	12
outs	8

Component Name	/scriptManager			
Functionality	Contains and manages the logic for stepping through the scripts as defined in D1.1			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	None			
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/scriptManager/commandSuccess:i	BufferedPort<Bottle>	Delib	i
	/scriptManager/startStop:i	BufferedPort<VectorOf<int>>	SMS, Delib	i
	/scriptManager/interventionCommand:o	BufferedPort<VectorOf<int>>	Delib	o
Date of submission core	**18/11/2016**			
Date of submission extended	**31/01/2018**			

ins	2
outs	1

Component Name	/deliberativeSubsystem			
Functionality	Takes input from the environment (including sensory information, and the therapist via the GUI) and uses the script manager to propose a robot action to the self-monitoring subsystem (which in turn passes this via the therapist using the GUI when using SPARC).			
Primitives implemented	getInterventionStatus(interventionDescriptor, stateDescriptor, cognitiveModeDescriptor)			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	/deliberativeSubsystem/getChildBehaviour:i	BufferedPort<VectorOf<double>>	CBS (WP5)	i
	/deliberativeSubsystem/getChildPerformance:i	BufferedPort<VectorOf<double>>	CBS (WP5)	i
	/deliberativeSubsystem/checkMutualGaze:i	BufferedPort<VectorOf<int>>	SI (WP4)	i
	/deliberativeSubsystem/getArmAngle:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getBody:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getBodyPose:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getEyeGaze:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getEyes:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getFaces:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getGripLocation:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getHands:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getHead:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getHeadGaze:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getObjects:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getObjectTableDistance:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getSoundDirection:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/identifyFace:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/identifyFaceExpression:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/identifyObject:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/identifyTrajectory:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/identifyVoice:i	BufferedPort<VectorOf<int>>	SI (WP4)	i
	/deliberativeSubsystem/recognizeSpeech:i	BufferedPort<Bottle>	SI (WP4)	i
	/deliberativeSubsystem/trackFace:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/trackObject:i	BufferedPort<VectorOf<double>>	SI (WP4)	i
	/deliberativeSubsystem/getEyeGaze:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/getGripLocation:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/getHeadGaze:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/getObjects:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/getObjectTableDistance:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/getSoundDirection:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/identifyFace:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/identifyFaceExpression:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/identifyObject:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/identifyTrajectory:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/trackFace:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/trackHand:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/trackObject:o	BufferedPort<VectorOf<double>>	SI (WP4)	o
	/deliberativeSubsystem/interactionEvent:o	BufferedPort<VectorOf<int>>	CBS (WP5)	o
	/deliberativeSubsystem/getInterventionStatus:o	BufferedPort<VectorOf<double>>	SI (WP4) - SMS - CBS (WP5)	o
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/deliberativeSubsystem/actionFeedback:i	BufferedPort<Bottle>	Actuate	i
	/deliberativeSubsystem/interventionCommand:i	BufferedPort<VectorOf<int>>	Script	i
	/deliberativeSubsystem/selectedAction:i	BufferedPort<Bottle>	SMS	i
	/deliberativeSubsystem/userDelib:i	BufferedPort<Bottle>	SMS	i
	/deliberativeSubsystem/sandtrayEvent:i	BufferedPort<Bottle>	sandtray	i
	/deliberativeSubsystem/sandtrayReturn:i	BufferedPort<Bottle>	sandtray	i
	/deliberativeSubsystem/robotSensors:i	BufferedPort<Bottle>	Actuate	i
	/deliberativeSubsystem/attentionBias:o	BufferedPort<VectorOf<double>>	ARS	o
	/deliberativeSubsystem/commandSuccess:o	BufferedPort<VectorOf<int>>	Script	o
	/deliberativeSubsystem/startStop:o	BufferedPort<VectorOf<int>>	Script	o
	/deliberativeSubsystem/deliberativeFeedback:o	BufferedPort<Bottle>	SMS	o
	/deliberativeSubsystem/sensorySummary:o	BufferedPort<Bottle>	SMS	o
	/deliberativeSubsystem/sandtrayCommand:o	BufferedPort<Bottle>	sandtray	o
	/deliberativeSubsystem/suggestedAction:o	BufferedPort<Bottle>	SMS	o
Date of submission core	**18/11/2016**			
Date of submission extended	**31/01/2018**			

ins	31
outs	22

Component Name	/userModel			
Functionality	Loads and updates the user model file for each child			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	None			
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/userModel/userID:i	BufferedPort<VectorOf<int>>	GUI	i
	/userModel/updatedData:i	BufferedPort<Bottle>	SMS	i
	/userModel/userData:o	BufferedPort<Bottle>	SMS	o
Date of submission core	**18/11/2016**			
Date of submission extended	**31/01/2018**			

ins	2
outs	1

Component Name	/systemGUI			
Functionality	Wizard GUI for the therapist to control the intervention and the robot			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	/systemGUI/getChildBehaviour:i	BufferedPort<VectorOf<double>>	CBS (WP5)	i
	/systemGUI/getChildPerformance:i	BufferedPort<VectorOf<double>>	CBS (WP5)	i
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/systemGUI/proposedToSupervisor:i	BufferedPort<Bottle>	SMS	i
	/systemGUI/smsSummary:i	BufferedPort<Bottle>	SMS	i
	/systemGUI/selectedBySupervisor:o	BufferedPort<Bottle>	SMS	o
	/systemGUI/therapistCommand:o	BufferedPort<VectorOf<double>>	SMS	o
	/systemGUI/userID:o	BufferedPort<VectorOf<int>>	UM	o
Date of submission core	**18/11/2016**			
Date of submission extended	**31/01/2018**			

ins	4
outs	3

Component Name	/actuationSubsystem			
Functionality	Receives inputs from other subsystems and produces outputs to the robot interface. It combines actions from the subsystems and sends the next action to perform by the robot.			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	None			
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/actuationSubsystem/elicitedAttention:i	BufferedPort<VectorOf<double>>	ARS	i
	/actuationSubsystem/eyeBlinking:i	BufferedPort<Bottle>	CC/ARS	i
	/actuationSubsystem/fallingInterruption:i	BufferedPort<VectorOf<int>>	CC/ARS	i
	/actuationSubsystem/fallingReaction:i	BufferedPort<VectorOf<int>>	CC/ARS	i
	/actuationSubsystem/fallingReactionSpeech:i	BufferedPort<VectorOf<int>>	CC/ARS	i
	/actuationSubsystem/socialFacialExpression:i	BufferedPort<VectorOf<int>>	CC/ARS	i
	/actuationSubsystem/socialReaction:i	BufferedPort<VectorOf<int>>	CC/ARS	i
	/actuationSubsystem/socialReactionSpeech:i	BufferedPort<VectorOf<int>>	CC/ARS	i
	/actuationSubsystem/sensorFeedback:i	BufferedPort<Bottle>	NI	i
	/actuationSubsystem/robotMotorFeedback:i	BufferedPort<Bottle>	NI – PI	i
	/actuationSubsystem/sandtrayReturn:i	BufferedPort<Bottle>	sandtray	i
	/actuationSubsystem/selectedAction:i	BufferedPort<Bottle>	SMS	i
	/actuationSubsystem/robotSensors:o	BufferedPort<Bottle>	Delib, ARS	o
	/actuationSubsystem/disableRobot:o	BufferedPort<VectorOf<int>>	NI – PI	o
	/actuationSubsystem/enableRobot:o	BufferedPort<VectorOf<int>>	NI – PI	o
	/actuationSubsystem/grip:o	BufferedPort<VectorOf<int>>	NI – PI	o
	/actuationSubsystem/moveHand:o	BufferedPort<VectorOf<double>>	NI – PI	o
	/actuationSubsystem/moveHead:o	BufferedPort<VectorOf<double>>	NI – PI	o
	/actuationSubsystem/moveSequence:o	BufferedPort<VectorOf<int>>	NI – PI	o
	/actuationSubsystem/moveTorso:o	BufferedPort<VectorOf<double>>	NI – PI	o
	/actuationSubsystem/pointAt:o	BufferedPort<VectorOf<float>>	NI – PI	o
	/actuationSubsystem/release:o	BufferedPort<VectorOf<int>>	NI – PI	o
	/actuationSubsystem/say:o	BufferedPort<Bottle>	NI – PI	o
	/actuationSubsystem/sandtrayCommand:o	BufferedPort<Bottle>	sandtray	o
	/actuationSubsystem/actionFeedback:o	BufferedPort<Bottle>	SMS, ARS, Delib, GUI	o
Date of submission core	**18/11/2016**			
Date of submission extended	**31/01/2018**			

ins	12
outs	13

Component Name	/sandtrayEvent			
Functionality	Manages data communication with the sandtray game engine			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	None			
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/sandtrayEvent/sandtrayEvent:o	BufferedPort<Bottle>	Delib	o
Date of submission core	**extended delivery only**			
Date of submission extended	**18/11/2016**			

ins	0
outs	1

Component Name	/sandtrayServer			
Functionality	Manages data communication with the sandtray game engine			
Primitives implemented	None			
System architecture ports used	**Port**	**Port Type**	**Comm's with**	
	None			
Other ports used	**Port**	**Port Type**	**Comm's with**	
	/sandtrayServer/sandtrayCommand:i	BufferedPort<Bottle>	Delib, Actuate	i
	/sandtrayServer/sandtrayReturn:o	BufferedPort<Bottle>	Delib, Actuate	o
Date of submission core	**extended delivery only**			
Date of submission extended	**18/11/2016**			

ins	1
outs	1

SPARC: an efficient way to combine reinforcement learning and supervised autonomy.

Emmanuel Senft
Centre for Robotics and Neural Systems
Plymouth University
Plymouth, PL4 8AA, United Kingdom
emmanuel.senft@plymouth.ac.uk

Séverin Lemaignan
Centre for Robotics and Neural Systems
Plymouth University
Plymouth, PL4 8AA, United Kingdom
severin.lemaignan@plymouth.ac.uk

Paul E. Baxter
Lincoln Centre for Autonomous Systems
University of Lincoln
Lincoln, LN6 7TS, United Kingdom
pbaxter@lincoln.ac.uk

Tony Belpaeme
Centre for Robotics and Neural Systems
Plymouth University
Plymouth, PL4 8AA, United Kingdom
iMinds – Ghent University
Department of Electronics and Information Systems
B-9052 Ghent, Belgium
tony.belpaeme@plymouth.ac.uk

Abstract

Shortcomings of reinforcement learning for robot control include the sparsity of the environmental reward function, the high number of trials required before reaching an efficient action policy and the reliance on exploration to gather information about the environment, potentially resulting in undesired actions. These limits can be overcome by adding a human in the loop to provide additional information during the learning phase. In this paper, we propose a novel way to combine human inputs and reinforcement by following the Supervised Progressively Autonomous Robot Competencies (SPARC) approach. We compare this method to the principles of *Interactive Reinforcement Learning* as proposed by Thomaz and Breazeal. Results from a study involving 40 participants show that using SPARC increases the performance of the learning, reduces the time and number of inputs required for teaching and faces fewer errors during the learning process. These results support the use of SPARC as an efficient method to teach a robot to interact with humans.

1 Introduction

To be widely used by non-technical people, robots have to be able to learn, in order to adapt their behaviour to new challenges and tasks. These robots have to acquire knowledge whilst interacting in an environment which possibly includes other people. Reinforcement Learning [14] is a machine learning algorithm specifically designed to address the issue of learning how to interact efficiently based on feedback from the environment. This learning method has already been widely applied to robots [10], however, as pointed by Knox and Stone in [9], the reward function from the environment can either not be defined for certain tasks or at least be sparse in its assignation of reward. A solution is to include a human in the learning process, moving from classical machine learning to interactive machine learning. In this framework, a human supervisor is fully integrated in the learning process and can provide additional information to the algorithm to improve the learning [4]. Furthermore, this approach also provides end users with the ability to steer the learning in the direction they desire, which can improve the robot's usability [1].

Multiple approaches have been proposed to combine human feedback and reinforcement learning. In [8], Knox et al. present the TAMER framework, designed to teach an action policy in the absence of any environmental feedback using a human to provide the missing rewards used for the reinforcement learning. Thomaz and Breazeal [16] propose to combine human and environmental rewards and use them directly as input for a Q-Learner. During their experiments, they observed that participants tried to use rewards as a way to guide the robot's actions. Consequently, they introduced a second guidance channel to guide the robot action in follow-up studies and observed better learning.

However, we argue that the lack of control over the robot's action in these methods limits the impact of the human in the learning loop. By taking inspiration from Learning from Demonstration [2, 3], the human can provide demonstration of the desired action policy, and at the same time interactively teach the robot. Following this approach, we have proposed the Supervised Progressively Autonomous Robot Competencies – SPARC [13]. This is based on the supervised autonomy framework [15]: the robot can act autonomously, but a human is supervising it to prevent undesirable actions from being executed if necessary. By adding machine learning (reinforcement learning in this case), in which the robot can learn from the human corrections and improve its action policy over time whilst only executing actions deemed appropriate by the supervisor.

This paper presents the combination of SPARC and reinforcement learning and compares it with a previously applied approach following the principles of *Interactive Reinforcement Learning* [16] using four metrics: performance, teaching time, number of inputs and risks taken while teaching. We show that in each of these metrics, SPARC leads to a significant improvement over *Interactive Reinforcement Learning*, supporting its use as a framework to teach robots in an interactive fashion in sensitive environments, such as those typically encountered in human robot interaction.

2 Methodology

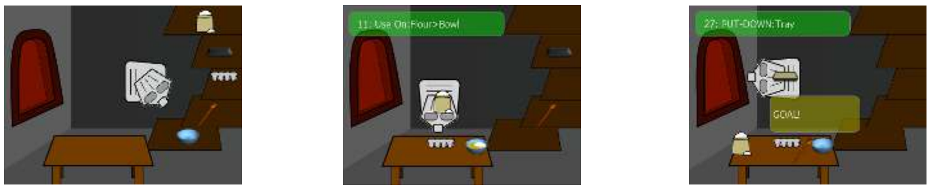
Problem specifications In this paper, we tackle the action selection problem in an environment modelled as a deterministic Markov Decision Process. An agent can execute actions changing the current state to a new one according to a fixed deterministic transition function. A limited number of states provide rewards (positive or negative), and the agent has to maximise the rewards obtained over time. Additionally, a human supervisor is present and can provide additional information to the robot to improve the learning (rewards and guidance for IRL or commands for SPARC).

Interactive Reinforcement Learning Due to its clarity, simplicity and aim to be used for human-robot interaction, the method used as a benchmark in this paper is *Interactive Reinforcement Learning* (IRL) following principles proposed in [16]. Thomaz and Breazeal proposed a first example of incorporating a human in the learning process by directly combining the reward from the environment with human rewards: a human supervisor can provide rewards which are combined to the environmental ones and used with Q-Learning. Following early studies, authors enriched the interaction with three mechanisms to improve the teaching: a guidance mechanism to direct the robot's attention to some actions (without covering the entire action space), a communication of uncertainty and an undo behaviour executing an action cancelling the previous one after a negative reward. This study uses an algorithm inspired from the one proposed by Thomaz and Breazeal and implementing these additions.

Supervised Progressively Autonomous Robot Competencies In [12] and [13], we proposed the Supervised Progressively Autonomous Robot Competencies (SPARC) as an interaction framework allowing a supervisor (human or other) to teach a robot an action policy. SPARC is centred around a suggestion/correction mechanism whereby the agent suggests actions to its supervisor which can either correct the action by selecting another one or let the action be executed after a short delay by not reacting to the suggestion. This system allows the supervisor to be totally in control of the actions being executed by the robot. A learning algorithm, here reinforcement learning, learns from the supervisor decision to improve the suggestions over time, decreasing the necessity of correcting actions and thus reducing the workload on the supervisor.

As the supervisor only provides commands and no reward to the robot, this approach is initially not designed to be used with reinforcement learning. However the constant control of the supervisor on the robot's action implies that every action executed by the robot has been implicitly or actively validated by the supervisor, so all these executed actions can receive a positive reward: we reward 0.5 for actions actively selected by the supervisor, and 0.25 for the actions passively accepted.

Evaluation task To evaluate the efficiency of combining SPARC with reinforcement learning and to compare the results with IRL, we run a study using Sophie's kitchen, the initial setup used by Thomaz and Breazeal in [16]. Participants have to teach a virtual robot how to bake a cake in the environment presented in Figure 1. The robot can pick-up, drop or use objects and can move left or right between three locations: the shelf, the table and the oven. Six main steps have to be completed to bake the cake: placing the bowl on the table, putting a first ingredient in the bowl (flour or eggs), then the other one, mixing with the spoon, emptying the bowl in the tray and finally putting the tray in the oven. As shown later, we used these six steps to evaluate participants' performance. As argued by Thomaz and Breazeal, this environment is interesting for interactive learning due to the large number of states (more than 10,000), multiple non-trivial successful action policies (minimum of 28 actions to achieve the goal) and success and failure states used to provide environmental rewards: for example, if the spoon is put in the oven, a failure state is reached, providing a negative reward, ending the current teaching episode and returning the environment to the initial state. More detailed information can be found in [16]. This environment has been reimplemented to be used in this study, and the two interaction methods are using strictly the same learning algorithm, only the way to interact changes.



(a) Step 0: Initial state. (b) Step 3: ingredients in the bowl. (c) Step 6: Success.

Figure 1: Sophie's kitchen, the environment used in the study in three different states.

Study setup The study involves 40 participants (age *M*=25.6, *SD*=10.09; 24F/16M) divided into two groups. The first group interacts with IRL and the second one with SPARC. Participants first teach the robot how to complete the task and then a testing phase, where participants' inputs are disabled, evaluates the robot behaving on its own to assess participants performance in teaching. To limit the study time, a hard limit of 25 minutes for the teaching phase has been set for both systems, but participants could move on to the testing whenever they desired.

This paper presents a subset of the results of a larger study having each participant interacting three times with each system. The full results are currently being analysed.

3 Results

As not all participants reached the goal state (i.e., a cake in the oven) during training, the performance is expressed on a scale from 0 to 6 representing how many of the 6 main steps presented in section 2 (putting the bowl on table, adding an ingredient...) are autonomously completed by the robot in the testing phase. Results on four metrics (performance, interaction time, number of failure during teaching and number of inputs given by the teacher) are presented in Table 1 and Figure 2.

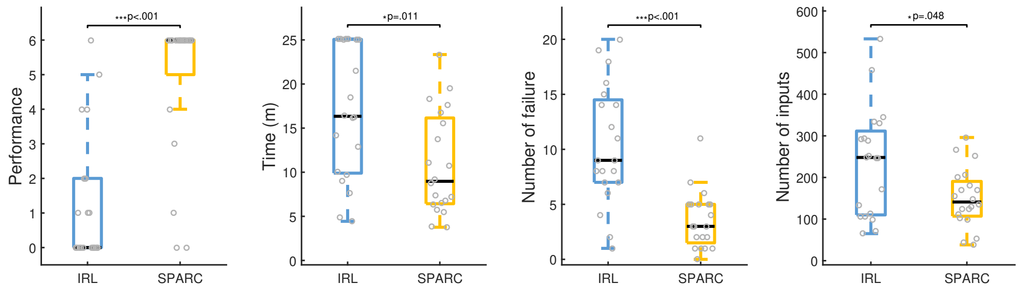


Figure 2: Comparison of the performance, interaction time, number of failures and number of inputs for the two conditions. Black horizontal bars represent medians and grey circles raw data points.

Table 1: Results of evaluation metrics for the two systems ($n = 20$). Using Wilcoxon rank sum test (results being non-normal), SPARC is significantly more efficient than IRL on all four metrics.

Metric	Median IRL	Median SPARC	Z-$value$	p-$value$	Effect size
Performance	0	6	-4.2	$< .001$	-0.67
Time (min)	16.3	8.97	2.53	0.011	0.4
Number of failure	9	3	4.06	$< .001$	0.641
Number of inputs	248	141	1.98	0.048	0.312

In this study, most of the participants using IRL did not reach a single step toward success (median of 0). This does not mean that this method cannot be used to teach an action policy: some participants reached the goal state with IRL and an expert would consistently achieve the goal state in this task. However, due to the more complex reward scheme and other challenges to interpret the trainers' rewards [5, 6, 11] not tackled by this method, participants need to have more in-depth understanding of how to interact with the algorithm to achieve success. These results support our thesis that relying only on feedback and guidance is a suboptimal method to teach a robot: even in this simple scenario, non-expert participants perform poorly. On the other hand, the median performance of 6 for SPARC shows that at least half of the participants reached the goal when interacting with SPARC and this in a shorter time, facing fewer failures during teaching and using fewer input. As such the combination of reinforcement learning and SPARC seems a more efficient teaching method.

4 Discussion

This study used a relatively simple environment: it has discrete states and a deterministic transition function. Realistic environments will be more complex and challenging. Furthermore, this simulation did not contain human interactants, but interacting with people adds two major constraints; unlike simulation we cannot train the agent for a long time before obtaining a correct action policy. In addition, as soon as the robot is used with people its behaviour has to be appropriate: suboptimal actions might have negative consequences. For this reason, the presence of a human supervisor having control over the robot's action has many advantages: it ensures that the behaviour expressed is appropriate and provides robustness against probabilistic environments, sensory errors and imperfect action policies. Whilst it is being controlled, the robot can progressively learn from the supervisor, and smoothly become more autonomous over time, reducing the workload on the supervisor. As the robot is acting in the real world and is executing a correct action policy, the need for exploration is reduced thus accelerating the learning process.

By relying on human commands and corrections, SPARC changes the teaching paradigm compared to classical interactive reinforcement learning methods. The human control over the robot's actions allows to bypass the need for users to manually assign rewards or evaluations to actions. It also uses only one-way feedbacks (selection) compared to two-way feedback in classical approaches (positive or negative reinforcement), thus preventing SPARC to face some of the challenges of human rewarding practices as described in [5, 11]. For example, a lack of feedback (absence of correction of an action) can either be direct passive support of the proposed action or if the action should have been corrected, it can be due to a slow reaction time or a desire not to interact. However, the control over the robot actions can allow supervisors to correct the trajectory when required and thus assuming a passive support in all cases where feedback is missing can still lead to efficient learning. SPARC could be combined with more classical Learning from Demonstration approaches [3] to teach lower level action policies, such as direct motor control where correct actions cannot be easily selected.

In this study, *Interactive Reinforcement Learning* achieved a poor performance with only a limited number of participants succeeding to use it to teach the robot how to bake the cake. On the other hand, SPARC achieved a high success rate in a shorter time and with fewer failures and lower teaching effort. This is consistent with [7], where authors argue that feedback channels are not an efficient method to teach an action policy from scratch, they recommend to start with Learning from Demonstration and then move to feedbacks for fine tuning. By relying on human intervention to prevent poor performance before it occurs, this paper has shown how SPARC can be usefully applied to teach an action policy while maintaining high performance, avoiding dangerous situations, and yet without overloading the human supervisor.

Acknowledgments

This work was supported by the EU FP7 DREAM project (grant no. 611391) and EU H2020 Marie Sklodowska-Curie Actions project DoRoThy (grant 657227).

References

[1] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

[4] J. A. Fails and D. R. Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003.

[5] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2625–2633, 2013.

[6] C. L. Isbell Jr, M. Kearns, S. Singh, C. R. Shelton, P. Stone, and D. Kormann. Cobot in lambdamoo: An adaptive social statistics agent. *Autonomous Agents and Multi-Agent Systems*, 13(3):327–354, 2006.

[7] T. Kaochar, R. T. Peralta, C. T. Morrison, I. R. Fasel, T. J. Walsh, and P. R. Cohen. Towards understanding how humans teach robots. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 347–352. Springer, 2011.

[8] W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009.

[9] W. B. Knox and P. Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[10] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 2013.

[11] R. Loftin, B. Peng, J. MacGlashan, M. L. Littman, M. E. Taylor, J. Huang, and D. L. Roberts. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30(1):30–59, 2016.

[12] E. Senft, P. Baxter, and T. Belpaeme. Human-guided learning of social action selection for robot-assisted therapy. In *4th Workshop on Machine Learning for Interactive Systems*, 2015.

[13] E. Senft, P. Baxter, J. Kennedy, and T. Belpaeme. SPARC: Supervised progressively autonomous robot competencies. In *International Conference on Social Robotics*, pages 603–612. Springer, 2015.

[14] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[15] S. Thill, C. A. Pop, T. Belpaeme, T. Ziemke, and B. Vanderborght. Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. *Paladyn, Journal of Behavioral Robotics*, 3(4):209–217, 2012.

[16] A. L. Thomaz and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6):716–737, 2008.

Pattern Recognition Letters
journal homepage: www.elsevier.com

Supervised Autonomy for Online Learning in Human-Robot Interaction

Emmanuel Senft[a,**], Paul Baxter[b], James Kennedy[a], Séverin Lemaignan[a], Tony Belpaeme[a,c]

[a]*Plymouth University, Drake Circus, Plymouth PL4 8AA, United Kingdom*
[b]*Lincoln Centre for Autonomous Systems, University of Lincoln, Brayford Pool, Lincoln LN6 7TS, United Kingdom*
[c]*Ghent University, imec – IDLab, Department of Electronics and Information Systems, Ghent, Belgium*

ABSTRACT

When a robot is learning it needs to explore its environment and how its environment responds on its actions. When the environment is large and there are a large number of possible actions the robot can take, this exploration phase can take prohibitively long. However, exploration can often be optimised by letting a human expert guide the robot during its learning. Interactive machine learning, in which a human user interactively guides the robot as it learns, has been shown to be an effective way to teach a robot. It requires an intuitive control mechanism to allow the human expert to provide feedback on the robot's progress. This paper presents a novel method which combines Reinforcement Learning and Supervised Progressively Autonomous Robot Competencies (SPARC). By allowing the user to fully control the robot and by treating rewards as implicit, SPARC aims to learn an action policy while maintaining human supervisory oversight of the robot's behaviour. This method is evaluated and compared to Interactive Reinforcement Learning in a robot teaching task. Qualitative and quantitative results indicate that SPARC allows for safer and faster learning by the robot, whilst not placing a high workload on the human teacher.

1. Introduction

In the not too distant future robots will be expected to have social skills, leaving the factory to interact with people in environments designed exclusively for use by humans (Fong et al., 2003). Their users will not be academics or engineers but the elderly, therapists, children or simply non-experts in technology and science. Each user will have specific needs that cannot be totally anticipated at the robot's design stage. Many researchers have argued that this issue can be best addressed by having the user involved in generating the behaviour (e.g. Gorostiza and Salichs, 2011; Hoffman, 2016). However, we cannot assume that users will have the technical knowledge required to make changes to the code controlling the robot. Therefore, we believe that robots need to have a mechanism allowing a human to teach the robot in an easy, natural and efficient manner.

One way to provide a robot with such learning capability is to use machine learning. Classic machine learning is often designed by experts to be used by experts, its interface being often too complex for people not involved in the design process (Amershi et al., 2014). Many methods also suffer from practical issues: Deep Learning (LeCun et al., 2015) relies on having large datasets to train networks, while Reinforcement Learning (Sutton and Barto, 1998) uses extensive and costly exploration to gather data points used for learning. As we aim at allowing a non-expert end-user to personalise the robot's behaviour, complex interfaces are not desirable, large dataset are not available and random exploration can lead to undesired actions by the robot. This suggests two main challenges: how to empower the user with the ability to teach the robot and how to gather safe training experiences for the robot. A solution aiming to solve these two challenges is *interactive machine learning* (Amershi et al., 2014; Fails and Olsen Jr, 2003; Olsen, 2009). In this framework, the human is part of the machine learning process. By providing ground truth labelling or guiding the agent during exploration to the interesting parts of the environment, the human can bootstrap and guide the learning. Furthermore, the human can provide more information than simply labelling the samples, bringing further improvements to the learning (Holzinger, 2016; Stumpf et al., 2007) and if enough control is provided, the human teacher can also prevent the robot from making undesirable or potentially

**Corresponding author:
e-mail: `emmanuel.senft@plymouth.ac.uk` (Emmanuel Senft)

dangerous errors.

In this paper, we present a novel approach to combine reinforcement learning with interactive machine learning following the Supervised Progressively Autonomous Robot Competencies (SPARC) method proposed in Senft et al. (2015b). By giving control of the robot's actions to a teacher, we aim to maximally use the human's knowledge and transfer it to a robot in a quick, safe and efficient manner. This method is compared to *Interactive Reinforcement Learning* (IRL), described in Thomaz and Breazeal (2008), using a study involving 40 participants interacting with both approaches in Sophie's Kitchen, the environment used to demonstrate IRL.

The reminder of the paper is organised as follows. Section 2 presents different approaches used to teach robots in an interactive fashion. We then describe the scope of the study, including our hypotheses (Section 3) and methodology (Section 4). Results are presented in Section 5 and are discussed in Section 6. We also propose guidelines for designing robots which interactively learn from people. Finally, we conclude by summarising the main results and the guidelines in Section 7.

2. Related Work

In human-robot interaction, the expected behaviour of the robot is often solely known by the users: for therapies, therapists are the experts and they know how the robot is supposed to behave when interacting with patients. For assistive robots in homes, each user has his own desires and preferences concerning the robot's behaviour. Consequently, these users have to be able to adapt the behaviour of the robot in a way which suits them without requiring technical skills. One approach to allow non-technical persons to teach a robot an action policy is *Learning from Demonstration* (Billard et al., 2008; Argall et al., 2009). In this framework, a human provides a robot with demonstrations of the expected behaviour and the robot learns the correct action policy. This methods is often used for teaching motor trajectories to a robot, but is also applicable to high level action policy learning in robotics (Taylor et al., 2011). The conventional approach consists of a set of demonstrations from the teacher followed by additional learning without supervision until reaching an appropriate action policy. However, human-robot interactions are not a static process, the learning should happen during all interactions and be interactive: the user should at all times be able to correct the robot when it selects a suboptimal action.

In *interactive machine learning* a human is included in the learning loop, allowing him to provide input during the learning process, this approach has received increased attention over the last decade. One of the main domains being extensively researched is active learning (Settles, 2010). Active learning has been used in a range of fields: from medical image classification (Chyzhyk et al., 2013) to robotics (Chernova and Veloso, 2009). In this framework, an agent has to classify points in a dataset and an 'oracle' is present and available. The oracle, often a human, can provide ground truth labelling, but its use has a cost (time or money for example) and consequently should be minimised. As such, the conventional challenge of active learning is to find how to optimise the use of the oracle to improve the learning. Multiple approaches have been tested, such as requiring labels for the points with the higher uncertainty or which categorisation would provide the best improvement of the learning.

However, as pointed out by Cakmak and Thomaz (2012), one of the main limits of active learning is that the robot is in control of the interaction: the robot takes initiative to request training data from the user, regardless of what the human wants the robot to do, potentially leading to frustration or incomprehension on the human side. For this reason, methods have been developed to give the initiative back to the human, placing the human in a teaching role. For example, when set in a reinforcement learning framework, the human teacher can provide additional feedback (Knox and Stone, 2010; Thomaz and Breazeal, 2008) and actively decides to reward or not to reward a specific action.

In human robot interactions, the robot's actions can have a real impact on the world and some actions, if executed at an incorrect moment, can create discomfort for the user or even cause physical or psychological harm. These errors can be the result of an incorrect action policy or a sensor failure for example, but they have to be prevented. When using a robot in real human-robot interaction applications, a safeguard should therefore be present to prevent the robot from executing undesirable actions, especially when working with vulnerable users, where some actions would have severely negative effects. It is on this basis that the concept of *supervised autonomy* was introduced (Thill et al., 2012): a safeguard is provided by a human supervising the robot in a semi-autonomous setup. The robot is mainly autonomous, but a human teacher has enough control over the interaction to step in at any time to correct the action about to be executed by the robot. This approach ensures that only desired actions will be executed by the robot whilst not relying completely on a human to control the robot as with Wizard of Oz (Riek, 2012). The challenge is then the incorporation of robot learning into this scheme to facilitate progressive performance improvement: this approach can be combined with interactive machine learning to let the robot learn from its errors without requiring the robot to actually make them. At the same time, the human is used to bootstrap the learning with their knowledge, but also to ensure that the robot behaviour is always appropriate. This would allow the robot to improve its behaviour over time, while reducing the frequency of human interventions, having the robot learning without needing to face the consequence of its actions.

An analogous system is predictive texting on mobile phones: as a user types a message, possible words are suggested, but the user has full control over which word to select. All the while, the algorithm learns: it adopts new words, spellings and tunes its predictive models to suit the user's particular language use and preferences. We propose a similar mechanism for Human-Robot Interaction, and in this context we introduce the Supervised Progressive Autonomous Robot Competencies (SPARC) (Senft et al., 2015a,b).

By combining interactive machine learning and supervised autonomy, SPARC provides an agent with online learning whilst keeping the control of the agent's actions in the user's hand. This method based on a suggestion/correction mechanism allows the robot to adapt its behaviour to the user whilst ensuring, due to the presence of the human teacher, that the actual actions executed by the robot are suited to the current interaction. This approach is especially useful in context where the cost of having the robot making errors is high, such as when interacting with vulnerable population.

3. Scope of the study

Following on from our earlier research on using people to teach an action policy to a robot during interaction (Senft et al., 2015b), we seek to evaluate SPARC when combined with the widely used learning paradigm of Reinforcement Learning (RL) (Sutton and Barto, 1998). We compare this approach to an alternative method combining interactive machine learning and reinforcement learning: IRL (Thomaz and Breazeal, 2008). To this end we tested both learning methods in the environment initially used by Thomaz and Breazeal and described in Section 4.

3.1. Interactive Reinforcement Learning

IRL implements the principles presented in Thomaz and Breazeal (2008). In IRL the human teacher can provide positive or negative feedback on the last action executed by the robot. The robot combines this with environmental feedback into a reward which is used to update a Q-table: a table with a Q-values (the expected discounted reward) assigned to every state-action pair and used to select the next action. Three additions to the standard algorithm have been proposed and implemented by Thomaz and Breazeal and are used here as well: guidance, communication by the robot and an undo option.

The guidance emerged from the results of a pilot study where participants assigned rewards to objects to indicate that the robot should do something with these objects. With the guidance, teachers can direct the attention of the robot toward certain item in the environment to indicate the robot that it should interact with them.

The robot can communicate its uncertainty by directing its gaze toward different items in the environment with equally high probability of being used next. The aim of this communication of uncertainty is to provide transparency about the robot's internal state, for example indicating when a guidance should be provided.

Finally, after a negative reward, the robot tries to cancel the effect of the previous action (if possible), resulting in a undo behaviour. As shown in the original paper, these three additions improve the performance on the task.

3.2. SPARC

SPARC (Supervised Progressively Autonomous Robot Competencies) uses a single type of input similar to the guidance present in IRL. However with SPARC, it is used to control the actions of the robot. The robot communicates every of its intentions (i.e the action it plans to execute next) to its teacher. The teacher can either not intervene and let the robot execute the suggested action or he can step in and force the robot to execute an alternative action. This combination of suggestions and corrections gives the teacher full control over the actions executed by the robot. This also makes the rewards redundant: rather than requiring the human to explicitly provide rewards a positive reward can directly be assigned to each action executed by the robot as it has been either forced or passively approved by the teacher.

3.3. Differences of approaches

Unlike IRL, SPARC offers full control over the actions executed by the robot. SPARC changes the learning paradigm from learning from the environment's response to learning from the users preferences. We use an expert in the task domain to evaluate the appropriateness of actions before their execution and we use this evaluation and control provided to the expert not to rely on observing negative effect of an action to learn that this action should be avoided, but rather what the best action is for each state. Even in a non-deterministic environment such as HRI, some actions can be expected to have a negative consequence. The human teacher can stop the robot from ever executing these actions, preventing the robot from causing harm to itself or its social or physical environment.

Another noticeable difference is the way in which the robot communicates with the user: in IRL, the robot communicates its uncertainty about an action and with SPARC its intention of executing an action.

It should also be noted that the quantity of information provided by the user to the robot is similar for both IRL and SPARC: in SPARC the user can offer the whole action space as commands to the robot, but removes the need for explicit rewards. While in IRL, the teacher can guide the robot toward a subset of the action space but has to manually provide feedbacks to evaluate the robot's decisions.

3.4. Hypotheses

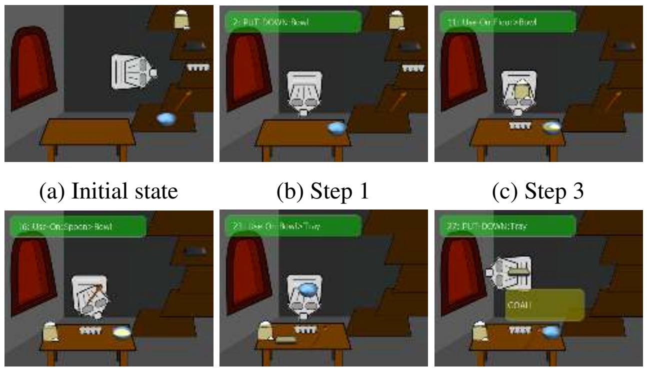
Three hypotheses are tested in this study:

- H1: *Effectiveness and efficiency with non-experts.* Compared to IRL, SPARC can lead to higher performance, whilst being faster, requiring fewer inputs and less mental effort from the teacher and minimising the number of errors during the teaching when used by non-experts.
- H2: *Safety with experts.* SPARC can be used by experts to teach an action policy safely, quickly and efficiently.
- H3: *Control.* Teachers prefer a method in which they can have more control over the robot's actions.

4. Methodology

4.1. Task

The task used in this study is the same as Thomaz and Breazeal (2008): Sophie's kitchen, a simulated environment on a computer where a virtual robot has to learn how to bake a cake in a kitchen. As the source code was not available, the task



(a) Initial state (b) Step 1 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

Fig. 1: Presentation of different steps in the environment. 1a initial state, 1b step 1: the bowl on the table, 1c step 3: both ingredients in the bowl, 1d step 4: ingredients mixed to obtain batter, 1e step 5: batter poured in the tray and 1f step 6 (success): tray with batter put in the oven. (Step 2: one ingredient in the bowl has been omitted for clarity)

was reimplemented to stay as close as possible to the description in the paper and the online version of the task[1].

The scenario is the following: a robot, Sophie, is in a kitchen with three different locations (shelf, table and oven) and five objects (flour, tray, eggs, spoon and bowl) as shown in Figure 1a. Sophie has to learn how to bake a cake and the user has to guide the robot through a sequence of steps while giving enough feedback so the robot can learn a correct series of actions. As presented in Figure 1, there are six crucial steps to achieve a successful result:

1. Put the bowl on the table.
2. Add one ingredient to the bowl (flour or eggs).
3. Add the second ingredient.
4. Mix the ingredients with the spoon to obtain batter.
5. Pour the batter in the tray.
6. Put the tray in the oven.

The environment is a deterministic Markov Decision Process, defined by a state, a set of actions (move left, move right, pick up, drop and use), a deterministic transition function, absorbing states (success or failure) after which the simulation is restarted in its initial state and an environmental reward function (+1 for success and -1 for failure and -0.04 for every other step to penalise long sequences). Different action policies can lead to success, but many actions end in a failure state, for example putting the spoon in the oven results in a failure. As argued by Thomaz and Breazeal, this environment provides a good setup to evaluate teaching methods to a robot due to the large number of possible states (more than 10,000), the presence of success and failure states and the sparse nature of the environmental reward function which increases the need for a teacher to aid the learning. More details on the environment are available in the original paper.

[1]`http://www.cc.gatech.edu/~athomaz/sophie/WebsiteDeployment/`

4.2. Implementation

In this experiment two systems are tested: IRL and SPARC. The underlying learning algorithm is strictly identical for both system, only the way of interacting with it is different: participants have more control in SPARC, implicitly reward action rather than explicitly and evaluate the intention of the action rather than its results. The learning algorithm (see algorithm 1) is a variation on Q-learning, without reward propagating[2]. This guarantees that any learning by the robot is only due to the teaching by the human, and as such provides a lower bound for the robot's performance. By using Q-learning, the performance of the robot would be higher.

4.2.1. Interactive Reinforcement Learning

We have implemented IRL following the principles presented in Thomaz and Breazeal (2008). The user can use the left click to display a slider in order to provide rewards. The guidance is implemented by right-clicking on objects: it directs the robot's attention to the object if facing it (a click on objects in different locations has no effect). Following the guidance, the robot will execute the candidate action involving the object. The action space is not entirely covered by this guidance mechanism: for example, it does not cover moving from a location to another. This guidance if used correctly, limits the exploration for the current step to the part of the environment evaluated as more interesting by the user without preventing the robot to explore in further steps. The robot can communicate its uncertainty by looking at multiple objects having similarly high probability of being used.

Some modifications were required to the original study due to the lack of implementation details in the original paper, one of them being the use of a purely greedy action selection instead of using softmax, due to the absence of parameters descriptions. The reliance on human rewards and guidance limits the importance of autonomous exploration, and thus, the greediness of the algorithm should assist the learning by preventing the robot to explore outside of the guided policy. Additionally, as the environment is deterministic and the algorithm is greedy, the concept of convergence is altered: once a trajectory has Q-Values high enough on all state-action pairs, it will be reinforced automatically.

4.2.2. SPARC

SPARC uses the gaze of the robot toward objects or locations to indicate which action the robot is suggesting to the teacher. Similarly to the guidance in IRL, the teacher can use the right click of the mouse on objects to have the robot execute the action associated to this object in the current state and this has been extended to also cover locations. With SPARC, the command covers all the action space: at every time step, the teacher can specify, if desired, the next action executed by the robot. If an action is not corrected, a positive reward of 0.25 is automatically received (as it has the implicit approval from the teacher) and

[2]In Q-learning the update function is $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma(\underset{a}{max}Q(s_{t+1}, a)) - Q(s_t, a_t))$

```
while learning do
    a = action with the highest $Q[s, a]$ value
    look at object or location used with $a$
    while waiting for correction (2 seconds) do
        if received command then
            $a$ = received command
            reward, $r = 0.5$
        else
            reward, $r = 0.25$
        end
    end
    execute $a$, and transition to $s'$
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma(\max_a Q(s_t, a)) - Q(s_t, a_t))$
end
```

Algorithm 1: Algorithm used in SPARC.

if the teacher selects another action, a reward of 0.5 is given to the correcting action (the corrected action is not rewarded). That way, actions actively selected are more reinforced and participants can still have give higher rewards when using IRL. This system allows for the use of reinforcement learning with implicit reward assignation, which simplifies the Human-Robot Interaction.

4.3. Experimental design

Participants are divided into 2 groups and interact first either with IRL or SPARC as shown in Figure 2. Before interacting, participants receive a information sheet explaining the task (describing the environment and how to bake a cake) and one explaining the system they are interacting with. Then they interact for three sessions with the assigned system. Each session is composed of a training phase and a testing phase. The training phase is composed of as many teaching episodes as the participant desires, a teaching episode ends when a success or failure state has been reached which returns the environment to the initial state. In the same way as in the initial experiment by Thomaz and Breazeal, participants can decide to terminate the training phase whenever they desire by clicking on a button labelled 'Sophie is ready', however it is also terminated after 25 minutes to impose an upper time limit to the study. After the end of a training phase, the robot will run a testing phase where the participant's inputs are disabled and which stops as soon as a ending state is reached or the participants decide to stop it (for example if the robot is stuck in a loop). This testing phase is used to evaluate the performance of the participants for this session. The interaction with a system consists of three repeated independent sessions with their own independent training and testing phases to observe how the interactions evolve as participants are getting used to the system.

After participants completed their three sessions with the first system, they are asked to interact for three more sessions with the other system. This way, every participant interacts three times with each system (IRL and SPARC) and the order of interaction is balanced. Additionally, a demographic questionnaire is given before the first interaction, a first post-interaction questionnaire after the interaction with the first system, a second identical one after the interaction with the second system and a final post-experiment questionnaire at the end of the experiment. All information sheets and questionnaires can be found online [3].

This experimental design prevents the risk of having an ordering effect by having a symmetry between conditions. Both conditions having a identical experimental procedure only with the order of interaction varying.

4.4. Participants

A total of 40 participants have been recruited using a tool provided by the university to reach a mixed population of students and non-student members of the local community. All participants gave written informed consent, and were told of the option to withdraw at any point. All participants received remuneration at the standard U.K. living wage rate, pro rata. Participants were distributed randomly between the groups whilst balancing gender and age (age M=25.6, SD=10.09; 24F/16M). Participants were mostly not knowledgeable in machine learning and robotics (average familiarity with machine learning M=1.8, SD=1.14; familiarity with social robots M=1.45, SD=0.75 - Likert scale ranging from 1 to 5).

In addition to naive non-expert users, an expert user (one of the authors) interacted five times with each system following a strictly optimal strategy in both cases. These results from the expert are used to evaluate hypothesis 2 and show the optimal characteristics of each system (IRL and SPARC) when used by trained experts such as therapist in a context of assistive robotics.

4.5. Metrics

4.5.1. Objective Metrics

We collected three metrics during the training phase: the number of times a participant reached a failure state while teaching, which can be related to the risks taken during the training and the teaching time (from 0 to 25 minutes) and the number of inputs provided during the training, which can be seen as the efforts invested in the teaching. The testing phase being only a single run of the taught action policy ending as soon as the robot reaches an ending state (failure or success) or if stopped by the participants. We only use the performance achieved during this single test as evaluation of the success of training. As not all participants reached a success during the testing phase, we used the six key steps defined in Section 4.1 as a way to evaluate the performance ranging from 0 (no step has been completed) to 6 (the task was successfully completed) during this testing run: for example a testing where the robot puts both ingredients in the bowl but reaches a failure state before mixing them would have a performance of 3.

4.5.2. Subjective Metrics

The post-interaction and post-experiment questionnaires provide additional subjective information to compare with the objective results from the interaction logs. Two principal metrics are gathered: the workload on participants and the perception of the robot.

[3] `http://www.tech.plym.ac.uk/SoCCE/CRNS/staff/esenft/experiment2.html`

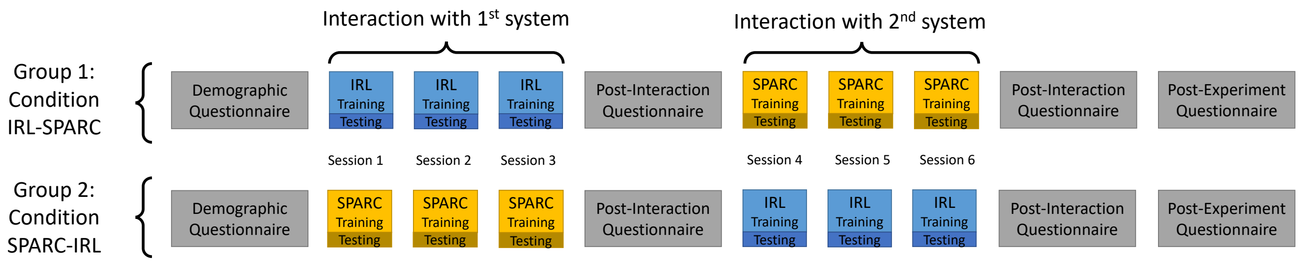

Interaction with 1st system
Interaction with 2nd system
Group 1: Condition IRL-SPARC
Group 2: Condition SPARC-IRL
Demographic Questionnaire
IRL Training Testing
SPARC Training Testing
Post-Interaction Questionnaire
Post-Experiment Questionnaire
Session 1
Session 2
Session 3
Session 4
Session 5
Session 6

Fig. 2: Participants are divided into two groups. They first complete a demographic questionnaire, then interact for three independent sessions (with a training and a testing phase each) with a system (IRL or SPARC). After a first post-interaction questionnaire, participants interact for another three sessions with the other system before completing the second post-interaction questionnaire and a final post-experiment questionnaire.

Workload is an important factor when teaching robots. As roboticists, our task is to make the teaching of the robot as undemanding as possible, meaning that the workload for user should be minimal. Multiple definitions for workload exist and various measures can be found in the literature. Due to its widespread use in human factors research and clear definition and evaluation criteria, we decided to use the NASA-Task Load Index (TLX) (Hart and Staveland, 1988). We averaged the values from the 6 scales (mental, physical and temporal demand, performance, effort and frustration) to obtain a single workload value per participant for each interaction. So we have two measures for each participant, after interaction with the first system (IRL or SPARC) and after the interaction using the other system.

Finally, the perception of the robot has been evaluated in the post-interaction and post-experiment questionnaires using subjective questions (measured on a Likert scale), binary questions (which robot did you prefer interacting with) and open questions on preference and naturalness of the interaction.

5. Results

Most of the results are non-normally distributed. Both ceiling and floor effects can be observed depending on the conditions and the metrics. For the teaching time, some participants preferred to interact much longer than others, resulting in skewed data. Likewise for the performance: often participants either reached a successful end state or did not hit any of the subgoals of the task ending often in two clusters of participants: one at a performance of 6 and one at 0. Similarly, some participants who interacted a long time with the system did not complete any step, while others could achieve good results in a limited time. Due to the data being not normally distributed, non-parametric statistical tests have been used. We use a combination of Friedman test for one way comparison with repeated measures, Wilcoxon rank sum test for between subject comparisons and the Wilcoxon signed rank test for within subject pairwise comparisons. Additionally, as each interaction consists of three sessions, a Bonferroni correction has been applied to pairwise comparison between sessions. A similar correction was used when comparing between systems to account of the two different groups. To apply the Bonferroni correction, we multiply the p-values by the correcting factors, which allows us to keep a global significance level at $p = .05$.

Initial results of the first interaction of the participants have been reported in Senft et al. (2016).

5.1. *Effectiveness and Efficiency with non-experts*

Four objective metrics (performance, teaching time, number of inputs used and number of failures) and one subjective metric (workload) have been used to evaluate the efficiency of IRL and SPARC.

5.1.1. *Performance*

Figure 3 presents the performance of participants during the interaction. In the first three sessions participants interacted with either IRL or SPARC, and swapped for the remaining three sessions. There is a significant difference of performance between systems; a Friedman test shows a significant difference between systems during the first three sessions ($\chi^2 = 50.8$, $p < .001$) and during the next three sessions ($\chi^2 = 36$, $p < .001$). Similarly, a significant difference in performance is noted within participants (Group 1: $\chi^2 = 37.9$, $p < .001$ - Group 2: $\chi^2 = 55.3$, $p < .001$). So in all the cases, participants interacting with SPARC achieved a significantly higher performance than those interacting with IRL, regardless of the order in which they interacted ($p < .05$ for all pairwise comparison). No difference of performance has been observed when using Wilcoxon signed rank test on the three repetitions between participants when interacting with the same system, so interacting for a second or third session with the same system does not have a significant impact on participants' performance.

It must be noted that in our study, only a limited number of participants managed to teach the robot to complete the task using IRL, this observation will be discussed in more details in Section 6.

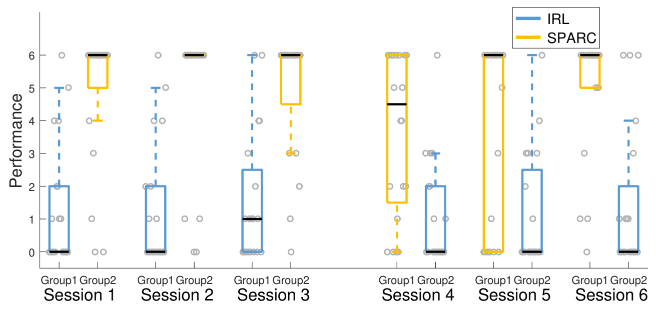


Fig. 3: Comparison of the performance for the six sessions (three with each system, IRL and SPARC, with interaction order balanced between groups). A 6 in performance shows that the taught policy leads to a success. The circles represent all the data points (n=20 participants per group), the black horizontal line the median and the top and bottom of the boxes the first and third quartiles. The learning is consistently better when using SPARC.

5.1.2. *Teaching Time*

The teaching times for all the interactions are shown in Figure 4. Regardless of the order in which they used SPARC or IRL, participants needed significantly less time to teach the robot when using SPARC than with IRL (Friedman test between participants for the first three sessions: $\chi^2 = 9.77$, $p = .0018$ - next three sessions: $\chi^2 = 20.2$, $p < .001$). Pairwise comparison also show significance ($p < .05$) except for sessions 3 and 5 which can be explained by the floor effect observed when teaching with SPARC and a potential loss of motivation when using IRL.

Additionally, when interacting multiple times with the same system, participants interacted significantly less in the second interaction with a system than during the first one (cf. Table 1) and only for SPARC the teaching time significantly decreases again between the second and the third session.

Table 1: Medians of the teaching time. In the first three sessions, group 1 interacted with IRL and group 2 with SPARC and participants interacted with the other system for the next three sessions.

	$\widetilde{X}_1$	$\widetilde{X}_2$	$\widetilde{X}_3$	$\widetilde{X}_4$	$\widetilde{X}_5$	$\widetilde{X}_6$
Group 1	16.3	7.44	6.17	3.97	2.45	1.53
Group 2	8.97	3.57	2.49	9.36	5.18	3.01

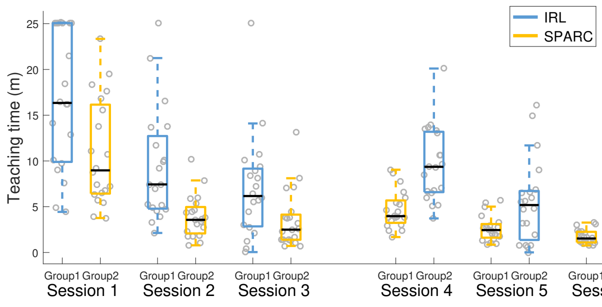


Fig. 4: Comparison of the teaching time (in minutes) for all the interactions. Participants spent less time teaching the robot when using SPARC than IRL.

5.1.3. *Number of Inputs*

The number of inputs used in both system is presented in Figure 5. For IRL, this represents every time a participant provided guidance or a reward to the robot, and for SPARC every time a participant provided a command. The number of inputs used is lower when teaching with SPARC than with IRL (Friedman test between participants for the first three sessions: $\chi^2 = 11.7$, $p < .001$ - next three sessions: $\chi^2 = 11$, $p < .001$). However with pairwise comparisons only session 2 ($p = .008$) and session 4 ($p < .001$) present a significantly different number of inputs used.

5.1.4. *Number of failures*

Figure 6 shows the number of failures observed with both systems for every session. In all the interactions, participants interacting with SPARC faced fewer failures during the training of the robot than those interacting with IRL (Friedman test between participants for the first three sessions: $\chi^2 = 47.8$, $p < .001$- next three sessions: $\chi^2 = 41.8$, $p < .001$ - within

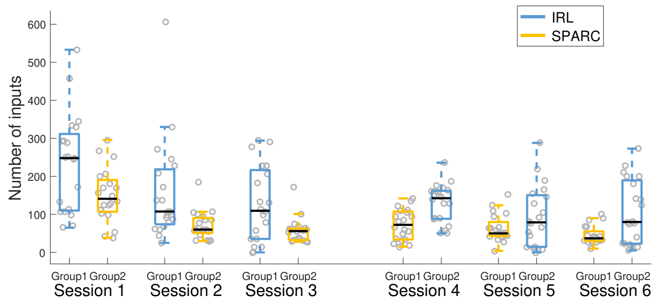


Fig. 5: Comparison of the number of inputs used during the teaching phases.

participants in group 1: $\chi^2 = 56.6$, $p < .001$ - group 2: $\chi^2 = 20.7$, $p < .001$ - all pairwise comparison: $p < .002$).

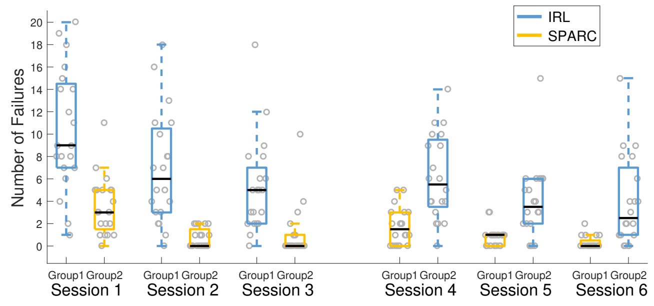


Fig. 6: Comparison of the number of failure states reached during the teaching process. Due to the ability to stop the robot from executing a suggested action, there are fewer failure states when using SPARC.

5.1.5. *Workload*

The average workload felt by participants after each interaction with a system is shown in Figure 7. As the workload data is normally distributed, a student t-test has been used. Participants interacting with IRL first reported an average workload of 12.9 (SD=2.33), with SPARC first this was 8.95 (SD=3.02). With SPARC after having interacted with IRL the reported workload was 7.44 (SD=3.33) and with IRL after SPARC it was 13.9 (SD=2.85). We found a significant difference between the reported workload when interacting with IRL or SPARC regardless of the order of interaction. This was also observed between participants (interaction with system 1, independent t-test: $t(38) = 4.63$, $p < .001$ - system 2, independent t-test: $t(38) = -6.5$, $p < .001$ - Group 1, paired t-test: $t(19) = 9.82$, $p < .001$ - Group 2, paired t-test: $t(19) = -6.8$, $p < .001$). Regardless of the interaction order, participants rated SPARC as having a lower workload than IRL.

5.1.6. *Validation of the hypothesis*

The objective data (performance, teaching time, number of inputs and number of failures) show that despite spending a shorter time interacting with SPARC and using less inputs, participants reached a higher performance than with IRL whilst facing fewer failures during the teaching. Additionally, when interacting with SPARC, participants' time required to teach the robot decreased with successive sessions, without affecting the performance. This indicates that after the first session, participants understood the interaction mechanism behind SPARC

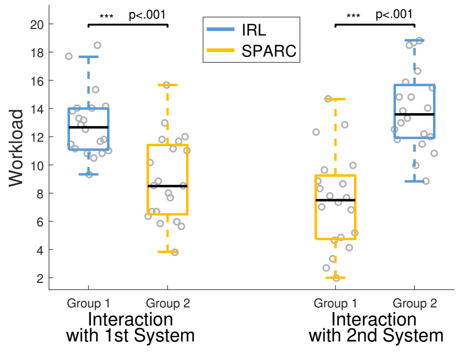


Fig. 7: Comparison of the workload experienced by participants. SPARC was perceived as having a lower workload. Results being normal, student t-test has been used for the comparisons.

and consistently managed to achieve a high performance whilst requiring less time to teach the robot the task. On the other hand, when interacting with IRL, participants' performance remains low over the session, and their teaching time decreases between session 1 and 2 but not between session 2 and 3. This might be due to a loss of motivation after session 1 where often participants did not succeed to teach the robot, reducing the desire to further interact in successive sessions.

The results suggest that teaching the robot using SPARC allows the robot to achieve a higher performance than with IRL, in a shorter time, without requiring more inputs, while making fewer errors when teaching. These objective results are also supported by subjective measures: the workload on the teacher is lower when using SPARC than when using IRL. For these reasons, H1 ('Compared to IRL, SPARC can lead to higher performance, whilst being faster, requiring fewer inputs and less mental effort from the teacher and minimising the number of errors during the teaching when used by non-experts.') is supported.

5.2. *Safety with experts*

To evaluate the safety offered by SPARC and IRL, an expert (one of the authors) interacted five times with each systems. In both cases, the expert followed a strictly optimal strategy. This shows the expected behaviours in optimal conditions, the best metrics achievable. Results of the interactions are presented in Table 2. In both cases, the expert successfully taught the robot (as indicated by a performance of 6), which indicates that both systems can be used to teach a robot an action policy. However the time required to teach the robot with IRL is significantly higher than with SPARC.

Additionally, when using IRL, even an expert cannot prevent the robot from reaching failure states during the training due to the lack of control over the robot's action. This is prevented when interacting with SPARC, due to the full control and clear communication, the teacher can ensure that only desired actions are executed. So with sufficient knowledge, an expert can teach the robot to behave safely without having to explore undesired states. This has real world applications, as random exploration is often impossible or undesirable, SPARC offers a way for the teacher to stop the robot from executing actions with negative consequences.

Similar results have been observed with the non-expert participants: in their last interaction with SPARC, both groups had a median of 0 failures for a performance of 6, meaning that more than half of the participants taught the robot the task without ever hitting a failure state. These results support H2 ('SPARC can be used by experts to teach an action policy safely, quickly and efficiently').

Table 2: Results of an expert interacting 5 times with each system following an optimal strategy. Both IRL and SPARC reached a success during all the testing phase, but the time required to teach SPARC was significantly shorter, and unlike IRL, not a single failure was reached during the training with SPARC. Data following a normal distribution, student t-test has been used.

	IRL *M(SD)*	SPARC *M(SD)*	$t(8)$	p
Perf.	6 (0)	6 (0)	NA	NA
Time (mn)	4.5 (0.67)	0.60 (0.03)	13.1	$< .001$
# of Fail.	3.2 (0.84)	0 (0)	8.55	$< .001$

5.3. *Control*

One of the main differences between the two methods is the way in which the concept of teaching is approached. With IRL an exploratory individual learning approach is followed: the robot has freedom to explore, and it can receive feedback on its actions and hints about actions to pursue next from a teacher. This is to some extent inspired by how children are taught, where the learning process can be more important than the achieved results. This is supported by the behaviours observed by Thomaz and Breazeal: their participants gave motivational rewards to the robot, just as one would to do to keep children motivated during learning, despite the absence of effect or use in classical reinforcement learning.

The post-experiment questionnaire included the open question: 'which robot did you prefer interacting with and why?'. Almost all the participants (38 out of 40) replied that they preferred interacting with SPARC. Half of all the participants used vocabulary related to the control over the robot actions ('control', 'instruction', 'command', 'what to do' or 'what I want') to justify their preferences without these words being used in the question. Furthermore, multiple participants reported being frustrated to have only partial control over the robot's actions with IRL, they would have preferred being able to control each action of the robot.

To the question 'which interaction was more natural?', 10 participants rated IRL as being more natural, using justifications such as: 'The robots thinks for itself', 'Some confusion in the [IRL] robot was obvious making it more natural', 'More like real learning', 'Because it was hard to control the robot' or 'People learn from their mistakes faster'. But despite acknowledging that IRL is more natural, closer to human teaching, participants still preferred teaching using SPARC. This suggests that when humans teach robots, they are focused on the results of the teaching: can the robot do the new task requested. This relates to the role of robots, they often interact in human-centred scenario where they have to complete a task for their users. And due to the absence of life-long learning for robots today, it is not worth investing time and energy to allow the robot to improve its learning process or explore on its own. These comments from the participants show support for H3 ('Teachers prefer a method providing more control over the robot's actions.').

6. Discussion

Despite not being originally designed to be used in combination with Reinforcement Learning, SPARC does achieve good results. This shows that principles covered by SPARC (control over the robot's actions, communication and evaluation of intentions and automatic execution of proposed actions) are agnostic to the learning algorithm and promote efficient teaching. Furthermore, SPARC achieves a higher performance, in a shorter time and facing less failures than IRL, whilst requiring a lower workload from the human teacher (supporting H1). Finally, when used by experts, SPARC demonstrates that teaching can be safe and quick: the full control over robot's action in the teacher's hands ensures that only desired actions will be executed (validating H2). These results show an interesting feature of teaching; as robots mainly interact in task oriented, human-centred environments, human teachers seem to prefer direct approaches focused on commands rather than letting the robot explore on its own (partial support for H3).

6.1. Comparison with original Interactive Reinforcement Learning study

Unlike in the original experiments evaluating IRL (Thomaz and Breazeal, 2008), in the study presented in this paper most of the participants did not succeed in teaching the robot the full cake baking sequence using feedback and guidance. In the Thomaz and Breazeal (2008) study, the participants were knowledgeable in machine learning (M=3.7, SD=2.3 - range: 1 to 7), but the population in the current study was drawn from a more general public having little to no knowledge of machine learning (M=1.8, SD=1.13 - range: 1 to 5). This can explain why a much larger number of participants did not achieve success with IRL in this study whereas Thomaz and Breazeal only reported 1 participant out of 13 failing the task. In our study, 12.5% of the participants and the expert did manage to train the robot using IRL. This seems to be largely due to participants not consistently rewarding correct actions, preventing the reinforcement learning algorithm from learning. This is why implicit rewards –every action allowed by the teacher is positively rewarded– tend to work better than explicit ones. This is consistent with Kaochar et al. (2011) who note that feedback is not well suited for teaching an action policy from scratch, but better for fine tuning. For teaching the basis of the action policy, they recommend using demonstrations, the method used by SPARC.

6.2. Advantages and limitations of SPARC

In the SPARC implementation for this study, SPARC reproduces actions selected by the teacher. So one can argue that no learning algorithm is required, instead the actions could just be blindly reproduced by the robot. However SPARC combined with reinforcement learning does provide advantages: due to the Q-Table, all the loops in the demonstration are removed when the robot interacts on its own and it provides a way to deal with variations in teaching. It also allows the robot to continue from any state in the trajectory. And finally, due to the suggestion/correction mechanism, the teacher can leave the robot to act on its own as long as it attempts correct actions, and the human to intervene only when the robot is about to execute an incorrect action.

Over the 79 successful trials using SPARC, participants used 47 different strategies to teach the robot the task of baking a cake. This shows how SPARC, as a single control mechanism, allows for different action policies to be learnt depending on the person teaching the robot. With SPARC the robot can adapt its behaviour to the human it is interacting with, profiling the user to find the desired way of behaving.

However SPARC also has limitations in the current implementation, related to the quality of the human supervised guidance. If the teacher allows an action to be executed by mistake (through inattention or by not responding in time), this action will be reinforced and will have to be corrected later on. This might lead to loops when successive actions are cancelling each other (such as move left, then right). In that case, the teacher has to step in and manually guide the robot to break this cycle. Furthermore, due to the automatic execution of actions, the teacher has to be attentive at all times and ready to step in when a wrong action is suggested by the robot.

In this version, SPARC has been applied to a scenario where a clear strategy with optimal actions is present. The interaction also takes place in a virtual environment with a discrete time. Real HRI are stochastic, happen in real time and often there is no clear strategy known in advance. However, we argue that human experts in the application domain can know what type of actions should be executed when, and which features of the environment they used for their decision. As this knowledge can not be available to the robot's designers, robots should be able to learn from a domain user in an interactive fashion. In the current implementation, SPARC mainly receives inputs from a teacher at predefined discrete times and still does not use the human knowledge to it's fullest: the learning algorithm is still simple and with limited inputs, but as described in Section 6.4, we are working on improving SPARC to suit real-world HRI.

Nevertheless, we argue that SPARC allows for easy and safe teaching due to the presence and control by the teacher. And the suggestion/correction mechanism with automatic execution of actions allows for a smooth teaching process where the workload on the teacher can decrease over time as shown in Senft et al. (2015b). The workload of the teacher when starting is relatively high, when the robot has no information on which actions to take yet, and decreases over time requiring only limited intervention by the teacher.

6.3. Recommendations for designing interactive machine learning for human-robot interactions

From observing the participants interacting with both systems, we derived four recommendations for future designs of interactive learning robot. Although the study here used a simulated robot, we believe these to be also relevant for real-world, physical installations.

6.3.1. *Clarity of the interface*

Algorithms used in machine learning often need precisely specified inputs and outputs and require an internal representation of the world and policies. These variables are often not accessible to a non expert: the weights of a neural network or the values in a Q-table are not easily interpreted, if at all. The inner workings of the machine learning algorithms are opaque, and people only have access to input and output of the black box that is machine learning. As such, care needs to go into making the input and output intuitive and readable. For example, in this study (following Thomaz and Breazeal's original study), the communication between the robot and the teacher occurred through the environment: using clicks on objects rather than buttons on a graphical user interface. This design decision has important consequences as participants first have to familiarise themselves with the interface: how to interpret the robot's behaviour, what actions are available for each state and what is the exact impact of the actions? This lack of clarity leads to a high number of failures and high teaching time during the first session in our study. So we argue that to avoid this precarious discovery phase for the teachers, roboticists have to design interfaces taking into account results from the Human Factors community as advocated by Adams (2002).

6.3.2. *Limits of human adaptability*

Human-Robot Interaction today is facilitated by relying on people adapting to the interaction, often making use of anthropomorphisation (Złotowski et al., 2015). Roboticists use people's imagination and creativity to fill the gaps in the robot's behaviour. However, human adaptivity has its limits: in our study, often participants adopted one particular way of interacting with the system and they hold on to it for a large part of the interaction. For example, participants clicked on an object requiring two actions to interact with, assuming that the robot had planning capabilities which it did not. Or when the robot was blocked in some cycles (due to constant negative reward in IRL or due to a loop created and not stopped with SPARC), participants kept on trying the same action to break the loop, without really exploring alternatives. For these reasons, if robots are to be used with a naive operator, they need a mechanism to detect these 'incorrect' uses and either adapt to these suboptimal human inputs or they need to inform the user that this type of input is not supported and clarify what human behaviour is appropriate instead.

6.3.3. *Importance of keeping the human in the learning loop*

Other methods have been used to provide a robot with an action policy, for example Liu et al. (2016) argue that instead of having a human teach the robot, interactive behaviours can be extracted from observing human experts interacting and by using big data machine learning techniques on these observations. This approach has shown some promise (Liu et al., 2014), but we argue that an action policy for human-robot interaction should be able to be modified online by a human. Furthermore, the presence of a human in the loop can allow the machine learning to deal with sensor errors or imperfect action policies. An expert supervising the robot should also be able to prevent the execution of specific actions or force the execution of others. This was one of the important points we considered when proposing SPARC: there is no distinction between a teaching and a testing phase, they are merged into a single phase. The teacher can correct the robot when needed and let it act when it behaves correctly. Participants used this feature of SPARC in this study: many participants corrected SPARC only when required rather than forcing every action, 37.5% of the participants even let the robot complete the task without giving a single command before starting the test to be sure that the robot is ready. So SPARC has been used as a tool to provide online learning to a robot whilst keeping the teacher in control, but reducing the need of intervention over time.

6.3.4. *Keeping people in control*

Most of the scenario where a robot has to learn how to interact with humans are human-centred: the robot has to complete a task to help a human (such as in socially assistive robotics). In these scenarios, the goal of the learning is to ensure that the robot can complete the task assigned to it, not to provide the robot with tools to learn more efficiently in further interactions. Similarly, participants in our study did not desire to have the robot exploring on its own and learn from its experience, they wanted to be able to direct the robot. Furthermore, a lack of control over the robot's actions can lead to frustration and loss of motivation for the teacher. This human control is especially critical when the robot is designed to interact with other people as undesired actions can have a dramatic impact, such as causing harm for the interaction partners or bystanders. For these reasons, we argue that when designing an interactively learning robot for Human-Robot Interaction in human-centred scenario, it is critical to keep the human in control.

However, this control does not mean that the robot cannot learn and become autonomous. We take a stronger inspiration from Learning from Demonstration, using human input more efficiently to guide the learning, speeding it up and making it safer, especially in the early stages of the learning. The human is in control mainly when the robot is prone to make exploratory mistakes, and can prevent them before they occur, but once the action policy is appropriate enough, the teacher can leave the robot learn mostly on its own and refine its action policy with limited supervision from a human.

6.4. *Future work*

We are currently working on a new experiment in which people interacting with a robot in a continuous time and non-deterministic environment. In this experiment, the teacher is able to send commands to the robot, provide rewards and identify features in the environment they consider important. The learning algorithm will take these inputs into account and combine them with interaction metrics to learn. An approach could be to use the actor-critic paradigm: the critic being an objective evaluation of the action results (environmental rewards), and the actor using results from the critic and teacher's guidance to update the action policy.

7. Conclusion

SPARC has been proposed to address the problem of providing a robot with adaptive behaviour whilst guaranteeing that the behaviour expressed by the robot remains suitable for task at hand. To achieve this, a suggestion and correction system has been used to allow a teacher to be in control of the robot at all times whilst not having to manually select every single action. This approach has been combined with reinforcement learning and was compared to IRL, where the operator manually provides feedback and guidance to the learning agent. The results from a user study involving 40 participants show that SPARC can be used to let naive participants successfully teach an action policy. While doing so SPARC requires less teaching time and limits undesired actions during the teaching phase when compared to IRL. Additionally, the workload on users was lower when using SPARC. Based on these results and other observations, we propose four guidelines to design interactive learning robots: (1) the interface to control the robot has to be intuitive, (2) the limits of human adaptability have to be taken into account (robots should detect deadlocks in human behaviours and adapt their way to be controlled or inform the human about it), (3) the operator should be kept in the learning loop and (4) teachers should stay in control of the robot behaviour when interacting in sensitive environment. The first two points can be seen to apply to all robot teaching methods, and should be addressed at the time of designing the interface. By definition, SPARC aims to address these last two points: maintaining the performance of an adaptive system by remaining under progressively decreasing supervision.

Acknowledgements

This work was supported by the EU FP7 DREAM project (grant no. 611391) and EU H2020 Marie Sklodowska-Curie Actions project DoRoThy (grant 657227).

References

Adams, J.A., 2002. Critical considerations for human-robot interface development, in: Proceedings of 2002 AAAI Fall Symposium, pp. 1–8.

Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T., 2014. Power to the people: The role of humans in interactive machine learning. AI Magazine 35, 105–120.

Argall, B.D., Chernova, S., Veloso, M., Browning, B., 2009. A survey of robot learning from demonstration. Robotics and autonomous systems 57, 469–483.

Billard, A., Calinon, S., Dillmann, R., Schaal, S., 2008. Robot programming by demonstration, in: Springer handbook of robotics. Springer, pp. 1371–1394.

Cakmak, M., Thomaz, A.L., 2012. Designing robot learners that ask good questions, in: Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, ACM. pp. 17–24.

Chernova, S., Veloso, M., 2009. Interactive policy learning through confidence-based autonomy. Journal of Artificial Intelligence Research 34, 1.

Chyzhyk, D., Ayerdi, B., Maiora, J., 2013. Active learning with bootstrapped dendritic classifier applied to medical image segmentation. Pattern Recognition Letters 34, 1602–1608.

Fails, J.A., Olsen Jr, D.R., 2003. Interactive machine learning, in: Proceedings of the 8th international conference on Intelligent user interfaces, ACM. pp. 39–45.

Fong, T., Nourbakhsh, I., Dautenhahn, K., 2003. A survey of socially interactive robots. Robotics and autonomous systems 42, 143–166.

Gorostiza, J.F., Salichs, M.A., 2011. End-user programming of a social robot by dialog. Robotics and Autonomous Systems 59, 1102–1114.

Hart, S.G., Staveland, L.E., 1988. Development of nasa-tlx (task load index): Results of empirical and theoretical research. Advances in psychology 52, 139–183.

Hoffman, G., 2016. Openwoz: A runtime-configurable wizard-of-oz framework for human-robot interaction, in: 2016 AAAI Spring Symposium Series.

Holzinger, A., 2016. Interactive machine learning for health informatics: when do we need the human-in-the-loop? Brain Informatics , 119–131.

Kaochar, T., Peralta, R.T., Morrison, C.T., Fasel, I.R., Walsh, T.J., Cohen, P.R., 2011. Towards understanding how humans teach robots, in: International Conference on User Modeling, Adaptation, and Personalization, Springer. pp. 347–352.

Knox, W.B., Stone, P., 2010. Combining manual feedback with subsequent mdp reward signals for reinforcement learning, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems. pp. 5–12.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature .

Liu, P., Glas, D.F., Kanda, T., 2016. Learning interactive behavior for service robots the challenge of mixed-initiative interaction, in: Proceedings of the workshop on Behavior Adaptation, Interaction and Learning for Assistive Robotics.

Liu, P., Glas, D.F., Kanda, T., Ishiguro, H., Hagita, N., 2014. How to train your robot-teaching service robots to reproduce human social behavior, in: Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on, IEEE. pp. 961–968.

Olsen, D., 2009. Building interactive systems: principles for human-computer interaction. Cengage Learning.

Riek, L.D., 2012. Wizard of oz studies in hri: a systematic review and new reporting guidelines. Journal of Human-Robot Interaction 1.

Senft, E., Baxter, P., Belpaeme, T., 2015a. Human-guided learning of social action selection for robot-assisted therapy, in: 4th Workshop on Machine Learning for Interactive Systems.

Senft, E., Baxter, P., Kennedy, J., Belpaeme, T., 2015b. Sparc: Supervised progressively autonomous robot competencies, in: International Conference on Social Robotics, Springer. pp. 603–612.

Senft, E., Lemaignan, S., Baxter, P.E., Belpaeme, T., 2016. Sparc: an efficient way to combine reinforcement learning and supervised autonomy, in: FILM Workshop at NIPS'16.

Settles, B., 2010. Active learning literature survey. University of Wisconsin, Madison 52, 11.

Stumpf, S., Rajaram, V., Li, L., Burnett, M., Dietterich, T., Sullivan, E., Drummond, R., Herlocker, J., 2007. Toward harnessing user feedback for machine learning, in: Proceedings of the 12th international conference on Intelligent user interfaces.

Sutton, R.S., Barto, A.G., 1998. Reinforcement learning: An introduction. volume 1. MIT press Cambridge.

Taylor, M.E., Suay, H.B., Chernova, S., 2011. Integrating reinforcement learning with human demonstrations of varying ability, in: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2.

Thill, S., Pop, C.A., Belpaeme, T., Ziemke, T., Vanderborght, B., 2012. Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. Paladyn, Journal of Behavioral Robotics 3, 209–217.

Thomaz, A.L., Breazeal, C., 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. Artificial Intelligence 172, 716–737.

Złotowski, J., Proudfoot, D., Yogeeswaran, K., Bartneck, C., 2015. Anthropomorphism: opportunities and challenges in human–robot interaction. International Journal of Social Robotics 7, 347–360.

Leveraging Human Inputs in Interactive Machine Learning for Human Robot Interaction

Emmanuel Senft
Centre for Robotics and Neural Systems
Plymouth University, UK
emmanuel.senft@plymouth.ac.uk

Séverin Lemaignan
Centre for Robotics and Neural Systems
Plymouth University, UK
severin.lemaignan@plymouth.ac.uk

Paul E. Baxter
Lincoln Centre for Autonomous Systems
University of Lincoln, UK
pbaxter@lincoln.ac.uk

Tony Belpaeme
CRNS – Plymouth University, UK
imec - ID Labs – Ghent University, Belgium
tony.belpaeme@plymouth.ac.uk

ABSTRACT

A key challenge of HRI is allowing robots to be adaptable, especially as robots are expected to penetrate society at large and to interact in unexpected environments with non-technical users. One way of providing this adaptability is to use Interactive Machine Learning, i.e. having a human supervisor included in the learning process who can steer the action selection and the learning in the desired direction. We ran a study exploring how people use numeric rewards to evaluate a robot's behaviour and guide its learning. From the results we derive a number of challenges when designing learning robots: what kind of input should the human provide? How should the robot communicate its state or its intention? And how can the teaching process by made easier for human supervisors?

Keywords

Interactive Machine Learning; Autonomy; HRI

1. INTRODUCTION

One important challenges in HRI is to allow users, who often have no technical expertise, to personalise the behaviour of the robot they are using. It seems infeasible to expect either users to be satisfied by a robot with a static behaviour or for the robot's designers to be able to anticipate all the needs of the users and all the different environments a robot could interact in. For this reason, we argue that robot behaviour should be adaptive at run-time, and especially that non-experts in technology should be able to teach a robot new action policies.

Interactive Machine Learning (IML) is a field of research which aims to include end-users in the machine learning process [1, 3]. The idea is to move away from robots as complex black boxes with inaccessible input and output, to systems that can be intuitively (re)programmed by the users. One advantage of this approach is it empowers users with the ability to personalise their robot according to their needs and desires.

IML has principally been tested on virtual agents. A good example of IML is the TAMER framework [4] which predicts the reward a human would give and use this prediction to select a next action maximising the predicted reward.

HRI '17 Companion March 06-09, 2017, Vienna, Austria
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4885-0/17/03.
DOI: http://dx.doi.org/10.1145/3029798.3038385

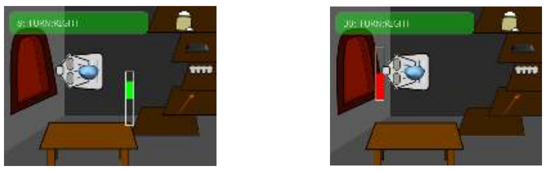


Figure 1: Examples of positive (left) and negative (right) reward in the robot cake baking task.

2. EXPERIMENT

Methodology.

IML has not often been applied to robotics. An application using virtual robots was presented in [6], which presented a study where we compared two different methods used to teach robots an action policy. The first, Interactive Reinforcement Learning (IRL), is derived from Reinforcement Learning (RL) [7], the difference being that user now provides rewards, rather than the environment (cf. Figure 1). In our implementation, the participant could evaluate the robot's actionsby moving a slider on a graphical interface, the value of the slider acted as the reinforcement learning reward. The second method, inspired by [5], uses a more direct control method in which the robot communicates its intentions to the participant who can either passively accept the suggestion or actively select an alternative action.

In the task, inspired by Thomaz and Breazeal [8], a virtual robot is in a kitchen and has to learn how to bake a cake. The users know what the robot should do to finish the cake, but multiple strategies can lead to success. The IRL algorithm is similar to that used by [8].

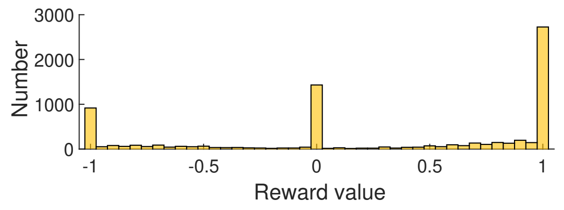


Figure 2: Distribution of rewards according to their numeric value.

Participants were divided into two groups of 20, each group interacting with a different system.

Results.

This paper reports how participants used the rewards in the IRL condition. Figure 2 presents the distribution of rewards given to the robot. In our case, the distribution was tri-modal, with only three types of rewards given to the robot. Over a total of 7364 rewards, we observe 899 with a value of -1, 1337 with a value of 0 and 2653 of a value of 1.

This indicates that even if participants had the opportunity to provide fine grained numerical rewards, they decided to evaluate the robot's actions either as *bad*, *neutral* or *good*. We identify three reasons potentially explaining this behaviour. The first one is related to the unambiguity of the expected behaviour: when a desired strategy is clear, humans might only use extremes to give feedback to the robot. Alternatively, the tri-modality of the reward distribution can be due to the interface used, a slider makes the use of extremes easier. And lastly, as participants were time constrained (they only had 2 seconds to evaluate the robot's action) they might not have taken the time to use a fine grained rewarding strategy.

3. DISCUSSION

From these results, we derive three challenges that robot designers will face when allowing humans to teach robots.

Type of inputs.

The first challenge is to make human input efficient and generalisable over different tasks. RL seems like a reasonable approach: the user can provide numerical rewards to evaluate the action executed by the robot. However, as the task becomes more complex, the algorithm converges only after a long series of trials and errors which is undesirable. Another limit of numerical rewards is that they generally are assigned *after* the execution of an action, and so do not allow the supervisor to prevent the robot from making an error, even if the supervisor could have known beforehand that this action was not appropriate. Reward-based learning is general, but it does not make good use of human domain knowledge and tutoring competency.

Other types of inputs could be used; in [6], we propose using commands rather than feedback. Commands allow the user to have more control of the robot, but limit the actions to a predefined set of actions. A way to generalise commands to a larger set would be to use natural language and ability to teach new actions associated to new commands. A robot could also combine different types of inputs from the human: both explicit (rewards or commands for example) and implicit (such as the reactions of other humans).

Clarity of robot's communication.

The robot should also provide the human with feedback about its internal state, including its intentions, uncertainty, learning progress and confidence. In [6], we argue that intention communication is especially important when robots are interacting in the real world, so as not to fluxom people or execute undesired actions. Furthermore, if the robot has planning abilities, the robot can also explain its actions and communicate what its next actions will be.

Similarly to the content of the robot's communication, the medium is important. Humans have evolved to use social signals, and robots should use these too. Speech could be a good way of communicating more complex states, intentions and plans, but it would be interesting to modulate the sentences expressed by the robot in a way which is socially acceptable and which does not annoy the long-term user, avoiding repetitions and bluntness often associated with robots.

Reduce the workload on human teachers.

A last important challenge is maintaining the user's comfort when teaching. As explained in [2], a robot using the same mode of communication without considering the human it is interacting with could annoy the user. In our study, many participants also reported frustration due to them often knowing what the robot should do, but not being able to have the robot execute the desired action. Robots learning from humans should reduce the workload on the teacher, and give the teacher enough control of the robot's actions while taking into account the human's state when learning.

Acknowledgments

This work was supported by the EU FP7 DREAM project (grant no. 611391) and EU H2020 Marie Sklodowska-Curie Actions project DoRoThy (grant 657227).

4. REFERENCES

[1] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[2] M. Cakmak and A. L. Thomaz. Designing robot learners that ask good questions. In *HRI'12*, pages 17–24, 2012.

[3] J. A. Fails and D. R. Olsen Jr. Interactive machine learning. In *IUI'03*, pages 39–45, 2003.

[4] W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *K-CAP'09*, pages 9–16, 2009.

[5] E. Senft, P. Baxter, J. Kennedy, and T. Belpaeme. SPARC: Supervised progressively autonomous robot competencies. In *ICSR'15*, pages 603–612, 2015.

[6] E. Senft, S. Lemaignan, P. E. Baxter, and T. Belpaeme. Sparc: an efficient way to combine reinforcement learning and supervised autonomy. In *FILM Workshop at NIPS'16*, 2016.

[7] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[8] A. L. Thomaz and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6):716–737, 2008.

Period 4

Robots in the classroom: Learning to be a Good Tutor

Emmanuel Senft[1,*], Séverin Lemaignan[1], Madeleine Bartlett[1], Paul Baxter[2], Tony Belpaeme[1,3]

1 CRNS, University of Plymouth, UK
2 L-CAS, University of Lincoln, UK
3 ID Lab – imec, University of Ghent, Belgium

*** emmanuel.senft@plymouth.ac.uk**

Abstract

To broaden the adoption and be more inclusive, robotic tutors need to tailor their behaviours to their audience. Traditional approaches, such as Bayesian Knowledge Tracing, try to adapt the content of lessons or the difficulty of tasks to the current estimated knowledge of the student. However, these variations only happen in a limited domain, predefined in advance, and are not able to tackle unexpected variation in a student's behaviours. We argue that robot adaptation needs to go beyond variations in preprogrammed behaviours and that robots should in effect learn online how to become better tutors. A study is currently being carried out to evaluate how human supervision can teach a robot to support child learning during an educational game using one implementation of this approach.

1 Introduction

Compared to lectures, tutoring has been showed to increase the learning gains of humans [4]. In particular, one-to-one tutoring enables a more inclusive teaching, by adapting the content of the lesson and the style of interaction to the needs and preferences of the student. As such, tutoring presents numerous opportunities for social robots in education: teaching language [3], how to write [9], maths or sciences [7].

To be as effective as human tutors, robots should not deliver a one-size-fits-all teaching content; they need to adapt their behaviour to the student they are teaching. Traditional methods of developing adaptable robot tutors have either used predefined behaviours that the robot can switch between or have adapted the difficulty of a class to meet the estimated knowledge of the user. But we are convinced that to thrive, robot tutors need to go beyond and learn how to behave efficiently within each situation. Furthermore, we also wish to empower the teachers who are ultimately leading the teaching and who know their students best. Robots should remain tools in the hands of the teachers, and teachers should have the freedom to shape the robot into their own personalised teaching assistant. To this end, we rely on the teacher to demonstrate to the robot the desired tutoring behaviour using a Wizard of Oz (WoZ) approach. As the robot is exposed to these demonstrations, it learns and starts producing its own suggestions of actions to support the students. Using human feedback and commands, the robot's action policy improves over time and when the teacher deems this behaviour to be adequate, the robot can take over the tutoring session, interacting autonomously (if desired) with the students and freeing the teacher to work with other students.

2 Related Work: Adapting Teaching Strategies in Robots for Learning

To increase the amount of learning children gain from the tutoring setup, robots can adapt their behaviour to suit the preferences and requirements of the student they are teaching. One solution, as used in [8], is to have different empathic strategies such as: encouraging comments, scaffolding, offering help or intentionally making errors. By modelling the child's preferences and reactions to these strategies the robot can select the most efficient one for each specific child. Other methods use Bayesian Network and Knowledge Tracing to estimate the learner's knowledge and provide advice on missing skills [10], or select a task and a difficulty level which will maximise the learning gain [6, 11]. Alternatively, if the task requires mainly practice of poor skills (such as handwriting), every aspect of the child's knowledge can be continuously monitored and training examples can be selected to encourage the practice of these poor skills [9].

One method which goes further than simple adaptation and allows the robot to tackle previously unseen or unanticipated child behaviours as a human tutor would, was introduced by Sequeira et al. in [15]. The authors proposed the restrictive-perception Wizard of Oz: the robot starts as non-autonomous; controlled by a human. Then an autonomous controller is developed from the human demonstrations and hand-coded rules before being deployed to interact autonomously and replicate the human demonstrations.

However, in [12] and [13], we argued that the learning of an action policy should occur online, with human supervision. This reduces the workload of the wizard, allowing them to monitor the robot's learning while ensuring that even in the learning phase, the robot's behaviour is efficient. While this method originated from the *Robots in Therapies* field, we are convinced that *Robots in Education* is an area which would greatly benefit from such an approach.

3 Progressive Autonomy for Robots in Education

3.1 A teacher-led learning process

Developed to reduce the workload on a robot's supervisor in a therapy scenario, the Supervised Progressive Autonomous Robot Competencies (SPARC) [12] uses online learning from demonstration combined with suggestions from the robot and potential corrections from the teacher to rapidly learn and improve a robot's action policy.

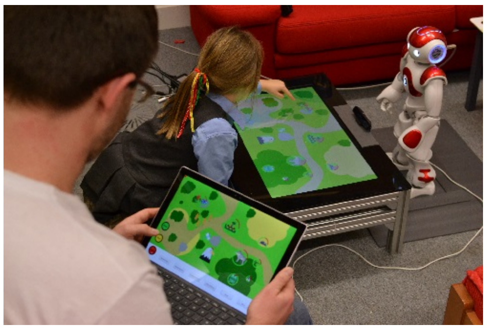


Figure 1. Interaction setup: the teacher (one of the authors) on the left uses a GUI on a tablet to control and teach the robot how to interact with the child until reaching a good action policy.

One advantage of such a technique is that it empowers the end-users, the teachers. They can control the robot's behaviour in a teaching phase, ensuring that the robot reacts properly to the different behaviours expressed by the child while monitoring the progress of the robot's learning (Figure. 1). As the robot learns a better action policy, the teacher can step back and focus more on the child's behaviour while letting the robot progressively take over the tutoring session, freeing the teacher to take care of other children. Keeping the human in the loop and in control of the robot's actions provides the algorithm access to efficient demonstrations and ensures that incorrect actions due to missing knowledge are corrected before being executed, which ensures quick and efficient learning [13]. Having been demonstrated to work only in simple or discrete (in space and in time) environments, this method has not yet been evaluated in a real-world, complex environment such as tutoring.

3.2 A high-dimensional example: a robot tutor to learn about food chains

SPARC for creating a teachable tutor has been implemented in a teaching scenario and is currently being tested with children (source code available [1] [2] [3]).

In this study, children are invited to learn about food chains in a gamified and open learning environment. The setup, as shown in Figure 1, uses the Sandtray paradigm [2] whereby a child is interacting with a robot through a large touchscreen sitting between them. The game presents movable animals and passive plants and the goal is to keep the animals alive as long as possible. Animals have energy that decreases as time goes by and the students have to make them interact with other animals or plants to feed them and replenish their energy. As the students learn how to feed animals to keep their energy high, by extension, they can learn what food each animal eats.

To support this learning, the robot can provide advice (move an animal to, toward or away from other animals or plants), verbal feedback (remind rules, provide congratulation and encouragement) or draw the child's attention to an animal.

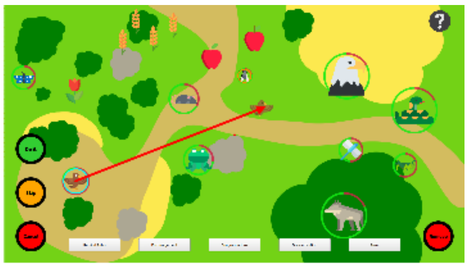


Figure 2. GUI used for supervising: the teacher moves the bird close to the fly and selected both of them as relevant feature for this action (blue and orange circles). Buttons at the bottom are used to have the robot provide feedback.

The teacher uses a tablet running a supervisor GUI replicating the state of the game as it is currently being played on the touchscreen. This GUI allows for remote control of the robot's actions (highlighting features to speed up the learning by providing relevant dimensions for the algorithm [14]) and receives suggestions from the robot about what action to do next (cf. Figure 2).

The robot has access to 655 discrete output actions and an abstracted representation of the state of the game and the interaction through a 210 dimensional vector of values bounded between 0 and 1 (distances between the elements, time since the child touched each elements, time since robot's actions or time since other interaction events). The system must therefore find a correct mapping between a 210 dimensional input vector to a 655 exclusive output one. Many algorithms can learn in such an environment, but traditional Reinforcement Learning algorithms would take a prohibitive amount of time, exhausting many children as the robot would at first be behaving randomly and providing incoherent messages. As such, a method like SPARC offers an opportunity to quickly learn a useful action policy despite the complexity of the environment.

To learn fast, the algorithm used is a variation of the Nearest Neighbours algorithm [5] where actions are defined on a sliced version of the general space [14]. This algorithm allows fast, lightweight and online learning with transparency as the algorithm can highlight which features of the space have been used to make the suggestion.

At the start of the first interaction, the database the algorithm has access to is a blank sheet without any actions, and as the supervisor selects actions, the database of demonstrations is filled, associating actions with the value of the state on a subset of the dimensions. As the database becomes richer, the robot suggests a larger number of correct actions, reducing the workload on the teacher until reaching a point where the teacher only has to correct/select a low number of actions to fine-tune the robot's policy.

This setup is currently being tested in primary schools in the UK with children in

[1] `https://github.com/emmanuel-senft/freeplay-sandbox-ros-sparc/tree/task`
[2] `https://github.com/emmanuel-senft/freeplay-sandbox-qt/tree/food-chain`
[3] `https://github.com/emmanuel-senft/freeplay-sandbox-qt-supervisor`

Years 4 and 5 (8-10 years old) with one of the authors (a PhD student in Psychology naive to the algorithm) acting as a teacher.

4 Discussion

4.1 Future work

The current implementation has several limitations that should be tackled in future work. Firstly, for now, the algorithm can only take demonstrations (and negative feedback) as input. It would be interesting to start with a set of rules defining a baseline of behaviour, which could then be refined online by adding either new rules or demonstrations. Additionally, currently the algorithm only reproduces a demonstrated action policy and does not have the opportunity to learn from its interaction with the world. Future work could focus on designing a system which adds the prediction and use of rewards in reaction to environmental events (such as with Inverse Reinforcement Learning [1]) and techniques to model a child's knowledge to potentially learn an action policy more efficient than the demonstrated one.

While allowing the robot to learn faster using initial knowledge from a human, including a supervisor in the action selection loop also limits the time-scale of the interaction. Allowing the human enough time to correct a suggested action requires the addition of a few seconds between the suggestion of an action and its auto-execution, which implies that the rate of action selection has to be below 1 Hz. This delay can reduce the optimality of an action between its suggestion and execution, slowing down the learning process. Future work could explore teaching at different levels of abstraction, giving the teacher time to override only high level actions where exact timing is less critical.

4.2 Opportunities

The goal of the approach is to provide teachers with a way to create their own personalised robotic tutors, which can be controlled by the teacher and taught how to interact with children according to the teacher's personal preferences. The robot learns from the first demonstration, and to obtain a correct autonomous action policy the teacher would need to spend enough time to cover the required actions in the domain of application. The time dedicated to teach the robot varies with the complexity of the policies from a few minutes for simple ones to more than one hour for complex ones. However, it needs to be pointed out that while the teacher is teaching the robot how to interact, s/he does also actively support students in their learning in a different, while similar, way than traditional human-to-human tutoring.

The mixture between WoZ, learning and autonomy additionally allows the teacher to take a more active supervisory stance for children with more difficulties to offer them an experience tailored to their specific needs, or to select a special (previously taught) action policy for the robot. If the study is successful, we would have demonstrated a way to teach a robot online, an efficient action policy to interact with humans in a complex (high dimensional), indeterministic (children are highly stochastic) environment. This or similar methods could be applied to other domains ranging from personal robotic assistants at home to collaborative manufacturing.

Acknowledgements

This work was supported by the EU FP7 DREAM project (grant no. 611391) and the EU H2020 Marie Sklodowska-Curie Actions project DoRoThy (grant 657227).

References

1. P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

2. P. Baxter, R. Wood, and T. Belpaeme. A touchscreen-based ‘Sandtray’to facilitate, mediate and contextualise human-robot social interaction. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 105–106. IEEE, 2012.

3. T. Belpaeme, P. Vogt, R. van den Berghe, K. Bergmann, T. Göksun, M. de Haas, J. Kanero, J. Kennedy, A. C. Küntay, O. Oudgenoeg-Paz, et al. Guidelines for designing social robots as second language tutors. *International Journal of Social Robotics*, 2017.

4. B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4–16, 1984.

5. T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

6. G. Gordon and C. Breazeal. Bayesian Active Learning-Based Robot Tutor for Children’s Word-Reading Skills. In *AAAI*, pages 1343–1349, 2015.

7. J. Kennedy, P. Baxter, and T. Belpaeme. The robot who tried too hard: Social behaviour of a robot tutor can negatively affect child learning. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 67–74. ACM, 2015.

8. I. Leite, G. Castellano, A. Pereira, C. Martinho, and A. Paiva. Modelling empathic behaviour in a robotic game companion for children: an ethnographic study in real-world settings. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 367–374. ACM, 2012.

9. S. Lemaignan, A. Jacq, D. Hood, F. Garcia, A. Paiva, and P. Dillenbourg. Learning by teaching a robot: The case of handwriting. *IEEE Robotics & Automation Magazine*, 23(2):56–66, 2016.

10. D. Leyzberg, S. Spaulding, and B. Scassellati. Personalizing robot tutors to individuals’ learning differences. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 423–430. ACM, 2014.

11. T. Schodde, K. Bergmann, and S. Kopp. Adaptive robot language tutoring based on bayesian knowledge tracing and predictive decision-making. *Proceedings of ACM/IEEE HRI 2017*, 2017.

12. E. Senft, P. Baxter, J. Kennedy, and T. Belpaeme. SPARC: Supervised Progressively Autonomous Robot Competencies. In *International Conference on Social Robotics*, pages 603–612. Springer, 2015.

13. E. Senft, P. Baxter, J. Kennedy, S. Lemaignan, and T. Belpaeme. Supervised autonomy for online learning in human-robot interaction. *Pattern Recognition Letters*, 2017.

14. E. Senft, S. Lemaignan, P. Baxter, and T. Belpaeme. Toward Supervised Reinforcement Learning with Partial States for Social HRI. 2017.

15. P. Sequeira, P. Alves-Oliveira, T. Ribeiro, E. Di Tullio, S. Petisca, F. S. Melo, G. Castellano, and A. Paiva. Discovering social interaction strategies for robots from restricted-perception Wizard-of-Oz studies. In *Human-Robot Interaction (HRI), 2016 11th ACM/IEEE International Conference on*, pages 197–204. IEEE, 2016.

Toward Supervised Reinforcement Learning with Partial States for Social HRI

Emmanuel Senft
CRNS
Plymouth University
United Kingdom

Séverin Lemaignan
CRNS
Plymouth University
United Kingdom

Paul Baxter
L-CAS
University of Lincoln
United Kingdom

Tony Belpaeme
CRNS
Plymouth University (UK)
ID Lab – imec
Ghent University (BE)

Abstract

Social interacting is a complex task for which machine learning holds particular promise. However, as no sufficiently accurate simulator of human interactions exists today, the learning of social interaction strategies has to happen online in the real world. Actions executed by the robot impact on humans, and as such have to be carefully selected, making it impossible to rely on random exploration. Additionally, no clear reward function exists for social interactions. This implies that traditional approaches used for Reinforcement Learning cannot be directly applied for learning how to interact with the social world. As such we argue that robots will profit from human expertise and guidance to learn social interactions. However, as the quantity of input a human can provide is limited, new methods have to be designed to use human input more efficiently. In this paper we describe a setup in which we combine a framework called Supervised Progressively Autonomous Robot Competencies (SPARC), which allows safer online learning with Reinforcement Learning, with the use of partial states rather than full states to accelerate generalisation and obtain a usable action policy more quickly.

Introduction

Human-Robot Interaction (HRI) studies how people and robot can co-exist in society, and how they can interact socially in different environments and contexts. Robot are expected to behave appropriately regardless of the domain of interaction. However, it is impossible to foresee all possible interaction outcomes in dynamic and open social domains, as such the robot's responses cannot be implemented in the robot before its deployment in the real world. Similarly to people, robots need to be able to learn how to complete tasks through creating and optimising action policies. This includes learning social norms and how to make sense of the social world.

For a robot, interacting in the real world often requires taking in a diverse and large range of sensory inputs, which results in a high-dimensional sensory space. The robot then is required to select actions based on its current state in sensor space. However, parts of the state can be irrelevant to the current goal, and as such should not be taken into account when selecting the current action. Often, only a limited number of features in the space are important. To interact efficiently, a robot has to learn to identify these salient features and associate them with appropriate actions.

In previous work (Senft et al. 2015; 2017), we introduced the Supervised Progressively Autonomous Robot Competencies (SPARC) as a way to teach a robot an action policy while interacting based on a human supervisor intervening and correcting actions before they were executed by the robot. In this paper, we propose to extend this approach to allow the supervisor to highlight features in the environment relevant for the selected action. During the action selection phase, the robot can compare these features, defined as *partial states*, with the current state to select an action. The selected action is presented to the human supervisor, who can either correct the proposed action or approve it for execution.

Background

Reinforcement Learning

The main framework for an agent to learn how to interact in an environment while interacting is Reinforcement Learning (RL) (Kober, Bagnell, and Peters 2013; Sutton and Barto 1998). In RL, an agent interacting in an environment receives numerical rewards in reaction to its actions. The agent subsequently learns an action policy to maximise the expected cummulative discounted reward.

In many cases where RL achieves success, the agent has access to a virtual environment where the only real cost of exploring is computational effort: the agent can interact as long as needed to gather enough information on the environment and the result of its actions on the environment in order to find a sufficiently optimal action policy. Using simulation RL can achieve impressive results, as shown in RL learning to play Backgammon in the 90s (Tesauro 1995) to the more recent success in mastering the game of Go (Silver et al. 2016). Even when a virtual environment is available, but especially when it is not, human knowledge and effort is required during the design of the algorithm or the learning phase before the learning can be successful.

Human impact on Reinforcement Learning

Initial knowledge is often crucial to allow an algorithm to learn an efficient action policy. This knowledge, originating from human expertise, can be exploited in many ways.

Design decisions: Initial knowledge has to be used in the design of the algorithm, the representation of the state and actions spaces and the reward funtion. For example, only carefully crafted features allowed Tesauro to improve its algorithm for Backgammon from a intermediate-level player to super-human level (Tesauro 1995). Similarly, the design of complex neural networks for state generalisation, the representation of actions as motor primitives rather than raw motor angles or adding additional information in the reward function have important impacts on the ability of the robot to reach a successful policy (Kober, Bagnell, and Peters 2013).

Demonstrations: Initial knowledge can be provided to the agent through demonstrations. These demonstrations can be used to create an initial policy which is sufficiently efficient to start interacting in the environment and gather information to improve over time. This initial policy, required to receive meaningful feedback from the environment can be impossible to reach by relying only on random exploration and feedback from the environment. For example the game of Go in its larger board size contains more than 10^{210} states, so exhaustive search is not possible. Silver et al. (Silver et al. 2016) started with supervised learning from Go masters' games to learn a decent enough policy and then proceeded using deep learning, self play and tree search to achieve super-human capabilities and the capacity to beat the best human players.

These demonstrations can also be used to learn a reward function. With Inverse Reinforcement Learning, the agent is not provided with a reward function, but derives it instead from a set of expert demonstrations. The agent can then explore around the demonstrated policy to optimise the reward function. With this approach, Abbeel and Ng achieved better than human control for a robotic helicopter (Abbeel and Ng 2004) based on demonstrations from experts and further exploration and autonomous learning.

Guiding the learning: Rather than solely providing initial knowledge to the agents, humans can also guide the agent during its learning, a method more resembling human teaching.

A first approach consists in sequencing the tasks the robot will face. This approach, known as "scaffolding" (Saunders, Nehaniv, and Dautenhahn 2006), progressively increases the difficulty and complexity of the task as the robot is learning to reach policies which would take prohibitively long without scaffolding.

Agents learning in environments providing rewards can also benefit from additional rewards from human teachers. Depending on the task and the environment, different ways exist to combine rewards from multiple sources, and studies show that augmenting rewards from the environment by human ones can speed up the learning and reduce the number of undesired behaviours (Griffith et al. 2013; Knox and Stone 2010; Judah et al. 2010).

When environments do not supply rewards, they can be replaced by human ones. For example, the TAMER framework (Knox and Stone 2009) derives a reward function from the human feedback and uses this function to evaluate the current behaviour. In a similar approach, MacGlashan et al. proposed COACH in (MacGlashan et al. 2017). COACH assumes that human rewards represent the advantage function, i.e. how much the current action is better than the current policy, allowing to adapt the reward function to the strategies used and the current state of the learner.

Another approach to guide the learning is to bias the action selection. In (Thomaz and Breazeal 2008), the authors propose using a human supervisor to supply an agent with both rewards and potential guidance indicating what the agent should pay attention to, or what action it should take next. They show that giving the power to the supervisor to bias the action selection can improve the learning, making it faster and safer.

In (Senft et al. 2015), we introduced the Supervised Progressively Autonomous Robot Competencies (SPARC). SPARC relies on a supervisor with the ability to control the robot's actions. This supervisor is presented with the action the robot is about to execute. He can then choose to cancel it, allow it to be executed, or can select an alternative action. Based on the supervisor's decisions, the robot learns which actions are desired and can as such refine its policy over time, thus progressively reducing the need for the supervisor to correct or select an action.

SPARC gives control of the robot's actions to an expert, who can guide the exploration in the desired part of the environment which ensures the robot's behaviour is consistently appropriate. As the exploration is guided, and all the actions are useful, the learning can be faster than autonomous learning or learning based on human feedback as illustrated in Figure 1. In Autonomous Learning, the agent has first to discover its environment to start gathering relevant feedback, leading to a low initial performance and a slow improvement in early stage of learning. With teaching based on human feedback, the teacher can quickly provide information on actions to reach an efficient policy more quickly. However, as the teacher only provides feedback and cannot prevent the agent making mistakes, the initial performance can be poor. On the other hand, SPARC, with the control of the teacher over the executed actions, can prevent the agent from making mistakes in early stages, achieving a high performance even at the start of the learning.

Compared to similar algorithms (Chernova and Veloso 2009; Walsh et al. 2010), which allow the agent to request demonstrations and/or the teacher to provide demonstrations to correct mistakes made by the robot, SPARC allows the teacher to correct any action before its execution, thus reducing importantly the risk of the agent making errors. The total blending between autonomous execution of actions and demonstrations and the control over *every* actions executed by the agent are the specificities of SPARC.

In (Senft et al. 2017), we presented a way to combine SPARC and RL, by assigning a positive reward to every action executed by the robot, making the assumption that every action has been explicitly or implicitely approved by the supervisor. However this method directly mapped a single (state, action) pair to a reward without making use of any kind of generalisation. As such it was not applicable to

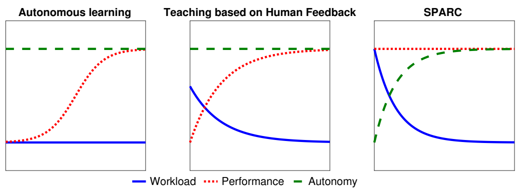


Figure 1: An illustration of the evolution over time of the performance, autonomy and human workload for an autonomous learner, an approach using human feedback, and SPARC.

environments with a continuous or high dimensional space or non-deterministic transitions from one state to another, elements which are typically present in social interactions. Similarly to TAMER or COACH, this approach allows to reproduce an action policy from a teacher even in the absence of a reward function, but as the teacher has control over the robot actions, the learning can be faster and safer.

Partial State Supervised Reinforcement Learning

To make RL applicable in high dimensional or continuous states, the algorithm requires a way to generalise knowledge to unseen states. A classic approach is to use a feed-forward neural network or deep learning to learn a the value function generalising to unseen states. Neural networks rely on having a large number of datapoints to converge toward a good function approximator. Alternatively, Cobo et al. proposed in (Cobo et al. 2011) to abstract features automatically from demonstrations to learn faster. But even in this case, the number of datapoints required is still high (around 1000 samples per participant). However, in many applications and especially in HRI, these amounts of data are not available or obtainable due to practical constraints. Furthermore, human responses are often noisy and lack consistency, so methods are needed which can generalise from a low number of noisy data points.

In this paper we propose to use a human user to highlight the relevant features of the environment to reduce the state dimension of the points only to relevant information. We introduce the concept of a *partial state*, a sliced version of the state defined only on a subset of the dimensions of the state. This shifts the (state, action) pair paradigm to (partial state, action) and in the case of RL, the tuple (state, action, reward) to (partial state, action, reward). This allows a comparison of the current state and the datapoints only on relevant features for action selection. With this instance-based method (Aha, Kibler, and Albert 1991), the algorithm can have a state abstraction allowing it to generalise even with a few datapoints instead of the large numbers normally required to abstract features from examples.

Learning algorithm

An expert supervises the agent actions, and can assign rewards to actions and highlight the parts of the states which are relevant to assigning this reward to this partial state.

As the supervisor can estimate the future impacts of an action, the problem of credit assignment for delayed rewards can be ignored which allows us to consider only a myopic approach in a fashion similar to TAMER (Knox and Stone 2009).

For this paper, we will reuse the formalism of rewardless Markov Decision Process to identify the different elements of our system. The agent has access to a set of actions $\mathcal{A}$ and a state $\mathcal{S} \in [0,1]^n$. We also define the partial states $\mathcal{S}' \in [0,1]^{n'}$ with $n' \leq n$ as a slice of $\mathcal{S}$, a subset of $\mathcal{S}$ where some dimensions have been removed.

When the agent executes an action a in state s, it receives the reward r associated with the partial state s'. For example, a state s could be defined in 4 dimensions such as $s = [1, 0.2, 0, 0.5]$, and s' in two dimensions with $s' = [-, 0.2, 0, -]$ with symbol '$-$' reprensenting the dimensions removed. For the learning algorithm, this means that the action a has been evaluated r in the partial state s'.

To each action $a \in \mathcal{A}$ we can associate a set $\mathcal{C}_a$ of pairs (s', r) representing the rating done by the supervisor to action a with features highlighted for the multiple s'. When adding a new pair (s', r), we can discard potential previous pairs with an identical s' to represent the evolution of the policy evaluation by the supervisor.

Algorithm 1: Algorithm for selecting an action based on the previous (partial state, action, reward) tuples and the current state.

```
inputs : Current state $s$, set of $(a, s', r)$
output: selected action $\pi(s)$
foreach $a \in \mathcal{A}$ do
    foreach $p = (s', r) \in \mathcal{C}_a$ do
        compute similarity $\Delta$ between $s$ and $s'$:
        $\Delta(p) = 1 - \frac{\sum_i^{n'} (s'(i) - s(i))^2}{n'}$
    find closest pair $\hat{p}$:
    $\hat{p} = arg\,max_p \Delta(p)$
    compute expected reward $\hat{r}(a)$ for taking $a$ in $s$:
    $\hat{r}(a) = \Delta(\hat{p}) \cdot r(\hat{p})$
    with $r(p)$ the reward $r$ of the pair $p = (s', r)$
Select the action with the maximum expected reward:
  $\pi(s) = arg\,max_a \hat{r}(a)$
```

When facing a new state s where an action has to be selected, the agent can select an action following Algorithm 1 in a instance-based learning fashion. For each action $a \in \mathcal{A}$, we take the pair (s', r) with the closest s' to the current state (as defined by the average quadratic distance over the normalised dimensions where s' is defined). That way, each action a can be associated to an expected reward defined by the product between the similarity of the closest partial state known for a and the reward obtained for executing a in that partial state. Finally, the action with the highest expected reward can be selected.

The normalisation of each dimension of the state allows distances to be comparable as values on all dimensions have the same range. Additionally taking the average quadratic distance over each defined dimension allows to compare

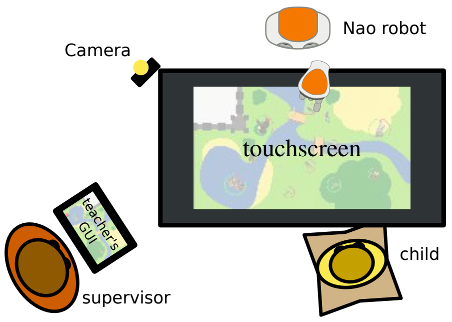


Figure 2: Interaction setup: the child and the robot are interacting on the touchscreen and a supervisor can control the robot using a GUI on a tablet.

similarities even when states are defined on a different number of dimensions.

Combination with SPARC

SPARC has been shown to be compatible with RL in (Senft et al. 2017), and can also be easily combined with the approach presented in this paper using partial states. For example, when selecting an action for the robot to be executed, the supervisor can also select which features in the environment should be selected as the partial state, and this associates the reward to this action in this partial state. Similarly, when an action is proposed to the supervisor the features represented by the dimensions of the closest partial state for this action can be exposed to the supervisor as a way to explain why this action has been selected. Facing this, the supervisor can: (1) not react, allowing the action to be executed and associating a reward of +1 to the partial state and the action proposed, (2) change the partial state to correct the features related to this action or (3) cancel it, preventing the execution and associating a reward of -1 to the partial state identified by the robot or the supervisor.

Application scenario

An example of an application is a social robot which interacts with children in an educational scenario. The robot plays an educational game with children to teach them notions about diverse topics according to the needs of the teacher. For this example, a child and a robot are playing a game about the food web, teaching which animals eat which ones on a Sandtray (Baxter, Wood, and Belpaeme 2012). In addition to the child and the robot, an adult supervises the robot using a tablet with a Graphical User Interface (GUI) to teach the robot how to interact with the child as shown in Figure 2.

The GUI (Figure 3) is an augmented version of the game itself, which can be use to make the robot move items on the game by dragging them on the GUI or which can display actions proposed by the robot with a cancel button to refuse an action. For example in Figure 3 the robot proposes to move the eagle to the rat, and highlights (as shown by blue circles) the eagle and the rat. This indicate that features relevant to the eagle and the rat have been used to select this action.

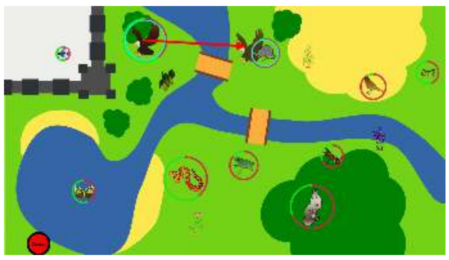


Figure 3: Interface for the supervisor with an action being proposed, moving the eagle to the rat highlighting both the eagle and the rat.

In the current implementation, the state is defined by the distance between each animal and their energy. With these features selected, the partial state transmitted to the user is the distance between the eagle and the rat, the eagle's energy and the rat's energy. Similarly, when selecting an action, the supervisor can select features in the state relevant to the action.

The main limitations of the approach reside in the difference of representation of the state and action spaces between the supervisor and the algorithm and the limit in communication. For example a user could try to move an animal close to another one, and depending on the representation of the actions on the algorithm side, the action might not be understood in the same way. Similarly, features used by the supervisor to select actions might not be represented in the state used by the algorithm. And lastly, in the case of implicit selection of features, a single case of features representation (for example highlighting two animals) might not be perceived in the same way by observers.

Future work

The system presented in the previous section will be improved and evaluated in the real world with children in the next months.

The current work could also be extended to allow the agent to continue to improve its behaviour even in the absence of a supervisor, progressively exploring around the learnt policy improving its behaviour beyond the demonstrations. This could be done by allowing the supervisor to provide rewards during the learning phase and combine these rewards with the demonstrations to learn a reward function in a fashion similar to Inverse Reinforcement Learning (Abbeel and Ng 2004) or TAMER (Knox and Stone 2009). This could also use partial states associated to these rewards to ease the generalisation of the reward function with only a low number of datapoints. However, without the supervisor, the assumption that only a myopic action selection is sufficient would not hold anymore and the problem of delayed rewards would have to be tackled.

Acknowledgments

This work was supported by the EU FP7 DREAM project (grant no. 611391) and EU H2020 Marie Sklodowska-Curie Actions project DoRoThy (grant 657227).

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*.

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine learning* 6(1):37–66.

Baxter, P.; Wood, R.; and Belpaeme, T. 2012. A touchscreen-based sandtray to facilitate, mediate and contextualise human-robot social interaction. In *Human-Robot Interaction (HRI), 7th ACM/IEEE International Conference on*, 105–106.

Chernova, S., and Veloso, M. 2009. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research* 34(1).

Cobo, L. C.; Zang, P.; Isbell Jr, C. L.; and Thomaz, A. L. 2011. Automatic state abstraction from demonstration. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22.

Griffith, S.; Subramanian, K.; Scholz, J.; Isbell, C.; and Thomaz, A. L. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, 2625–2633.

Judah, K.; Roy, S.; Fern, A.; and Dietterich, T. G. 2010. Reinforcement learning via practice and critique advice. In *AAAI*.

Knox, W. B., and Stone, P. 2009. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, 9–16.

Knox, W. B., and Stone, P. 2010. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 5–12.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32(11):1238–1274.

MacGlashan, J.; Ho, M. K.; Loftin, R.; Peng, B.; Wang, G.; Roberts, D. L.; Taylor, M. E.; and Littman, M. L. 2017. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning*.

Saunders, J.; Nehaniv, C. L.; and Dautenhahn, K. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 118–125.

Senft, E.; Baxter, P.; Kennedy, J.; and Belpaeme, T. 2015. Sparc: Supervised progressively autonomous robot competencies. In *International Conference on Social Robotics*, 603–612.

Senft, E.; Baxter, P.; Kennedy, J.; Lemaignan, S.; and Belpaeme, T. 2017. Supervised autonomy for online learning in human-robot interaction. *Pattern Recognition Letters*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

Tesauro, G. 1995. Temporal difference learning and td-gammon. *Communications of the ACM* 38(3):58–68.

Thomaz, A. L., and Breazeal, C. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence* 172(6):716–737.

Walsh, T. J.; Subramanian, K.; Littman, M. L.; and Diuk, C. 2010. Generalizing apprenticeship learning across hypothesis classes. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 1119–1126.