Imperial College
London

USER MODELLING FOR ROBOTIC COMPANIONS USING STOCHASTIC CONTEXT-FREE GRAMMARS

MIGUEL SARABIA DEL CASTILLO

Thesis submitted for the degree of Doctor of Philosophy

Supervised by DR YIANNIS DEMIRIS

Personal Robotics Lab

Department of Electrical and Electronic Engineering

Imperial College London

August 2015

COPYRIGHT DECLARATION

ORIGINALITY DECLARATION

I hereby declare that this thesis and the work herein detailed, was composed and originated by myself, except where appropriately referenced and credited.

London, August 2015

Miguel Sarabia del Castillo

ABSTRACT

Creating models about others is a sophisticated human ability that robotic companions need to develop in order to have successful interactions. This thesis proposes user modelling frameworks to personalise the interaction between a robot and its user and devises novel scenarios where robotic companions may apply these user modelling techniques.

We tackle the creation of user models in a hierarchical manner, using a streamlined version of the Hierarchical Attentive Multiple-Models for Execution and Recognition (HAMMER) architecture to detect low-level user actions and taking advantage of Stochastic Context-Free Grammars (SCFGs) to instantiate higher-level models which recognise uncertain and recursive sequences of low-level actions.

We discuss a couple of distinct scenarios for robotic companions: a humanoid sidekick for power-wheelchair users and a companion of hospital patients. Next, we address the limitations of the previous scenarios by applying our user modelling techniques and designing two further scenarios that fully take advantage of the user model. These scenarios are: a wheelchair driving tutor which models the user abilities, and the musical collaborator which learns the preferences of its users.

The methodology produced interesting results in all scenarios: users preferred the actual robot over a simulator as a wheelchair sidekick. Hospital patients rated positively their interactions with the companion independently of their age. Moreover, most users agreed that the music collaborator had become a better accompanist with our framework. Finally, we observed that users' driving performance improved when the robotic tutor instructed them to repeat a task.

As our workforce ages and the care requirements in our society grow, robots will need to play a role in helping us lead better lives. This thesis shows that, through the use of SCFGs, adaptive user models may be generated which then can be used by robots to assist their users.

Real stupidity beats artificial intelligence every time.

— Sir Terry Pratchett

ACKNOWLEDGEMENTS

Since every PhD thesis starts by acknowledging one's supervisor, there is always a risk of sounding vacuous. I would really like to avoid that here; yes, Yiannis has provided me with a great working environment, shielded me from distractions, pointed out the opportunities I should not miss and been a constant source of great discussions and ideas (and work!). But in the end, what I am most thankful for is that no matter how discouraged I was by the apparent lack of progress or stressed by deadlines, every single time I have walked into a meeting with Yiannis, I have come out energised to carry on with my research.

During the past 8 years, I have been —in one way or another— a part of the Personal Robotics Lab and been fortunate to meet and call my friends many of its members. I thank Tom, Simon, Paschalis, Yan, Jon, Eris, Arturo, Yanyu, Alex, Yixing, Theo, Martina, Hyung Jin, Maxime, Tobi, Josh, Ryan, Darío, Ayse and Dimitrios. These are some of the smartest people I know, and deserve more than just a passing mention. I have loved all our outings, coffee breaks and lunches. I am certain I will miss you.

There are a few Lab members without whom I would not have written *this* thesis, Murilo who convinced me it would be cool to do a PhD and has been my most staunch supporter, Harold who taught me so much about how to be an effective researcher and first built ARTY, KyuHwa who introduced me to SCFGs (and now they are on the title of my thesis!), Dimitri who drilled into my head that "if you know what you are doing, then it's not research" and Raquel for her friendship and advice all those times during my PhD when I tapped her shoulder and scared her to talk about anything including, but not limited to, research, politics and literature.

I want to acknowledge Patrick, our super-star administrator who helped me through all the PhD bureaucracy, Phil who runs the mechanical workshop and upgraded our wheelchairs as well as Liam and Max who gave me the opportunity to teach C++ to our undergraduates, something which I have thoroughly enjoyed (except for the marking!).

Credit is due to my friends who have at one point or another have endured (mostly stoically) my whining and or raving about the PhD. Thank you Sam, Julia, Pedro Costa, Aikaterini, Dave, Patricio, Dídac, Gokce, Giorgia, Alberto, Pol, Iñaki, Aksat, Rebecca,

Ioana, Andy, Naomi, Eugene, Nair, Michelle, Tempest, Sunny, Charo, Pedro Vitória, Lorenzo, Igor and Víctor.

My gratitude goes to my colleagues at the Chelsea and Westminster Hospital, in particular Marcela and Noel. It was an incredible experience to carry out the robotic user study in the hospital with you!

During these past 5 years I have been able to maintain myself thanks to the generous support of my sponsors: the Engineering and Physical Sciences Research Council and the ALIZ-E project from the European Union Framework Programme 7.

I am grateful to my family: Guille and Jacobo my ingenious brothers who keep me on my toes and yet can make me laugh at their will, my grandma who has impatiently been wanting me to finish and find a proper job since the day I started and Pablo who has lived the highs and lows of the PhD ride at the same time as me.

Finally, I want to thank mum, for her love and support and for making sure that I did not forget there are other things in life apart from robots (surprising as that sounds).

And dad, who —though he will deny it— has worried more about this thesis than myself and has, somewhat miraculously, managed to read every piece of text I have written.

CONTENTS

1 INTRODUCTION 19

1.1 Motivations, aims and objectives 21

1.2 Contributions 21

1.3 Thesis outline 23

1.3.1 Outline of appendices 24

2 BACKGROUND 27

2.1 Action recognition 27

2.1.1 Action recognition with HMMs 28

2.1.2 SCFGs for recognition of composite actions 29

2.2 Intelligent tutoring systems 30

2.3 Requirements for user modelling frameworks 31

2.4 Smart powered wheelchairs 32

2.5 Music generation 33

2.6 Human-robot collaboration and robotic companions 34

2.6.1 Robots in clinical settings 36

2.6.2 User modelling 38

2.6.3 Robotic tutors and teachers 38

2.7 Conclusion 39

3 ALGORITHMS AND SOFTWARE FOR USER MODELLING 41

3.1 HAMMER overview 41

3.1.1 Building blocks 42

3.1.2 Multiple models for execution and recognition 44

3.1.3 Hierarchical and attentive 44

3.1.4 Key features 45

3.2 HAMMER middleware implementation 46

3.2.1 Active vision: an application of the HAMMER middleware 49

3.3 Stochastic Context-Free Grammars overview 50

3.3.1 Scan 51

3.3.2 Complete 52

3.3.3 Predict 53

3.3.4 Viterbi parse 54
3.3.5 Prediction of next input 55
3.3.6 Initialisation, execution and termination 56
3.3.7 Parsing complexity 56
3.4 The SARTParser open-source library 57
3.4.1 From offline executable to online library 57
3.4.2 SARTParser API overview 59
3.5 Dancing NAO with HAMMER and SCFGs: a case study 60
3.5.1 State representation and acquisition 63
3.5.2 HAMMER's confidence function 64
3.5.3 HAMMER's inverse and forward models 65
3.5.4 HAMMER's finite state machine for full dance detection 65
3.5.5 SCFG's low-level detectors 65
3.5.6 Dance stochastic grammar 66
3.6 Conclusion 68

4 ROBOTIC COMPANIONS FOR WHEELCHAIR USERS AND HOSPITAL PATIENTS 71
4.1 A robotic sidekick for wheelchair users 72
4.1.1 Wheelchair sidekick system description 73
4.1.2 Pointing obstacles to children in wheelchairs 76
4.1.3 Giving driving directions to adults 78
4.1.4 Discussion 83
4.2 A companion for hospital patients 84
4.2.1 Hardware and software description 84
4.2.2 Trial procedure 87
4.2.3 Results 89
4.2.4 Discussion 95
4.3 Conclusion 97

5 USER MODELLING APPLICATIONS FOR ROBOTICS 99
5.1 Modelling users abilities: a robotic tutor for wheelchair users 99
5.1.1 Implementing an ITS with SCFGs 100
5.1.2 A robotic tutor for wheelchair users 102
5.1.3 Discussion 110
5.2 Modelling users preferences: the synchronised grammars framework 112
5.2.1 Synchronised grammars framework 113
5.2.2 Synthetic analysis of the synchronised grammars framework 115

5.2.3 Musical human robot collaboration study 117
5.2.4 Discussion 121
5.3 Conclusion 122
5.3.1 Comparison between user modelling frameworks 123

6 CONCLUSIONS AND FUTURE WORK 127
6.1 Summary of contributions 127
6.2 Limitations 128
6.3 Towards the future 129
6.3.1 Long-term studies 129
6.3.2 Studies with disabled children 130
6.3.3 Improving the tutoring skills by using the user model 130
6.3.4 Synchronisation between multiple SCFG parsers 131
6.3.5 A graphical interface for specification of stochastic tasks 131
6.4 Epilogue 131

P TOWARDS A PARALLEL STOLCKE-EARLEY PARSING ALGORITHM 133
P.1 Parallel algorithm description 134
P.1.1 Efficient parsing data structures 134
P.1.2 Probability representation 136
P.1.3 Scan 137
P.1.4 Predict 137
P.2 Preliminary results 141
P.3 Towards a parallel complete 141
P.4 Discussion and conclusion 142

R ROBOTS & SENSORS 143
R.1 ROS 143
R.2 ARTA 144
R.3 ARTY 145
R.4 Baxter 145
R.5 Reactable 146
R.6 NAO 146
R.7 Kinect 146

BIBLIOGRAPHY 147

AUTHOR'S PUBLICATIONS 161

LIST OF FIGURES

Figure 1.1 Pictures from each of the user studies carried out for this thesis. 22

Figure 1.2 Outline of this thesis. 25

Figure 3.1 HAMMER building blocks. 42

Figure 3.2 Diagrammatic representation of HAMMER. 43

Figure 3.3 Relationships between the main C++ classes of the HAMMER middleware. 46

Figure 3.4 Parsing speed comparison between the original and the current versions of SARTParser. 59

Figure 3.5 Dancing NAO sample activity implemented with HAMMER and SCFGs. 60

Figure 3.6 State representation for dancing NAO activity. 63

Figure 4.1 Different roles for NAO as a companion with diverse users. 72

Figure 4.2 Hardware components of the wheelchair sidekick system. 74

Figure 4.3 Software components of the wheelchair sidekick system. 74

Figure 4.4 Asteroids game designed to serve as a secondary task to increase the cognitive load of participants in our experiments. 75

Figure 4.5 Questionnaire responses of 14 children who tried the wheelchair sidekick system. 77

Figure 4.6 Path-driving environment. 78

Figure 4.7 Deviation from average in driving performance across path-driving experiments. 81

Figure 4.8 Questionnaire responses for Simulator and NAO considering all path-driving experiments attempts. 82

Figure 4.9 Preferred driving aids to give instructions to adults driving a wheelchair. 82

Figure 4.10 Overview of the hardware and software components of NAO as a hospital companion. 85

Figure 4.11 Overview of the remote operator interface. 86

Figure 4.12 Hospital trials population. 90

Figure 4.13 Qualitative and quantitative results of interactions with hospital companion. 91
Figure 4.14 Hospital companion activities by age group. 93
Figure 4.15 Timelines of interactions with 4 patients with dementia aged 84–99. 94
Figure 5.1 Main components of an Intelligent Tutoring System. 100
Figure 5.2 Map of the driving environment for the robotic tutor study. 103
Figure 5.3 Hardware components of our robotic tutoring system. 105
Figure 5.4 User study's questionnaire responses for users in both categories. 108
Figure 5.5 Boxplots showing driving scores for each user who was instructed to repeat a path by the robotic tutor. 109
Figure 5.6 Correct and incorrect evaluations made by baseline users of their own task performance. 110
Figure 5.7 Classification of participants usage of the robotic tutor hints. 110
Figure 5.8 Flowchart representation of the synchronised grammars framework. 113
Figure 5.9 Main components of our musical collaboration system with Baxter and Reactable. 118
Figure 5.10 Number of times users gave feedback to the musical collaboration system by participant. 120
Figure 5.11 Results of the questionnaire filled in by participants after the musical collaboration trials. 121
Figure 5.12 Visual comparison between the three different architectures for user modelling we have discussed in this thesis. 123
Figure P.1 Visual representation of the GPU encoding of stochastic grammar rules. 135
Figure P.2 Visual representation of a parsing state in a GPU. 135
Figure P.3 Pseudo-code for parallel scan algorithm. 138
Figure P.4 Visual description of sum-scan. 138
Figure P.5 Pseudo-code for parallel scan algorithm. 139
Figure P.6 Speed comparison between the CPU and the GPU implementations of scan and predict. 140
Figure R.1 Robots and other hardware used for this thesis. 144

LIST OF TABLES

Table 3.1 Example of a Viterbi parse. 54

Table 4.1 Example of patients' comments. 96

Table 5.1 Grammar definitions for each of the three wheelchair driving tasks. 104

Table 5.2 Grammar definitions for: influencing (user) grammar and base (robot) grammar. 116

Table 5.3 Approaches to low-level action recognition, composite action recognition (high-level action recognition) and user modelling by each of the frameworks discussed in this thesis. 125

LISTINGS

Listing 3.1 Sample instantiation of the HAMMER building blocks in C++. 48

Listing 3.2 Sample `main()` function for HAMMER. 49

Listing 3.3 SARTParser C++ usage example. 61

Listing P.1 C++ code to add two numbers in log-space together. 136

ACRONYMS

AMCL	Adaptive Monte-Carlo Localisation
ARTA	Assistive Robotic Transport for Adults
ARTY	Assistive Robotic Transport for Youngsters
API	Application Programming Interface
CPU	Central Processing Unit
CRF	Clinical Research Fellow
dof	degrees of freedom
FSM	Finite State Machine
GPU	Graphics Processing Unit
HAMMER	Hierarchical Attentive Multiple-Models for Execution and Recognition
HMM	Hidden Markov Model
HRI	Human-Robot Interaction
ITS	Intelligent Tutoring System
IQR	interquartile range
JSON	JavaScript Object Notation
PID	proportional-integral-derivative controller
ROS	Robot Operating System
SCFG	Stochastic Context-Free Grammar
WoZ	Wizard of Oz
ZPD	Zone of Proximal Development

MATHEMATICAL NOTATION

This thesis follows a straight-forward mathematical notation. Vectors and matrices are denoted in bold (eg. $\mathbf{M}$), and in particular $\mathbf{I}$ represents the identity matrix. The element at the i^{th} row and the j^{th} column of a matrix is denoted as $\mathbf{M}\{i, j\}$. We use $\odot$ to denote element-wise matrix multiplication.

Sets are shown in calligraphic lettering (eg. $\mathcal{X}$) with the exception of $\mathcal{S}$ which denotes the axiom of a SCFG. The Kleene star ($\mathcal{X}^*$) is used to represent zero or more elements of a set. $|\mathcal{X}|$ is the cardinality (the number of elements) of a set.

As expected, $P(x)$ represents the probability of the random variable x. $\mathbf{P}(\mathcal{X})$ is the vector that contains the probabilities of the elements of the $\mathcal{X}$ set: $P(x)\ \forall x \in \mathcal{X}$. Sometimes we normalise these probability vectors with the following operation:

$$\mathbf{P}'(\mathcal{X})\{x\} = \frac{P(x)}{\sum_{y \in \mathcal{X}} P(y)}$$

where $\mathbf{P}'(\mathcal{X})$ is the new normalised vector of probabilities. Normalisation is used to ensure $\mathbf{P}'(\mathcal{X})$ contents sum to 1.0.

STATISTICAL TESTS

Throughout this thesis we employ non-parametric tests:

CHI-SQUARED χ^2: we use this test to establish whether there is a link between a categorical predictor and a categorical outcome. For example, this test would be used to check if gender had any effect on whether a subject talked to the robot.

SPEARMAN CORRELATION: we used this to measure the correlation between two continuous variables. For instance, this test may be used to verify the correlation between age and number of times a user touched the robot.

POINT-BISERIAL CORRELATION: this statistical test is used to measure the correlation between a categorical variable and a continuous variable. We could for instance apply this test to verify if the age of a patient had any effect on whether she completed an experiment trial.

MANN-WHITNEY U: this test establishes whether two different sets of continuous measures share the same underlying distribution. Note, the Mann-Whitney U test may also be applied to ordinal data, such as Likert-scale rankings. This is the most used test in the thesis. An example application of this test is to establish whether having used a robot before had any effect on the duration of interaction.

WILCOXON SIGNED-RANK MATCHED PAIRS: this test is very similar to Mann-Whitney U, but we use it when the two sets of continuous measures were obtained from the *same* subject.

FRIEDMAN: this analogous is to Wilcoxon signed-rank matched pairs test but allows to test three sets of measures or more. It may be used to verify whether there were speed differences across four wheelchair driving trials.

The statistical significance threshold is set $p < 0.05$. We are aware that p-values are not without issues (Ioannidis, 2005; Nuzzo, 2014) and thus we report the effect size of all our findings as well as some measurements that do not reach the preceding threshold.

1

INTRODUCTION

A key aspect of our interactions with other people is our ability to form *models* about others which determine our actions. For instance, I would not speak in Spanish to my supervisor as my *model* of him tells me he probably would not be able to understand me. Though natural to humans, forming models about others is a sophisticated skill, in fact it is a matter of debate whether other animals share this skill (van der Vaart and Hemelrijk, 2014).

Most machines today completely lack models about their users. This does not represent a problem as long as the interactions with the machines are short and impersonal—like paying for our groceries with a self-checkout machine.

Robots are different though. We expect them to behave like humans (Dautenhahn et al., 2005; Kidd and Breazeal, 2004). This is particularly true for the type of robots on which this thesis focuses: companion robots, that is robots that accompany users throughout a task providing assistance for said task. The key difference between companion robots and other types of robots (eg. industrial robots) is that companion robots have to adapt to their users (Fong, Nourbakhsh and Dautenhahn, 2003).

It would be unacceptable if our driving instructor taught us where the throttle is at the beginning of *every* lesson. We assume a driving instructor will tailor what she teaches to our level. This goes beyond remembering what the previous lesson was, the driving tutor needs to know what the user knows. The same holds true for robotic companions.

Nevertheless, the abilities or preferences of users are not directly observable, they need to be inferred from the (noisy) observations of the *user actions*. This is what makes user modelling an interesting research problem.

In order to be able to work at the appropriate level of abstraction, we split the *user actions* into low-level actions and high-level actions. Low-level actions are the actions that can be directly detected (for instance, the user's left arm is raised). High-level actions are composed of sequences of low-level actions. An example of a high-level action would be a dance, which is composed of several dance steps.

Since we intend to develop a flexible framework, rather than an ad-hoc solution, for user modelling we need a representational formalism for high-level actions. Our approach is to make use of Stochastic Context-Free Grammars (SCFGs) in order to rep-

resent the tasks the user will be performing. SCFGs have been widely used in the action recognition literature (as we will see in section 2.1.2) and offer three compelling features for user modelling than other high-level action recognition algorithm do not: they allow for the representation of hierarchical and recursive tasks in an intuitive and formal fashion; furthermore, they let us to predict what are the next likely steps the user will perform; finally, they provide a simple measure of how well did the user actions conform to the task specification. Using these features, we can extract the user model by analysing the differences between the user performance and the task specification.

SCFGs operate on a high-level actions, and in consequence, we need to recognise the low-level actions which compose the user tasks. We achieve this by implementing a variant of the Hierarchical Attentive Multiple-Models for Execution and Recognition (HAMMER) architecture. Essentially, we run many low-level detectors concurrently and interpret the one with the highest confidence as the action the user is currently performing.

Any task that is comprised of a series of steps may be represented with an SCFG. Thus the frameworks we present in this thesis may be applied to a large number of problems beyond the ones presented in our case studies.

The user model, which we obtain through the SCFGs, can subsequently be used to alter the interaction of a robotic companion. In this thesis, we take advantage of the user model in two ways: by proposing a new task adapted to the user abilities and by changing the robot actions to better conform to the user predicted preferences. The robotic companions we present in this thesis utilise the user model to be of better assistance to their users.

Alongside, the development of the computational framework for user modelling, we present several user studies with robotic companions. We first introduce two companions which do not model the user: a robotic sidekick for wheelchair users and a remotely operated hospital companion. Though well-liked by users, these companions were limited in their interactions. Follow-up user studies with and a wheelchair tutor and a musical collaborator showcase how robotic companions become more useful with user models, since neither tutoring nor collaborating musically are roles that may be successfully performed without a user model.

Our focus on robotic companions is grounded in the belief that these can improve all of our lives. That is why the case studies we conducted for this thesis span a wide segment of the general population: from adults of all ages, to patients with dementia, to (healthy) children.

1.1 MOTIVATIONS, AIMS AND OBJECTIVES

To summarise, the main motivations of this thesis are:

- We are interested in learning models about users which influence the actions of robots.
- We focus on companion robots with the aim of providing personalised assistance to the user.

These motivations determine the following objectives for the thesis:

- We intend to devise new scenarios where a user may benefit from a companion robot.
- We aim to develop flexible frameworks for user modelling.
- Specifically, we want to model the abilities as well as the preferences of users. The representation of the models needs to be intuitive as it may need to be understood by other people than roboticists.
- Our purpose is to validate these frameworks with actual Human-Robot Interaction (HRI) studies.

In what follows, we outline our contributions towards meeting these objectives.

1.2 CONTRIBUTIONS

This thesis makes three types of contributions: algorithms, user studies, and open-source software.

ALGORITHMS We have designed several algorithms with the goal of capturing specific attributes of the user:

- We present a formal description of how to obtain the probability distribution of the most-likely next steps of a task represented as a SCFG.
- We devise a variant of the HAMMER architecture (Demiris and Khadhouri, 2006), specially tailored for recognition of tasks represented as SCFGs.

(a) Wheelchair sidekick study. (b) Wheelchair tutor study. (c) Hospital companion study. (d) Musical collaboration study.

Figure 1.1: Pictures from each of the user studies carried out for this thesis.

- We contribute a framework for modelling users' abilities which combines the aforementioned architecture with an Intelligent Tutoring System (ITS).
- We further contribute a second framework to learn the user preferences in a human-robot collaboration scenario where the user and the robot tasks are characterised by different grammars. Our framework is able to compensate between contradictory constraints, such as the robot choosing between the action the task definition tells it to do in and action the user would prefer it does.

USER STUDIES We have carried out the following four user studies with companion robots (fig. 1.1):

- A user study with a humanoid robot acting as a driving guide in a robotised power-wheelchair. The humanoid can point to obstacles and give driving directions (fig. 1.1a). Results show that users much preferred the physical robotic companion over a simulated robotic companion.
- In a follow-up study, we program the humanoid robot to act as a wheelchair driving tutor (fig. 1.1b). We find that users that repeated a task when the companion instructed them to do so improved their performance score.
- A user study with a humanoid robot acting as a companion for adult hospital patients of all ages, from 18 to 100 (fig. 1.1c). Patients were selected from six different acute-care wards in the hospital; they had been admitted to the hospital for a variety of reasons: dementia, falls, surgery, infections, etc. We found that most patients enjoyed their interaction and that the robot managed to engage several patients with dementia.
- An adaptive musical collaboration study where a humanoid robot creates the drum accompaniment for the user's melody using a tangible music table as a proxy

for interaction (fig. 1.1d). The study shows that 75% of participants agreed that the robot had become a better accompanist as trials progressed.

SOFTWARE Finally, we have developed two high-quality open-source libraries to be freely used by the community:

- A generic implementation of the HAMMER architecture which has been used by other researchers to investigate active vision with the iCub humanoid.
- A modernised and sped-up implementation of a reference SCFG parser. Our modernisation efforts yielded a 10x improvement in a synthetic benchmark over the original version of the parser.

1.3 THESIS OUTLINE

We now look at how the different chapters in this thesis are connected. Figure 1.2 provides a visual summary of this outline.

- The next chapter reviews the research relevant to the thesis. The following research fields will be covered: action recognition, intelligent tutoring, smart powered wheelchairs, music generation, human-robot collaboration, robot companions, user modelling and robotic tutors.
- Chapter 3 lays the algorithmic foundation of this thesis. We will describe in detail the HAMMER architecture as well as parsing of SCFGs. This chapter also introduces the software implementations of these two algorithms. We finish the chapter by designing a dance recognition application using both HAMMER and SCFGs.
- Chapter 4 features a humanoid robotic companion in a two distinct roles: an autonomous wheelchair sidekick which points out obstacles and gives directions to the user, and a hospital companion which interacts interacting with patients through a Wizard of Oz (WoZ) interface. We report on the results of user studies and find that users are overwhelmingly positive about our humanoid companion with 85.7% of children who drove with the wheelchair sidekick strongly agreeing that they liked having the robot by their side and 84% of hospital patients agreeing the enjoyed the interaction with the companion.
- In chapter 5, we propose frameworks based on the algorithms introduced in chapter 3 to model the abilities and the preferences of the user. We further

present two user studies where these frameworks are applied to robotic companions. The first user study develops the humanoid companion for wheelchair users we presented in chapter 4 into a robotic driving tutor. In contrast, for the second user study we introduce a whole new scenario: adaptive musical human-robot collaboration. Both user studies serve as a successful proof-of-concept implementation of the two frameworks described here.

- We finish this thesis in chapter 6 by presenting our conclusions and sketching the future work to be done.

1.3.1 Outline of appendices

This thesis has two appendices. The first one, appendix P, deals with a possible parallelisation of the Stolcke-Earley parsing algorithm which will be introduced in chapter 3. The appendix shows how to implement several operations of the algorithm in a Graphics Processing Unit (GPU) as well as benchmarking these operations against a serial implementation.

Appendix R on the other hand contains detailed descriptions of the robots and sensors used throughout this thesis as well as a brief explanation of the Robot Operating System.

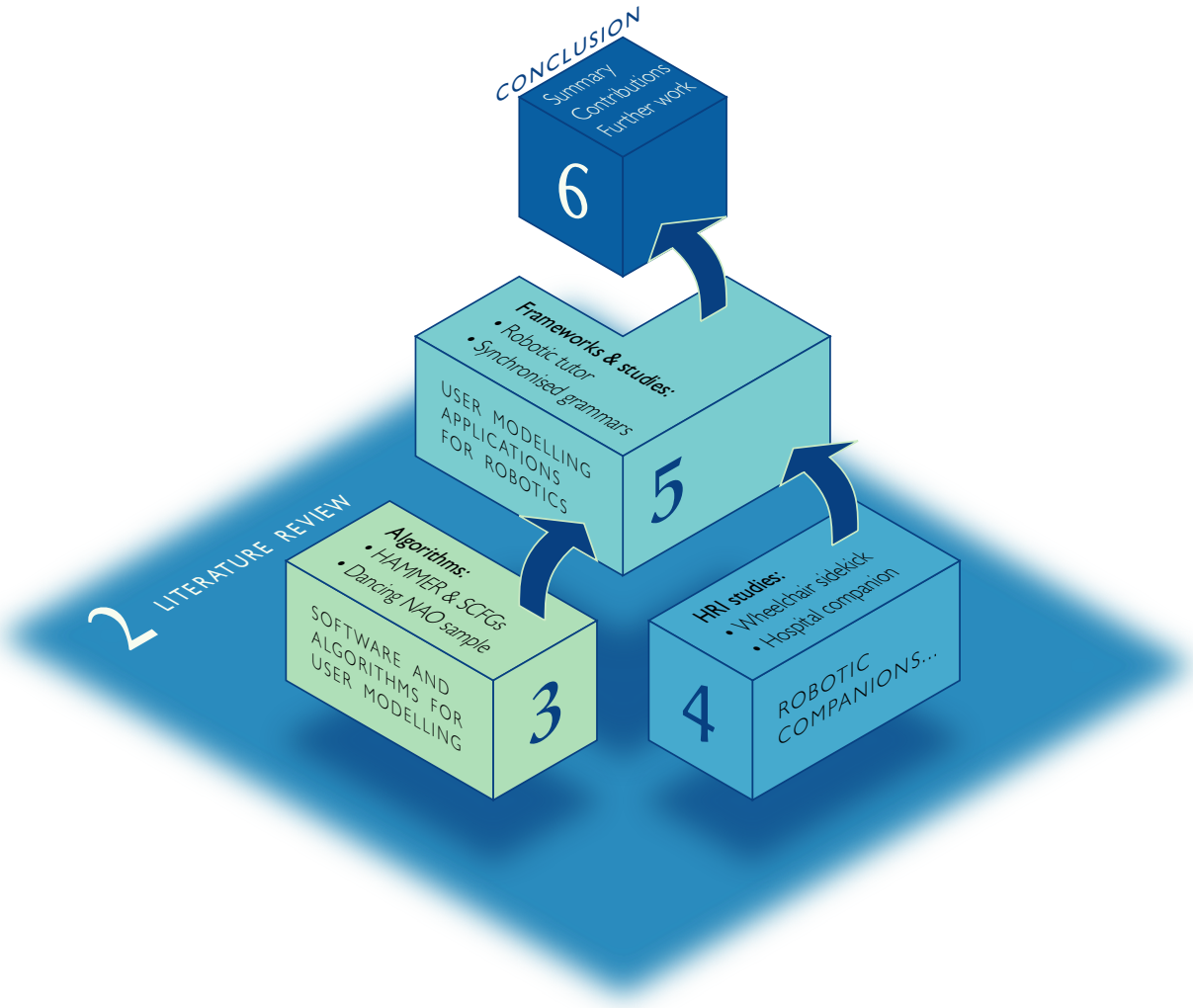


Figure 1.2: Outline of this thesis. Each box represents a forthcoming chapter of this thesis. The main contributions of each chapter are indicated at the top of their respective boxes.

2

BACKGROUND

As we saw in the introduction, this thesis presents several user studies: a robotic sidekick for wheelchair users, a musical collaborator, a hospital companion and a robotic wheelchair tutor. Moreover, we also develop frameworks for modelling the abilities and the preferences of the user. We have drawn inspiration from many fields of research to develop this work.

The purpose of this chapter is to review the most relevant articles of said research fields. Specifically, we will focus on human action recognition, intelligent tutoring systems, smart powered wheelchairs, music generation and Human-Robot Interaction (HRI). Within the field of HRI we will concentrate on human-robot collaboration, robotic companions, robots in clinical settings and user-modelling, since each of these sub-fields has contributed to our thesis.

2.1 ACTION RECOGNITION

Many automated systems need to be able to recognise the actions of humans. Though in this thesis we are motivated by the need to understand user actions so robotic companions can interact appropriately, there are other situations where automatic action recognition is relevant, eg. automated surveillance analysis, ambient intelligence or entertainment (Demiris, 2007).

Aggarwal and Ryoo (2011) provide an in-depth review of the main methods of human activity recognition in computer vision. They differentiate between *single-layered* approaches where the activity is recognised directly from the video input —be it through volume, trajectory or features comparison— and *hierarchical* approaches where the activity is classified depending on the classification of simpler activities. For this thesis, we make use of Stochastic Context-Free Grammars (SCFGs), which Aggarwal and Ryoo list as an example of hierarchical approach, to detect composite actions as we will discuss shortly.

The previous review is focused on computer vision, but it is also possible to apply many of the methods therein described to trajectories obtained from depth cameras. In particular, Shotton et al. (2011) describe how the Microsoft Kinect depth camera obtains

a human skeleton with randomised decision forests trained on a million labelled images. We will take advantage of this method in section 3.5 to create dance pose classifiers.

Krüger et al. (2007) wrote another review of action recognition with a focus in robotics. Amongst the articles in the review the work of Johnson and Demiris (2004) is specially pertinent to this thesis as it describes how to use the Hierarchical Attentive Multiple-Models for Execution and Recognition (HAMMER) architecture to recognise the actions of a demonstrator. HAMMER, as we will describe in detail in section 3.1, detects the demonstrator actions by concurrently executing different models and selecting the one that performs better. Furthermore, these models may be scaffolded so that higher-level models can recognise actions composed of low-level actions. The motor command *close gripper* is an instance of a low-level action, whereas *move can from one table to the other* an example of a high-level action. Johnson and Demiris (2004) use graph theory to build their high-level models. In this thesis, we will detect low-level actions in a very similar fashion, but for composite actions we will make use of SCFGs.

2.1.1 Action recognition with HMMs

Hidden Markov Models (HMMs) have long been used for action recognition. The first example is given by Yamato et al. (1992) which used HMMs to detect different tennis plays. Later, Starner and Pentland (1995) applied HMMs to automatically recognise the American sign language. More recently, Xia et al. (2012) have applied HMMs to three-dimensional joints time series captured by the Kinect (cf. appendix R.7) in order to recognise human actions.

SCFGs, the formalism we use in this thesis to represent tasks, have two advantages over HMMs. Firstly, with an SCFG we can *count*, that is request the user performs two events the same number of times. This is evidently not possible with HMMs due to the Markov assumption.

Secondly, SCFGs offer an intuitive representation that may be understood by non-experts. This makes them ideal to use in scenarios such as wheelchair-driving with disabled children where an operational therapist may want to specify a task on her own.

2.1.2 SCFGs for recognition of composite actions

Recognition of low-level actions does not give enough information about the current user activity to a robotic companion. As it has been previously stated, we recognise composite actions using Stochastic Context-Free Grammars (SCFGs), which we review now.

Stochastic Context-Free Grammars (SCFGs) —also known in the literature as Probabilistic Context-Free Grammars— have found use in fields such as natural language processing (Klein and Manning, 2003), ribonucleic acid analysis (Sakakibara et al., 1994), password cracking (Weir et al., 2009), as well as computer vision and robotics —which we will review shortly. The reason for their widespread adoption lies in the intuitiveness and simplicity of their formulation, allowing non-experts to understand and design their own grammars to encode tasks.

In 1956, Chomsky first formalised Context-Free Grammars with the aim of encoding natural language. Later, Earley (1970) described an efficient top-down parsing algorithm for Context-Free Grammars. Finally, Stolcke (1995) extended Earley's algorithm to Stochastic Context-Free Grammars (SCFGs) by adding inner and forward probabilities to the parsing states. The resulting algorithm is what we refer as the Stolcke-Earley parser, which will be described in detail in section 3.3. It is worth noting that neither Earley's nor Stolcke-Earley's require transforming the grammar into Chomsky's Normal Form (Stolcke, 1995) in contrast with other SCFG parsing algorithms such as CYK (Younger, 1967).

Ivanov and Bobick (2000) reports on one of the first examples of using SCFGs for action recognition. Using the Stolcke-Earley algorithm, they manage to successfully recognise human gestures as well as tracking people in the recording by a surveillance of a parking lot. Shortly afterwards, Minnen et al. (2003) used SCFGs to recognise a game of Towers of Hanoi, showing that stochastic grammars could recover from low-level detection errors. This property is one of the reasons behind our use of SCFGs since it allows us to successfully recognise a task even if the user does not perform it perfectly.

Ogale et al. (2007) continued exploring the applicability of SCFGs by using them to recognise more complex human actions —such as walking, turning, kneeling and kicking. The authors recorded a dataset of actions from 8 distinct viewpoints and then proceeded to extract key-frames from the dataset. A stochastic grammar was then generated from sequences of grouped key-frames, each group containing a key-frame from a different viewpoint. The advantage of this approach lies in that there is no need to explicitly link the 8 viewpoints, instead they were fed to the SCFG parser as input.

This computer vision research was brought into the field of robotics by Lee et al. (2012). The authors learnt a stochastic grammar for the Towers of Hanoi game in such a way that an iCub humanoid robot could recognise it and execute it. In a subsequent article, the authors applied the same technique to dance recognition and imitation (Lee et al., 2013). We will take a similar approach to recognise dancing steps in section 3.5.

All of these works have dealt with recognition of the actions of a single agent. In order to recognise the combined actions of several agents, researchers have merged Allen's temporal interval logic (1983) with SCFGs. We highlight the work of Ryoo and Aggarwal (2006) and Zhang et al. (2011). In the first case, Ryoo and Aggarwal detect low-level actions using HMMs, then they define the grammar terminals as the combination of low-level actions and a temporal operator from Allen's logic (eg. terminal t is defined as *action b before action a*). The authors use non-stochastic Context-Free Grammars rather than SCFGs as HMMs are already probabilistic and can deal with input noise. In contrast, Zhang et al. (2011) modifies the SCFG parser in such a way that it can reason about temporal logic during parsing. In both cases, there is only one grammar to represent the whole interaction. And, again in both cases, the generated grammar would be less intuitive and more computationally expensive to parse than two independent grammars. Our approach in this thesis to recognise multi-agent actions is to create two independent grammars, one per agent and then synchronise them (as we will present in section 5.2).

2.2 INTELLIGENT TUTORING SYSTEMS

Intelligent Tutoring Systems (ITSs) aim to teach humans new skills, be it of a theoretical or practical nature. ITSs is a sub-field of human-computer interaction and as such its focus is on the design of computer programs. Our reasons for reviewing ITS here are two-fold. Firstly, since these systems need a way of keeping track of the progress of its students —even something as rudimentary as a list of completed exercises— they serve as example applications of user modelling. Secondly, in section 5.1 we apply the findings of this field to a robotic tutor for wheelchair users. A key difference between ITSs and our research is that we obtain the user information from sensors rather than the inputs in a computer program.

Crucially, a review by VanLehn (2011) found that the learning effect[1] of a human tutor is very close to that of an ITS. This is relevant to us, as it shows that —in principle— a robotic tutor could obtain the same learning gains as a human tutor.

1 The learning effect was defined as the difference between mean exam scores before and after using the ITS divided by the standard deviation.

VanLehn (2006) presents an overview of all the behaviours that an ITS should exhibit. Briefly, VanLehn divides these behaviours in two: the inner loop and the outer loop. The inner loop, which executes as the student tries to solve the task at hand, attempts to determine whether the student requires hints or feedback. Critically, the inner loop is also in charge of assessing the student performance and updating the user model accordingly. The role of the outer loop is to select the most appropriate task to be taught next, possibly generating such a task if necessary.

There are many examples of ITSs in the literature. For instance, Gertner and VanLehn (2000) present *Andes* a tutor for students to learn about physics. Similarly, *AutoTutor* teaches students about science, technology, engineering and mathematics using natural language (Graesser et al., 2005). There are also more specialised tutors, such as *SQLTutor* —an online tutor to learn about database queries— by Mitrovic (2003) and *What* which is used to teach about the Haskell programming language (López et al., 2002).

D'Mello et al. (2012) showed an implementation of an ITS paired with a gaze tracking system that prompted students to pay attention to the tutor whenever their gaze drifted from the computer screen. Results show that students with the gaze reactive system had higher learning gains that students without. This finding inspired us to make our robotic tutor talk to the user when she became disengaged, though in our case we measured disengagement by the lack of input to a powered wheelchair.

2.3 REQUIREMENTS FOR USER MODELLING FRAMEWORKS

Previous research in the fields of student modelling and adaptive user interfaces is useful to identify which properties are desirable in a robotic user modelling framework.

Firstly, such a framework must be *adaptive*. This argument is showcased by Keates et al. (2002) which states that the "user can experience changes in their capabilities". The authors explain this in the context of a computer user interface which adapts to users with disabilities, though it is generalisable to any domain where different users may have different skill levels and those skill levels may change over time. Moreover, adaptiveness has been shown to improve the performance of users in computing tasks Trumbly et al. (1993).

Secondly, a user modelling framework must be *accurate*. Gajos et al. (2008) found that users preferred a user interface for an office suite which adequately suggested their next steps over a predictable, but less accurate, user interface.

Finally, a review of recent student modelling approaches by Chrysafiadi and Virvou (2013) indicates that managing *uncertainty* is another requirement for user modelling frameworks. Indeed, the authors highlight an increase of machine learning techniques in student modelling aimed at controlling uncertainty.

In summary, user modelling frameworks should be *adaptive*, *accurate* and able to manage *uncertainty* about the state of its user as well as her actions. These properties do not guarantee perfect user modelling, however as Self (1990) pointed out, a user model does not need to be perfectly precise to be useful.

2.4 SMART POWERED WHEELCHAIRS

Smart powered wheelchairs have equally been the subject of much research. A review of the field is presented by Simpson (2005). In 2008, it was estimated that 61% to 91% of wheelchair users may benefit from a smart powered wheelchair at some point during their lives (Simpson et al., 2008).

Carlson and Demiris (2010) used a precursor of the Assistive Robotic Transport for Adults (ARTA) to develop a collaborative control framework for wheelchairs. Their experiments with able-bodied users as well as a case study with a mobility-impaired user showed that the collaborative control framework reduced the cognitive load on users. In a follow-up article, this collaborative control framework was extended with a Brain-Computer Interface input system (Carlson and del R. Millán, 2013). The authors of these articles remark that the benefits of collaborative control come at the cost of a steeper learning curve, providing us with further encouragement to design a robotic tutor.

Another approach to smart power wheelchairs is showcased by Morales et al. (2013), where the authors programmed a fully automated wheelchair to follow the path that is the most comfortable for users. The authors define comfort in terms of speed at distance to obstacles. However, removing the autonomy of wheelchair users may entail losing the developmental benefits of having a wheelchair in the first place.

The need for powered mobility may be even more pronounced for children. Indeed, the Rehabilitation Engineering & Assistive Technology Society of North America supports the use of paediatric powered mobility as soon as the child possesses the necessary cognitive, sensorimotor and coping abilities (Rosen et al., 2009). According to the same study, the use of smart powered wheelchairs "enhances independence, improves psychosocial development and enables children to become productive and independent members of society".

Evans et al. (2007) analysed the use of powered wheelchairs by young people. Their results show that whilst there are many benefits to powered mobility such as "increased mobility" and the "ability to engage in more tasks", there are safety concerns as well — 10 out of 18 participants reported having had an accident with the power wheelchair. Moreover, the authors found a "demonstrated need for additional driving training".

Evidence of the benefits of smart wheelchairs comes from Montesano et al. (2010) where 4 children with cerebral palsy aged 11 to 16 successfully navigated around their school using a touch-screen. One of the main symptoms of cerebral palsy is random and uncontrolled body movements which prevents users from driving a regular power-wheelchair.

It has been shown that a robotic wheelchair with haptic guidance can be used to teach children to drive (Marchal-Crespo et al., 2010). This system, based on a game of *robot-tag*, was used by 22 able-bodied children and a 8 year old with cerebral palsy.

The Assistive Robotic Transport for Youngsters (ARTY), which we use for this thesis and is described in appendix R.3, was tested by 8 able-bodied children and a 5 year old with physical and cognitive disabilities by Soh and Demiris (2012). A subsequent study paired ARTY with a haptic device in order to learn the signals an expert driver would give to a wheelchair user (Soh and Demiris, 2014). This approach is complementary to the one we take in section 5.1 where we create a robotic tutor that evaluates users in terms of the overall path rather than trying to teach at the *input* level.

2.5 MUSIC GENERATION

Cicconet et al. (2013) developed a robotic system for human-robot collaborative percussion generation. Their focus is on understanding of social cues (in particular, visual cues) to anticipate the next action the user will perform. This approach is complementary to the one we will take with our musical companion in section 5.2, as we focus on predicting the intentions of the user based on the *structure* of the musical task.

Another robotic drummer was developed by Crick et al. (2006). In this case, the approach to synchronisation is made from a developmental point of view. Nico, a humanoid robot, learns to integrate its sensory inputs (visual, auditory and proprioceptive) to produce the appropriate drum-beats. This contrasts with our musical companion where all the information for Baxter (which we describe in appendix R.4) comes from the structure of the task and the tangible table itself (see appendix R.5).

Regarding automated music generation, already in 1986 Ebcioglu developed an expert system to generate harmony for music in the style of J.S. Bach's chorales.

More recently, there has been work in applying machine learning techniques to music generation. For instance, Tidemann and Demiris (2008) describes an architecture based on Echo State Networks to capture the *groove* of drummers. The authors define *groove* as variations in timing and musical pattern. We recognise the importance of these variations for the music produced to sound *natural* and our musical companion can indeed produce musical pattern variations, but not yet timing variations.

There are also approaches to music generation using Context-Free Grammars (Kitani and Koike, 2010; Quick and Hudak, 2013). In particular, Kitani and Koike (2010) describes a system to improvise a drum accompaniment which chooses the pattern to be played based on both the current context and previous history. Whilst our approach with the musical companion is similar in that we also derive the final action from a mixture of distributions, we explicitly consider a secondary (influencing) grammar to represent the actions of the other performer. Our approach has the advantage that it predicts the most likely next action of the collaborator rather than merely reacting to the observed actions.

2.6 HUMAN-ROBOT COLLABORATION AND ROBOTIC COMPANIONS

There has been research into the differences between a simulated robot and a physical one. Importantly, it has been shown that people rate interactions with an actual robot more positively than with one seen through a live video display (Bainbridge et al., 2011). Indeed, our own experiments will show that users prefer a real robot over a simulated one when the robot is acting as a wheelchair sidekick. This implies robots may be better companions than avatars.

Fong, Thorpe and Baur (2003) present the requirements for human-robot collaboration based on dialogue. In particular, they note the importance of robot adaptiveness. According to the authors "the robot has to be able to adapt to different operators and to adjust its behaviour as needed", thus highlighting the importance of user modelling in human-robot collaboration. Fong, Thorpe and Baur point out that any collaborative robot should be able to follow or ignore human advice depending on the circumstances. Our synchronised grammars framework (introduced in section 5.2) also provides the means to implement such a decision mechanism.

The work of Hoffman and Breazeal (2007) in human-robot collaboration for task assembly is relevant as well. The authors present a Markov process framework that can anticipate user actions. Their tests in a simulator with 27 subjects show that users much preferred a robot with anticipation capabilities. This finding is echoed in Shah et al. (2011) where experiments with an actual robot and 16 participants show that users spent 85% less time idling when the robot could anticipate their actions in a collaborative assembly task. Similar to both of these articles, our user modelling frameworks predict the most likely user action to decide what to do next through the use of SCFGs.

Schneider et al. (2012) evaluate different strategies to give feedback to users who are performing a difficult task. They found users appreciate more feedback relating to their performance in the task (eg. "your average answering time is too short") rather than on the generic feedback (eg. "you're halfway through it"). However, the authors did not investigate whether users preferred having a NAO humanoid robot giving them feedback to receiving no feedback at all.

In 2005, Dautenhahn et al. studied whether people would be willing to accept a robot companion in their homes. It was found that a large proportion of people were positive towards the idea. More recently Frennert et al. (2012) performed a similar study for old people in Sweden reaching analogous conclusions.

Examples of robotic companions are presented by Plaisant et al. (2000) and Leite et al. (2009). The former article describes a story-teller robot which children could program with their own movements. The latter, details a 5 week-long experiment with children and a social robot that plays chess and found that most participants regarded the robot more like an automaton at the end of the experiment. The authors attribute this partly to the lack of user adaptation on the robot. This finding highlights once more the importance of robotic companions adapting to their users.

There has been much research about robotic companions in Europe, with several projects funded by the European Framework Programme. Here we review three such projects: IROMEC, CompanionAble, and ALIZ-E[2].

IROMEC (Interactive Robotic Social Mediators as Companions) showcases the benefits of a robotic companion able to play with developmentally-impaired children (Ferrari et al., 2009; Marti and Giusti, 2010). Short-term experiments performed during therapy sessions with three children showed that the robot improved the playfulness of two of the three children (Klein et al., 2011).

CompanionAble, on the other hand, seeks to assist people in their homes and it is targeted for the elderly, particularly those who suffer from mild cognitive impairment

2 Part of research that led to this thesis was funded by the ALIZ-E project

(Gross et al., 2011). A related study by Nadrag et al. (2011) demonstrates how to develop a tele-operation control framework using force feedback. The authors report that force feedback may not have benefits for remote operators.

Thirdly, there is ALIZ-E which had the objective of developing a humanoid robot to accompany children with diabetes whilst they stay at the hospital. As Belpaeme et al. (2012) indicate, four activities were developed for the robot: a quiz game, a math game, human-robot interaction via a sandtray (a tangible table) and a dance game. As part of the project, a study carried out to find out exactly what activities should a companion robot for diabetic children perform (Baroni et al., 2014). The authors found that there were several areas where a robot could help the patient: *entertainment*, *self-management*, *knowledge increase*, *attention catching*, *self-confidence improvement* and *sensitive listening*. Of these, we paid close attention to *knowledge increase*, *attention catching* and *entertainment* and programmed our robotic companions with them in mind, for example the wheelchair sidekick in section 4.1 attracts the user attention to the obstacle, while the wheelchair tutor's main mission was to help users to improve their driving skills.

Furthermore, another related, user study found the importance of the robotic companion talking to the user whilst they are changing activities (Kruijff-Korbayová et al., 2014).

Ros et al. (2014) reports on a user study involving a robotic dancing tutor and 12 children in a hospital in Italy. The robotic tutor is able to evaluate the abilities of the children, similar to our wheelchair driving tutor (see section 5.1). However, their user model is specific to dancing and is derived from human-human interactions with processional dance teachers. The authors present 12 observations from the lessons and implement them as part of the tutor. Results show children mostly perceived the robot as a *friend* or a *sibling* rather than a *parent*, a *stranger*, a *neighbour*, a *classmate*, a *teacher* or a *relative*.

To the best of our knowledge, there is little published research in companions for robotic wheelchair users, making the wheelchair sidekick we present in sections 4.1 and 5.1 one of the first robots of its type.

2.6.1 Robots in clinical settings

One of the user studies in this thesis places the NAO humanoid robot (presented in appendix R.6) as a hospital companion. Consequently, we now review robotic companions in hospital and hospital-like settings.

Robots have been trialled with the aim of making children feel at ease in hospitals (Lu et al., 2011; Ros et al., 2014). As previously mentioned, Ros et al. is noteworthy as the

authors explore the use of a semi-autonomous robotic dancing tutor in a hospital in Italy. 12 children took part on the trials, though most of them were not actual hospital patients. It has been previously noted that dance is salient and easy to perceive (Michalowski et al., 2007). This makes it appropriate for both children and adults and gave us a strong reason for incorporating it into the hospital companion set of activities.

To our knowledge, ours is the first instance of a social companion with actual patients in an actual hospital. Most of the relevant literature is about socially assistive robots for older people outside the hospital. One such case is reported by Tapus et al. (2009). where a humanoid-like robot played musical games with 4 people with dementia over a period 6 months. Encouragingly, the authors found improvements in error rates and reaction times on the games as the study progressed.

Similarly, McColl and Nejat (2013) used a robot to keep 8 individuals, aged 82-93, company as they ate their meals in a care home. The authors noted a 87% compliance rate when the robot encouraged patients to eat.

PARO is a robotic soft toy in the shape of a baby seal that reacts to the user's presence and touch. PARO was developed with the aim of reducing social isolation amongst the older population. Robinson et al. (2013) demonstrated that PARO significantly reduced the patients' loneliness as measured by the UCLA Loneliness Scale. The trials were performed in a care home in New Zealand over 12 weeks with 40 residents. In a follow-up article, Robinson et al. (2015) showed that even if some participants were emotionally attached to PARO, they were fully aware it was a machine. Moreover, Takayanagi et al. (2014) established PARO was preferred by patients with mild/moderate dementia and severe dementia over a lion soft toy in a care home in Japan with patients talking and laughing more with PARO than with the soft toy.

It is important to highlight that NAO is a very different kind of robot than PARO, particularly regarding appearance: NAO is a humanoid and PARO is a baby seal. Further, NAO can talk, walk and has more degrees-of-freedom, which will impact user expectations (Goetz et al., 2003). Another difference between PARO and NAO is that our user study was conducted in an acute hospital setting, an unpredictable environment where issues such as infection control arise. Finally, NAO can clearly have more sophisticated interactions such as conversing or exercising.

2.6.2 User modelling

Many user modelling approaches in the robotics literature —including the ones we will present in chapter 5— consist of using a machine learning algorithm to model the task and extracting the user model from certain properties of the algorithm.

Huntemann et al. (2013) model a smart-power wheelchair driving task as a Dynamic Bayesian Networks. With this arrangement, the system proposes possible local trajectories which then are moderated by the learnt preferences of the user (using Gaussian Processes). Huntemann et al. tested their set-up with user with spastic quadriplegia symptoms and found that their framework does propose the correct local trajectories.

Another example of user modelling in human-robot collaboration is given by Nikolaidis et al. (2015). The scenario for this article is collaboratively painting a box with the applying the paint and the robot holding the box in different positions. In order to achieve this Nikolaidis et al. model different types of users as policies for a mixed-observability Markov decision process. The different types of users (policies) are learnt offline with an expectation maximisation clustering algorithm. During runtime, the framework infers the type of the current user and applies the corresponding policy to the mixed-observability Markov decision process. Experiments show that users considered a robot acting under this framework to be both more responsive and better at predicting their actions compared to manually controlling all the robot actions.

Demiris (2009) present a framework for user-modelling based on the HAMMER architecture. The user model in this framework is derived from the observed confidences of the models. Said user model may then be employed to determine which composite actions the user has not performed yet she can potentially perform. The framework is able to determine this set of actions, denoted as the Zone of Proximal Development (ZPD) due to its similarity with the concept from Vygotsky (1978), because the user model has performance measurements of the atomic actions that make up the composite action.

2.6.3 Robotic tutors and teachers

There are several instances of robotics tutors in the literature which we review in what follows. Nonetheless none of the articles we found applied the concepts developed by the ITSs field, in particular the separation between inner loop and outer loop.

One case of a robotic tutor is presented in Kanda et al. (2004). Two Robovie robots, 120cm tall humanoids with 15 degrees of freedom (dof), were installed during two weeks in a Japanese elementary school where 119 first-graders (5-6 years old) and 109

sixth-graders (aged 11 to 12) could interact with them during breaks. Robovie's speech recognition and synthesis language was English which meant participants had to practice the language in order to interact with the robot. Most children stopped interacting with the robot after a week, yet those who kept interacting with it saw their English scores rise in a controlled exam.

A similar study by Park et al. (2011) analysed the effect of positive and negative feedback in robotic teachers. In their experiments, NAO performed a short presentation about the Renaissance. The authors found that students normally preferred a human teacher, the reason for this may lie in the delay to reply as well as the lack of facial expressions. Moreover, it was also established that students were more susceptible to negative feedback coming from the robot, which is why in section 5.1 the tutor avoids giving negative feedback.

Kennedy et al. (2015) discuss the use of a NAO robot as a maths tutor for 45 children aged 7 and 8. In the study, conducted across 5 weeks in a primary school in the UK, NAO teaches children about prime numbers. Their results show that the robotic tutor helped to significantly increase the test scores. However, the authors found that if the robot was overly social, users would "pay attention to the social behaviour instead of the lesson content" and the learning benefits disappeared.

2.7 CONCLUSION

This chapter has reviewed the main influences in our work as well as competing approaches. We remark that current user modelling techniques are focused in trajectory-level prediction rather than at task level. Likewise, we did not find any examples in the literature of applying the principles of ITSs to robotic tutors.

For the rest of this thesis we will focus on learning the preferences and abilities of the user at task level and applying these user models to robotic companions.

3

ALGORITHMS AND SOFTWARE FOR USER MODELLING

Understanding the intentions and behaviours of users is a key requirement for Human-Robot Interaction (HRI). More precisely, robotic systems need to recognise the behaviours and intentions of users in order to build a model of the user and modify the interaction according to what is appropriate for each user.

In this chapter we present algorithms for user behaviour recognition and prediction, which are the basis of user modelling. Both algorithms may be used for *action recognition* and *next-step prediction* of the on-going activity; and both are *hierarchically organised* as well.

The first algorithm is Hierarchical Attentive Multiple-Models for Execution and Recognition (HAMMER), a bio-inspired cognitive architecture for behaviour recognition and execution. The second algorithm, Stochastic Context-Free Grammars (SCFGs), was first used in the field of Natural Language Processing but has found use in robotics too. We will formally describe both HAMMER and SCFGs.

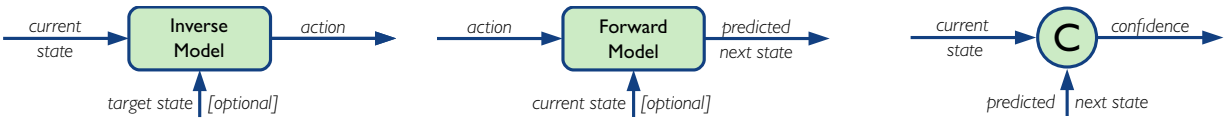
As part of the work for this thesis, we developed two computer libraries which implement HAMMER and SCFGs. We will discuss their main features, outline their Application Programming Interfaces (APIs) and provide the sample code for both approaches.

We will finish this chapter by implementing an HRI application with the two algorithms and examining the commonalities and differences between both approaches. For this thesis, we will regard SCFGs as a operationalisation of HAMMER's higher-level models.

Research from this chapter has been previously published in (Sarabia et al., 2015, 2011) and contributed to two other articles (Ognibene et al., 2012, 2013).

3.1 HAMMER OVERVIEW

Action recognition is one of the most important capabilities in humans and animals. The mirror neuron system, at least on primates, is hypothesised to use the same brain areas for executing actions as well as recognising those actions when executed by others. For an overview of the mirror neuron system see (Rizzolatti and Craighero, 2004). Subsequently, theories were developed stating that humans use internal inverse and forward models for motor control (Wolpert et al., 1998). HAMMER brought the concept of mir-



(a) Inverse Model (b) Forward Model (c) Confidence Function

Figure 3.1: HAMMER building blocks.

ror neurons to robotics by defining structures that can be used for both execution and recognition of actions (Demiris and Khadhouri, 2006).

HAMMER has been successfully used in many different contexts; for example, to recognise and imitate a human moving an object between two tables (Johnson and Demiris, 2004), to recognise compound and single actions of multiple robots (Martins and Demiris, 2010), to predict the intention of opponents in a real-time strategy game (Butler and Demiris, 2010) as well as to predict the intention of other drivers in a multi-agent traffic simulator (Sanderson and Pitt, 2011).

3.1.1 Building blocks

HAMMER has three building blocks: inverse models, forward models and the confidence function (cf. fig. 3.1). However, before describing these blocks we have to define *state of the world*. *State* is a set of variables, both external (environmental) and internal (proprioceptive) which are observable by the robot and that describe the *world* in which the robot is acting. We refer to the state variables as aspects to avoid confusion with programming variables.

An inverse model is defined by the following equation:

$$f_{IM}(\mathbf{s}_t, [\mathbf{s}^*_t]) = \mathbf{u}_t \tag{3.1}$$

An inverse model (fig. 3.1a) is a function which takes the state of the world ($\mathbf{s}_t$), and an optional explicit target state ($\mathbf{s}^*_t$) as inputs. It outputs the action signals ($\mathbf{u}_t$) to reach the target state. The target state may be implicitly coded in the model or passed as an external argument. An example of an inverse model would be a proportional-integral-derivative (PID) controller which takes as input the target state and outputs the commands for the plant to reach that state.

A forward model is likewise defined as:

$$f_{FM}(\mathbf{u}_t, [\mathbf{s}_t]) = \hat{\mathbf{s}}_{t+1} \tag{3.2}$$

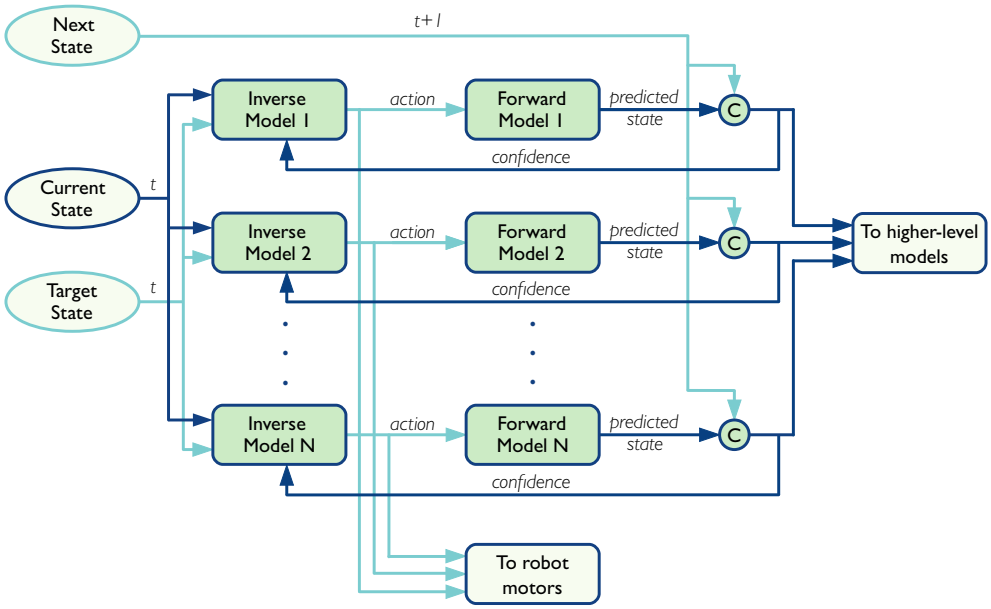


Figure 3.2: Diagrammatic representation of HAMMER. Each pair of inverse and forward model represents a hypothesis about the next state, which is then evaluated against the actual state to generate a confidence value.

The forward model (fig. 3.1b) is a function whose inputs are an action signal ($\mathbf{u}_t$) and, optionally, the current state ($\mathbf{s}_t$). It outputs the predicted state ($\hat{\mathbf{s}}_{t+1}$) the world would be in if the command was carried out. The term forward model has been used in the literature to represent many different concepts. For us, it is an output predictor, following the analysis by Karniel (2002). A forward kinematic chain is an example of a forward model where the motor rotations serve as input and the output is the position of the robot's end-effector.

Finally, the confidence function is characterised by the next equation:

$$f_{CF}\left(\mathbf{s}_{t+1}, \hat{\mathbf{s}}_{t+1}\right) = \Delta c \tag{3.3}$$

The confidence function (fig. 3.1c) thus assigns a score or confidence (Δc) depending on the current state ($\mathbf{s}_{t+1}$) and the predicted state ($\hat{\mathbf{s}}_{t+1}$). The confidence function fills a similar role to the reward function in reinforcement learning (Sutton and Barto, 1998) as it rewards the best models, ie. those which make the best predictions.

3.1.2 Multiple models for execution and recognition

Pairing together an inverse model and a forward model, we obtain a system that generates a hypothesis about the next state of the world:

$$f_{FM}(f_{IM}(\mathbf{s_t})) = \hat{\mathbf{s}}_{t+1} \tag{3.4}$$

By employing several inverse-forward pairs, normally executed in parallel, numerous hypotheses may be proposed. These hypotheses are evaluated at the next time-step by the confidence function. If we repeat this process iteratively, confidences for different behaviours can be observed over time. We can also use the confidence scores to select the inverse-forward pair that best fits the current and past observations (ie. the state). Figure 3.2 shows a diagram of this set-up.

A key feature of HAMMER is that neither the inverse nor the forward models nor the confidence function require the state to be centred on the robot. It can also be the state of the world around a demonstrator. Therefore, one can use both the robot's state and the user's state as the inverse model's input. If using the robot's state, the robot can *execute* the actions modelled by the inverse model by simply feeding the output actions of the inverse model ($\mathbf{u_t}$) to the motors of the robot.

Alternatively, if feeding the user's state to the inverse models, the robot can *recognise* actions, by using the forward models to predict what will the user do next, evaluating the predictions against the actual user behaviour with the confidence function and selecting the model with the highest confidence as the behaviour the user is performing. This mode of operation has been described as "placing the robot in the demonstrator's shoes" (Demiris, 2007; Johnson and Demiris, 2004).

3.1.3 Hierarchical and attentive

Besides being able to reuse the same structures to recognise and execute behaviours, HAMMER can be used to direct the attention of a robot. This characteristic was not used for this thesis, so we will not dwell on it. Suffice to say that, when there is competition for resources between the different inverse-forward pairs, the inverse models can list the resources they need and HAMMER may decide whether to allocate them depending on the inverse model confidence value (Demiris and Khadhouri, 2006; Ognibene et al., 2013).

The final ingredient in HAMMER is its hierarchical organisation. The confidence scores of inverse-forward pairs can be fed to to *higher-level* inverse-forward pairs whose mission is to reason about the behaviour of the *lower-level* models. As we will see in section 3.5.4, this could be achieved by creating an inverse model as a Finite State Machine (FSM) or even a SCFG.

Why is the hierarchical component of HAMMER important? It is because it allows to compose simple actions into more complex behaviours, in the same way that a dance is composed of multiple individual steps.

3.1.4 Key features

Now that we have seen how the HAMMER architecture is built, let us list what we consider to be its most important features:

BIO-INSPIRATION: HAMMER is inspired by the mirror-neuron system in primates. Further, it has been applied to developmental psychology to analyse the behaviour of infants (Demiris and Meltzoff, 2008).

INVERSE MODEL REUSE: It makes it possible to reuse inverse-models for execution of previously learnt actions and recognition of those actions by others (Johnson and Demiris, 2004).

RESOURCE ALLOCATION: It allows for resource allocation, such as attention in active vision scenarios (Ognibene et al., 2013).

ACTION RECOGNITION: It can be used for action recognition by running several concurrent inverse-forward pairs and selecting those with the highest confidence levels (Demiris and Khadhouri, 2006).

NEXT-STEP PREDICTION: Once the most likely inverse-forward pair has been selected, we can predict the most likely next step in the activity simply by reading the output of the forward model (Demiris, 2007).

HIERARCHICAL: It can be organised hierarchically to recognise complex behaviours. The confidence of low-level models serves input to the higher-level models (Johnson and Demiris, 2004).

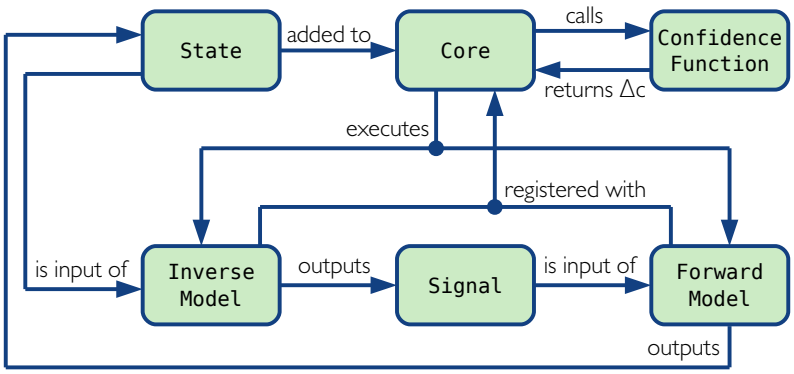


Figure 3.3: Relationships between the main C++ classes of the HAMMER middleware.

3.2 HAMMER MIDDLEWARE IMPLEMENTATION

We have coded a C++ library to instantiate HAMMER-like systems. The library, known as the HAMMER middleware is available at `imperial.ac.uk/PersonalRobotics`[1].

As we have seen in the previous section, several building blocks need to be specified to create a HAMMER-based architecture: state of the world at every time-step, inverse models, forward models, action signals to send commands between the inverse and the forward models, and the confidence function.

The HAMMER middleware provides C++ classes to help in the design of the above blocks, while at the same time making as few assumptions as possible about the intended application of the middleware. We now discuss the most important classes of the middleware (refer to fig. 3.3 for a diagram of these classes and how they are related).

The `State` class is akin to a polymorphic dictionary. Each variable in the container is an aspect of the world state. Aspects may be of any C++ type and must be labelled with a key (a `std::string`), which is used for storage and retrieval.

Although the class `Signals` is similar to the `State` class —it shares the same interface of a polymorphic dictionary— its semantics differ. `Signals` provide the means for inverse models to send action commands to the forward models. Further, they could be sent to the robot for execution of the commands contained within. Hence it is desirable —though not actually enforced— that any `Signals` instance contains low-level robot commands.

Inverse and forward models can be instantiated in the middleware with a free function. Otherwise, if the models need to store information, a C++ class which inherits from `InverseModel` or `ForwardModel` may be written. One important concept in our frame-

1 Also available from the project's development website: `http://miguelsdc.bitbucket.org/HAMMER`.

work is that of subscriptions. Every inverse-forward pair subscribes to a subset of the aspects of the world state. Users have to determine which aspects of the world state are needed for the operation of inverse models and subscribe to them. HAMMER will then filter out the irrelevant aspects for the inverse model. At the same time, subscription to an aspect entails the commitment to predict the value of that aspect at the next time-step. That is, the `State` generated by the forward model must contain all aspects to which the inverse model is subscribed to or an error will be raised.

HAMMER requires evaluation of the performance of the competing inverse-forward pairs. This is of course the role of the confidence function. The internals of the confidence function are left up to the end-developer to decide. The only requirement on the confidence function is to return a `double` representing the *change* in confidence. HAMMER calls the confidence function with both the predicted and observed aspects as inputs.

All interactions between the different components we have introduced thus far are controlled by an instance of `Core`. From the end-developer point of view, `Core` has a few crucial functions. All inverse and forward models must be registered with it. Additionally, it must also be fed the new `State` at every time-step. `Core` can additionally be used to obtain the current confidences of the inverse-forward pairs.

Code to instantiate sample inverse, forward models and the confidence function is presented in listing 3.1, whereas code to instantiate `Core` and register the inverse-forward pairs and the confidence function is shown in listing 3.2.

Implementing the inverse and forward models as well as the confidence function is left to the end-developer, the HAMMER middleware does not dictate how to do so. We will present an sample instantiation of these structures to recognise dance step in section 3.5.

Our framework has the ability to create hierarchies of inverse-forward pairs. A similar mechanism to that of aspect subscriptions is used to manage hierarchies. An inverse model simply needs to declare the list of lower level inverse-forward pairs that it wants to follow (known in the framework as dependencies). The middleware will then provide the confidence value at the previous time-step of those inverse-forward pairs declared as dependencies. Cyclic dependencies between inverse models are avoided since all dependencies are resolved at registration time; hence if an inverse model depends on another inverse model which has not yet been registered, `Core` will raise an error. Developers must be aware that hierarchies necessarily reduce concurrency as inverse models with dependencies cannot be executed at the same time as those without dependencies.

The approach taken for hierarchies on this middleware is not without disadvantages, as the end-developer is left to arrange the detection of potentially complex events. SCFGs,

```
//Bring HAMMER structures into scope
using namespace HAMMER;

Signals::KPtr sampleInverseModel(
    const State::KPtr& current, const State::KPtr& target){
    //Read data from state and target
    int currentValue = current->get<int>("aspectName");
    int targetValue = target->get<int>("aspectName");
    //Compute robot command based on the target and current values
    int command = targetValue - currentValue;
    //Create signals and put command(s) there
    Signals::Ptr result = Signals::make();
    result->put("robotCommand", command);
    return result;
}

State::KPtr sampleForwardModel( const Signals::KPtr& action){
    //Read command
    int command = action->get<int>("robotCommand");
    //Compute new state based on command
    int newState = command + 1;
    //Create state and put predictions there
    State::Ptr result = State::make();
    result->put("aspectName", newState);
    return result;
}

double sampleConfidenceFunction(const StateMap::KPtr& states){
    //Extract actual and predicted states
    AspectPair<int> pair = states->get<int>("aspectName")
    // Compute error
    int error = pair.actual - pair.predicted;
    // Compute (and return) confidence from error
    double confidenceDelta = 1.0/ (1.0 + std::exp(-error)) ;
    return confidenceDelta;
}

// Function to instantiate states (normally would query the real world)
State::KPtr newState(const State::KPtr& previous){
    static int count = 0;
    State::Ptr result = State::next(previous);
    result->put("aspectName", ++count);
    return result;
}
```

Listing 3.1: Sample instantiation of the HAMMER building blocks in C++.

```
int main (void){
    // Instantiate HAMMER's core
    Core::Ptr sampleCore = Core::make();

    //Declare the inverse-forward pair subscriptions
    StringVector subscriptions;
    subcriptions.push_back("aspectName");

    //Register inverse and forward models with core
    sampleCore->registerInverseForwardPair("nameOfPair", subscriptions,
        &sampleInverseModel,
        &sampleForwardModel );

    //Instantiate original state and register the confidence function
    State::Ptr sampleState = State::make();
    sampleState->setConfidenceFunction(&sampleConfidenceFunction);

    // Do one computation step
    sampleState = newState(sampleState);
    sampleCore->addState(sampleState);

    // Wait for inverse/forward models to finish
    sampleCore->wait();

    //Print confidences
    std::cout << "Confidences: " << sampleCore->getConfidenceMap() << "\n";
}
```

Listing 3.2: Sample `main()` function for HAMMER.

which we will introduce shortly, provide an intuitive way of expressing these same dependencies between models.

3.2.1 Active vision: an application of the HAMMER middleware

The HAMMER middleware has been used in other projects beside those presented in this thesis. Specifically, Ognibene et al. (2013) present a framework for action recognition in unknown environments which uses the HAMMER middleware.

The problem the article attempts to solve is that of directing attention —which is limited due to the field of view of the robot— in order to recognise a task —a demonstrator reaching for an object— when the robot is uncertain about the position of the hand and the position of the reachable objects. In this scenario, the robot has to deal with

three sources of uncertainty: uncertainty about the object the demonstrator is reaching for, uncertainty about the position of the objects and the demonstrator's hand and uncertainty introduced by performing saccades (since the movement of the robot itself is noisy).

This scenario is a good fit for the HAMMER architecture as the robot needs to execute multiple hypotheses. Indeed, the proposed framework instantiates inverse models which predict the position of the demonstrator's hand and reachable objects and executes them in parallel. These inverse models are implemented as Kalman filters, this way it is possible to model the uncertainties listed earlier. The models are updated as new observations are made available and the framework then system then allocates attention (that is, performs saccades) according to which model has higher confidence and can lower their entropy the most.

Results show that the proposed framework is both more accurate and faster at task recognition than systems without visual attention and other visual attention models.

3.3 STOCHASTIC CONTEXT-FREE GRAMMARS OVERVIEW

In the previous sections, we introduced an algorithm for execution and recognition as well as its software implementation. This section is dedicated to provide a summary of the theory behind another algorithm useful for user modelling, Stochastic Context-Free Grammars (SCFGs). SCFGs are ideal for representing structured tasks (Ivanov and Bobick, 2000), such as dividing a dance into several steps.

For this thesis, we make use of the parsing algorithm by Stolcke (1995), who extended Earley's top-down Context Free-Grammar parser (1970) to SCFGs. Note that this overview, unless otherwise indicated, has been adapted from Stolcke (1994). Let us start with the definition of a grammar:

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{S}, \mathcal{R}, \mathcal{P}) \tag{3.5}$$

where $\mathcal{N}$ is the set of non-terminals, $\mathcal{T}$ is the set of terminals, $\mathcal{S}$ is the starting non-terminal, $\mathcal{R}$ is the set of rules of the form $X \to \lambda$ with $X \in \mathcal{N}$ and $\lambda \in (\mathcal{N} \cup \mathcal{T})^*$, and $\mathcal{P}$ is the set of rule probabilities, that is:

$$\mathcal{P} = \bigcup_{\forall r \in \mathcal{R}} P(r) \tag{3.6}$$

The role of the parser consists in generating states of the following form:

$$i : \quad {}_{k}X \to \lambda.\mu \quad [\alpha, \gamma, \nu] \tag{3.7}$$

where i is the state-set indicator and denotes at which point this state was created, k denotes at which point this chain was first considered by the parser, X is a non-terminal, λ and μ are a combination of terminals and non-terminals (and both could be empty), the dot represents the next symbol to be scanned, α is the forward probability (i.e. the probability of the parser generating this rule), γ is the inner probability (that is, the probability of generating the current string starting at point k) and ν is the Viterbi probability (the probability of the sequence which best fits the grammar).

There are three functions to generate all the required states that the parser will execute iteratively: scan, complete and predict. Scan is in charge of incorporating inputs (terminals) into the parser, complete takes care of moving the dots of non-terminals whose rules are finished and predict adds states to expand every non-finished non-terminal.

We denote the state-set $\mathcal{SS}_i$ as the set of all states introduced in step i. In other words, every new state generated by scan, complete or predict at the i^{th} step will belong to $\mathcal{SS}_i$.

3.3.1 Scan

As mentioned before, scanning incorporates terminals into the parser. Scan only considers states from the previous step $\mathcal{SS}_{i-1}$, this way the input terminals order is respected by the parser.

Since scan only considers terminals, not all states from $\mathcal{SS}_{i-1}$ will be relevant. In fact, the only relevant states will have $\mu = x\rho$ with $x \in \mathcal{T}$ and $\rho \in (\mathcal{N} \cup \mathcal{T})^*$. We define this set of states as the candidate set:

$$\mathcal{CS}_i = \bigcup_{\forall x \in \mathcal{T}} \left(i : \ {}_{k}X \to \lambda.x\rho \ [\alpha, \gamma, \nu] \right) \tag{3.8}$$

For every state in $\mathcal{CS}_{i-1}$, scan performs the following operation:

$$i-1 : \quad {}_{k}X \to \lambda.x\rho \quad [\alpha, \gamma, \nu] \qquad \Longrightarrow \qquad i : \quad {}_{k}X \to \lambda x.\rho \quad [\alpha', \gamma', \nu'] \tag{3.9}$$

Note how the dot moves to reflect the parser has incorporated terminal x. Following Ivanov and Bobick (2000), we allow scan to accept probabilistic terminals by computing the new forward, inner and Viterbi probabilities as follows:

$$\alpha' = \alpha\, P(x)$$
$$\gamma' = \gamma\, P(x)$$
$$\nu' = \nu\, P(x)$$

where $P(x)$ is the observed probability of x. If $P(x) = 0$, then no new state is generated. Accepting probabilistic terminals allows the parser to deal with noisy inputs, since states for every possible interpretation of the terminal are generated and tracked.

3.3.2 Complete

The role of the complete operation is to check whether any state is finished. A finished state is of the form $i : {}_jY \to \rho.\,[\alpha, \gamma, \nu]$ with $Y \in \mathcal{N}$ and $\rho \in (\mathcal{N} \cup \mathcal{T})^*$. If a state has finished, we can consider its left-hand side non-terminal as accepted by the grammar and thus we must advance the dot everywhere where the parser was waiting to accept that non-terminal. This operation can, and generally does, add new states which can themselves complete other states.

Another complication appears if the grammar has rules of the form: $X \to Y$ with $X, Y \in \mathcal{N}$. These are rules are known as *unit rules* and can create infinite recursions. This can be easily observed if the rule $Y \to X$ was also part of the grammar.

To deal with these cases we define the probabilistic, reflexive, transitive unit rule relation $\mathbf{R_u}$ as a matrix of $|\mathcal{N}|$ by $|\mathcal{N}|$ where each entry $\{X, Y\}$ compensates for all recursive chains induced by unit rules from X to Y (including transitive ones). $\mathbf{R_u}$ is defined as:

$$\mathbf{R_u} = (\mathbf{I} - \mathbf{P_u})^{-1} \tag{3.10}$$

where $\mathbf{P_u}$ is the matrix whose entries $\{X, Y\}$ represent the total probability of all unit productions $X \to Y$.

With $\mathbf{R_u}$ in hand, we can define the complete operation as follows:

$$\left.\begin{array}{ll} i: & {}_jY \to \rho. \quad [\alpha, \gamma, \nu] \\ j: & {}_kX \to \lambda.Z\mu \quad [\alpha', \gamma', \nu'] \end{array}\right\} \implies i: \quad {}_kX \to \lambda Z.\mu \quad [\alpha'', \gamma'', \nu''] \tag{3.11}$$

for all $Y, Z \in \mathcal{N}$ such that $\mathbf{R_u}\{Z, Y\} \neq 0$ and $|\rho| > 1$, that is the finished rule is not a unit rule. The importance of the k element becomes now clear, as it is what allows the

algorithm to determine whether a state may be completed by another finished state. The forward and inner probabilities are updated as follows:

$$\alpha'' = \sum_{j:\,{}_kX\to\lambda.Z\mu\,[\alpha',\gamma',\nu']} \gamma\,\alpha'\,\mathbf{R}_u\{Z,Y\}$$
$$\gamma'' = \sum_{j:\,{}_kX\to\lambda.Z\mu\,[\alpha',\gamma',\nu']} \gamma\,\gamma'\,\mathbf{R}_u\{Z,Y\}$$
$$\nu'' = \max_{j:\,{}_kX\to\lambda.Z\mu\,[\alpha',\gamma',\nu']} \left(\nu\,\nu'\,\mathbf{R}_u\{Z,Y\}\right)$$

3.3.3 Predict

Predict is in charge of expanding states that are currently waiting on a non-terminal, ie. $i :\,{}_kX \to \lambda.Z\mu\,[\alpha,\gamma,\nu]$. Roughly speaking, predict simply adds a new state for each one of the rules where Z appears on the left-hand side ($Z \in \mathcal{N}$). We repeat this operation for every Z in the state-set waiting on a non-terminal. As with completion, the newly added states may well cause predict to spawn new states.

Similarly to complete, there is a risk of infinite recursion if the grammar contains prediction loops, for example $X \to Y\mu$ and $Y \to X\rho$. These rules are known as *left-corner rules* and are the form $X \to Y\lambda$.

We compensate for the left-corner rules with the probabilistic, reflexive, transitive left-corner relation $\mathbf{R}_l$. Every entry $\{X,Y\}$ represents the transition probability for all paths from X to Y according to the grammar rules. We compute $\mathbf{R}_l$ as follows:

$$\mathbf{R}_l = (\mathbf{I} - \mathbf{P}_l)^{-1} \tag{3.12}$$

$\mathbf{P}_l$, in the equation above, is a matrix whose entries $\{X,Y\}$ contain the sum of transition probabilities of all rules $X \to Y\lambda$, with $X,Y \in \mathcal{N}$ and $\lambda \in (\mathcal{N} \cup \mathcal{T})^*$.

We are now in a position to define the predict operation as follows:

$$i :\quad {}_kX \to \lambda.Z\mu \quad [\alpha,\gamma,\nu] \qquad \Longrightarrow \qquad i :\quad {}_iY \to .\rho \quad [\alpha',\gamma',\nu'] \tag{3.13}$$

NON-TERMINALS & AXIOM: $\mathcal{N}[\mathcal{S}]$	{S} [S]
TERMINALS: $\mathcal{T}$	{a,b}
STOCHASTIC RULES: $\mathcal{R}$ [$\mathcal{P}$]	{S→a S b [0.50], S→a b [0.50]}
INPUTS	a [0.8] b [0.2] a [0.5] b [0.5] a [0.5] b [0.5] a [0.2] b [0.8]
VITERBI PARSE	$4:_0 \to$ S. [0.04, 0.04, 0.04] ↘ $4:_0$ S→aSb. [0.04, 0.04, 0.04] ↘ $3:_1$ S→ab.[0.05, 0.13, 0.13]
PARSED TERMINALS	a a b b

Table 3.1: Example of a Viterbi parse.

for all rules $Y \to \rho$ with $\mathbf{R}_l\{Z, Y\} \neq 0$ and

$$\alpha' = \sum_{i:_k X \to \lambda . Z \mu \, [\alpha,\gamma,\nu]} \alpha \, \mathbf{R}_l\{Z, Y\}$$

$$\gamma' = P(Y \to \rho)$$

$$\nu' = P(Y \to \rho)$$

3.3.4 Viterbi parse

The Viterbi parse is the set of states that yields the maximum parsing probability for a given sequence of input terminals. Table 3.1 shows a sample Viterbi parse for a simple grammar with uncertain inputs. Notice how the second and third input are assigned to a and b respectively as this is the assignment that best fits the structure of the grammar.

In order to obtain the Viterbi parse we track the Viterbi probability as shown in eqs. (3.9), (3.11) and (3.13).

Additionally, we have to annotate the states with their *predecessor*. For scan and predict there is only one *predecessor* —only one state is necessary to scan or predict a new state. For complete on the other hand, there are two *predecessors*.

Once the final state has been found (see section 3.3.6) we build a parse tree by following the predecessors, and branching in *completed* states. Terminals found in the parse tree belong to the Viterbi parse.

Throughout this thesis we will use the Viterbi probability of the final state, as a measure of how well a set of inputs follows the structure determined by the stochastic grammar. The Viterbi probability depends on the length of the input, which makes comparisons across sequences of different length impossible. We compensate for this by following Lee et al. (2013) and defining the *scaled Viterbi probability*:

$$\bar{\nu} = \nu^{1/l} \tag{3.14}$$

where $\bar{\nu}$ is the scaled Viterbi probability, ν is the Viterbi probability of the final step, and l is the input sequence length.

Unless otherwise specified, when we mentioned Viterbi probability in this thesis we are referring to the *scaled Viterbi probability* ($\bar{\nu}$).

3.3.5 Prediction of next input

The Stolcke-Earley parsing algorithm allows us to easily predict the likelihood of the next input terminals. This formulation is based on our previous work (Sarabia et al., 2015).

Recall the definition of the candidate set, $\mathcal{CS}_i$ as the set of states which can be scanned (eq. 3.8). We further define $\mathcal{CS}_{i,s}$ as a subset of the candidate set where the next symbol to be read is terminal s, in other words:

$$\mathcal{CS}_{i,s} = \bigcup \left(i : {}_k X \rightarrow \lambda . s \rho \ [\alpha, \gamma, \nu] \right) \tag{3.15}$$

At this point, it is worth reiterating that α —the forward probability— represents the probability of the grammar generating the sequence up to the dot. Consequently, adding all the α from all states which accept s as their next terminal yields the expectation the parser has of terminal s being the next input. That is:

$$P(s_{i+1}) = \frac{\sum_{i: {}_k X \rightarrow \lambda . s \rho \ [\alpha, \gamma, \nu] \in \mathcal{CS}_{i,s}} \alpha}{\sum_{i: {}_k X \rightarrow \lambda . x \rho \ [\alpha, \gamma, \nu] \in \mathcal{CS}_i} \alpha} \tag{3.16}$$

gives us the expected probability of encountering terminal s at the next *scan* step, $P(s_{i+1})$. If we compute this probability for all terminals, we find the expected probability distribution across all terminals of the grammar. We denote the terminal probability distribution as $\mathbf{P}(\mathcal{T}_i)$.

3.3.6 Initialisation, execution and termination

Now that we have described all the operations in the Stolcke-Earley algorithm, it only remains to explain how all the functions work together. First, the initial state is generated: $0 : \ _0 \rightarrow .\mathcal{S} \quad [1.0, 1.0, 1.0]$.

No input has been received yet, therefore scan and complete will not generate any new states. On the other hand, predict needs to be called at this point to generate states that belong to the first candidate set $\mathcal{CS}_0$. After executing predict, the initial state-set, $\mathcal{SS}_0$, has been generated and we can already perform next input prediction (cf. section 3.3.5) to obtain the initial terminal probability distribution, ie. the probabilities of each terminal being the first input to the grammar.

Next, for every new input we perform scan, complete and predict —in that order— to process the incoming data.

Finally, whenever we are requested to check the Viterbi parse we look for a state of the form: $l : \ _0 \rightarrow \mathcal{S}.[\alpha, \gamma, \nu]$, where l is the input length. From there we backtrack via the states with maximum probability, which we annotated during parsing for this purpose, thus building the parse tree. If the final state is not found, that means the parser has not yet recognised the input sequence as valid.

Another error may occur if the candidate set, $\mathcal{CS}_i$, is empty at any point. This means no new states will be generated by scan. This error usually happens when all parsing branches have been exhausted or if the forward and inner probabilities underflow, a common problem when using 32-bit floats.

3.3.7 Parsing complexity

An in-depth analysis of the parsing complexity of the Stolcke-Earley algorithm is beyond the scope of this overview of SCFGs. We refer to the interested reader to Chapter 6 of Stolcke's thesis (1994) for a detailed complexity analysis. Instead, we will only briefly state the results of the analysis. The algorithm takes $O(l^3)$ time and $O(l^2)$ space with l being the input length of the grammar. Similarly the complexity scales in cubic time with the number of non-terminals: $O(|\mathcal{N}|^3)$.

Predicting the next input is linear with the number of states in the state-set $\mathcal{SS}_i$ and therefore grows quadratically with the input length.

3.4 THE SARTPARSER OPEN-SOURCE LIBRARY

As part of the work for this thesis we developed an open-source SCFG parser which can be used for any project, the SARTParser library. The source-code and documentation can be found online at `imperial.ac.uk/PersonalRobotics`[2].

The SARTParser library was used for each application we will present in chapter 5. This section highlights the work done to update and speed-up the library as well as an usage example of the library.

3.4.1 From offline executable to online library

The original SARTParser program was written by Yuri Ivanov in 1997 following the Stolcke-Earley algorithm we described in section 3.3. KyuHwa Lee then performed slight adjustments to the code. As part of our modernisation effort on the source code we contacted the original authors which agreed the library should be published under the open-source MIT licence.

The main issue with the original SARTParser code was that it was conceived as a single executable, where the inputs and outputs would be in text form. Although this approach is perfectly adequate for offline applications, it is problematic when attempting to run the parser in real-time as it requires encoding the data in a text-file, executing a remote program and parsing the output data. Diagnosis of errors in this way is even more difficult.

Another obstacle on the way of using SARTParser in an online fashion was that the codebase was quite old: it was written before C++'s standard template library —now a fixture of most C++ programs— was standardised. As such it used language idioms and structures that are difficult for the compiler to optimise, resulting in slower —and less readable— code.

We have made 169 revisions to the code thus far. We now list the changes between the original and current versions of SARTParser:

- Replacing `#define` macros by C++ constants (this helps the compiler optimise the binary).

2 Also available from the project's development website: `http://miguelsdc.bitbucket.org/SARTParser`.

- Change build system to CMake. CMake is a modern and popular build system for C++ code which Robot Operating System (ROS) uses, thus simplifying the interfacing between ROS code and SARTParser.
- Increase type-safety by replacing custom array class by `std::vector`s, old-style strings (`char*`) by C++ strings (`std::string`), and using C++ enums instead of `int`s.
- Speeding up the code by making methods const-correct and inlining functions that the profiler identified as slow.
- Added methods to obtain the next terminal probability distribution, $\mathbf{P}(\mathcal{T})$, following the algorithm in section 3.3.5.
- Create a new API for the library, with few and distinguishable methods. The aim was to make the library easy-to-use even without a deep-knowledge of SCFGs and without having to generate and parse text-files.
- Create Python bindings for the library, which allows for quick prototyping of SCFG-based applications.

As a result of these changes the SARTParser is now faster. To measure the speed-up, we benchmarked both the original and current versions of the parser. The grammar we applied to the benchmark had 9 terminals 10 non-terminals and 35 rules. We parsed sequences with input length varying from 0 to 121, repeated the benchmarks 10 times and discarded all the output of the parser[3]. The benchmark was carried out on an Intel Core i7 machine with 16GB of RAM. The results are shown in fig. 3.4. Interestingly, the current version of the parser took an average 12.103s (σ =0.087s) to parse the longest sequence compared with 121.544s (σ =1.655s) for the original version.

The complexity of the grammar is obviously unchanged by the source code optimisations, and as expected both curves in fig. 3.4 are cubic. However, the speed-up is still significant since it lets us make use of the SARTParser in real-time robotics scenarios with much longer sequences. For example, considering the data we collected from the benchmarks, the original version can only process an input sequence length of 49 steps before breaking real-time constraints, whereas the current version can process a length of 112 steps before breaking the real-time constraints. By breaking the real-time constraints we mean that a scan-complete-predict cycle takes more than 290ms —this is

3 By redirecting the parser output to `/dev/null`.

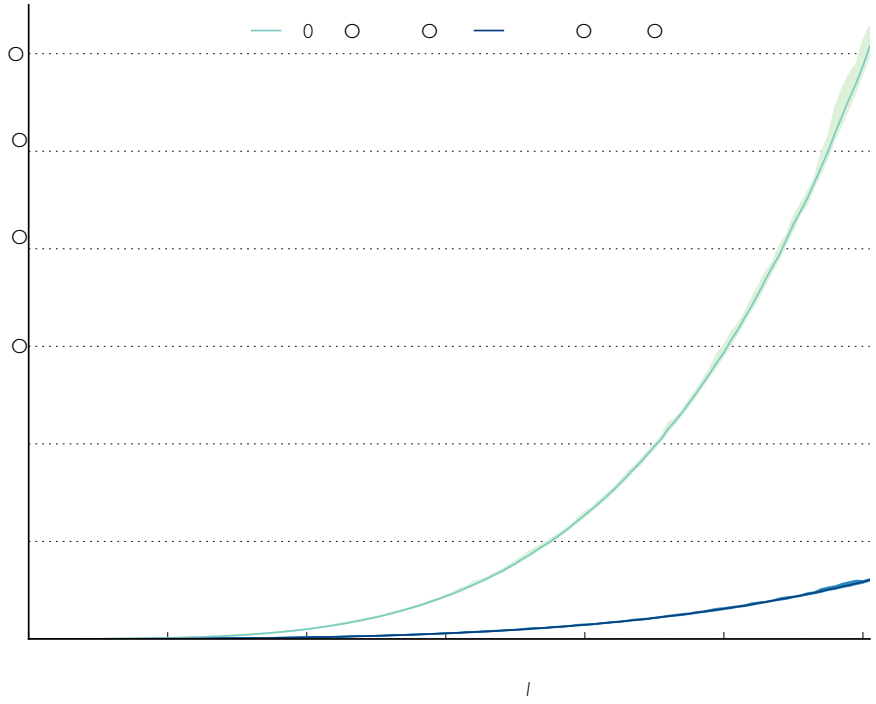


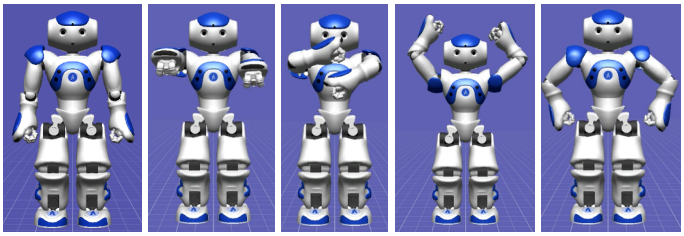
Figure 3.4: Parsing speed comparison between the original and the current version of SARTParser. Lines shown are averages across 10 rounds. Bounds shown in light green for the original version and light blue for the current version. Both lines are cubic, in line with the performance analysis in section 3.3.7.

the median human reaction time according to Taberner (1980). In summary, the speed-ups achieved by modernising SARTParser are important as they allow us to detect much more complex sequences.

3.4.2 SARTParser API overview

In this section we present a brief overview of the newly designed API for the SARTParser library. Though here we will focus on the native C++, the Python API is very similar — sometimes even, exactly the same.

There are two key classes in the library: `CFGrammar` and `SParser`. The first one, represents the grammar itself and as such its methods include everything needed to specify a SCFG. The methods `addTerminal()` and `addNonTerminal()` are used to determine $\mathcal{T}$ and $\mathcal{N}$ respectively. Similarly, `addAxiom()` sets the grammar axiom $\mathcal{S}$ and `addRule()` adds a rule in $\mathcal{R}$. There is a function, `loadGrammar()`, to create a grammar from a text stream.



(a) Recognised poses shown in the NAO robot. From left to right: *resting*, *extended*, *crossed*, *to-head*, *to-hip*.

(b) NAO imitating the recognised movements of a user.

Figure 3.5: Dancing NAO sample activity implemented with HAMMER and SCFGs.

The `SParser` class represents the stochastic parser. As with `CFGrammar`, it has a few straightforward methods. `parse()` performs a full parsing step (scan, complete and predict) whereas `getViterbiParse()` generates the Viterbi Parse tree and the sequence of terminals it contains (as in table 3.1). Finally, `getPrediction()` returns the expected terminal probability distribution, $\mathbf{P}(\mathcal{T})$, for the next step.

There are a few other support classes and methods, but we will omit them as they do not showcase any extra functionality. An example of a full parse on a simple grammar is shown in listing 3.3.

The parser runs in a single thread, as paralellising the Stolcke-Earley algorithm is hard; particularly complete which tends to generate many states recursively. Note that appendix P discusses a possible parallelisation of this library for Graphics Processing Units and expands on why complete is difficult to parallelise.

The parser methods are not thread-safe, in the sense that not more than one thread may call them concurrently. Nevertheless, the library does not have any static state and thus several parsers may be run in parallel.

3.5 DANCING NAO WITH HAMMER AND SCFGS: A CASE STUDY

So far in the chapter we have described HAMMER and SCFGs and introduced software libraries to operationalise both algorithms. For the remainder of the chapter we will focus on the features of HAMMER and SCFGs that are most useful to user modelling and present an HRI application that demonstrates these features.

From section 3.1.4, we know that the main features of HAMMER are: *bio-inspiration*, *inverse model resuse*, *resource allocation*, *next-step prediction* and *hierarchical organisation*. But which ones are most useful for user modelling?

```cpp
#include <iostream>
#include <fstream>
#include <SARTParser/All.h>
int main()
{
    //Construct grammar
    sartparser::CFGrammar g;
    //Add terminals
    g.addTerminal("a");
    g.addTerminal("b");
    //Add non-terminals
    g.addNonTerminal("S");
    //Add axiom
    g.addAxiom("S");
    //Add production rules
    sartparser::Rule firstRule;
    firstRule.lhs = "S";
    firstRule.rhs.push_back("a");
    firstRule.rhs.push_back("b");
    firstRule.probability = 0.5;
    g.addRule(firstRule); //Adding S -> a b [0.5]
    sartparser::Rule secondRule;
    secondRule.lhs = "S";
    secondRule.rhs.push_back("a");
    secondRule.rhs.push_back("S");
    secondRule.rhs.push_back("b");
    secondRule.probability = 0.5;
    g.addRule(secondRule); //Adding S -> a S b [0.5]

    //Create parser
    sartparser::SParser p(g);
    //Parse one step
    sartparser::PInput firstStep;
    firstStep.push_back( sartparser::PTerminal("a", 0.8) );
    firstStep.push_back( sartparser::PTerminal("b", 0.2) );
    p.parse(firstStep); //Parsing a[0.8] b[0.2]
    //Parse another step
    sartparser::PInput secondStep;
    secondStep.push_back( sartparser::PTerminal("a", 0.3) );
    secondStep.push_back( sartparser::PTerminal("b", 0.7) );
    p.parse(secondStep); //Parsing a[0.3] b[0.7]

    //Get viterbi parse results
    std::cout << p.getViterbiParse() << "\n";
    return 0;
}
```

Listing 3.3: SARTParser C++ usage example.

Demiris (2009) presents a framework for lifelong assistive robotics which relies on HAMMER performing user modelling. The proposed system needs to: *recognise* which *action* is the user currently performing, *predict* the user's *next step* and verify that the predicted step falls within the current abilities of the user. The last operation is achieved by exploiting the *hierarchical* nature of HAMMER, as it decomposes the action in lower-level models and checks the user's past performance with those models. Thus we consider the key features of HAMMER for user modelling to be: *action recognition*, *next-step prediction* and *hierarchical organisation*.

These three key features are also applicable to SCFGs. The latter two features do not require much justification: *next-step prediction* was already introduced in this chapter (section 3.3.5) and stochastic grammars are *hierarchical* by nature.

Regarding *action recognition*, remark that parsing states may be interpreted as a hypotheses about the input terminals —obtained by observing the state of the world. The role of the parser is thus to generate several competing hypothesis to explain the input. Additionally, several parsers may be concurrently executed to provide further hypotheses about the state of the world. Whether one or many parsers are executed, the parsing state with the highest likelihood can be recognised as the demonstrated task. This process is clearly analogous to how multiple inverse-forward pairs are hypotheses which compete to best explain the observed state of the world. Thus *action recognition* with SCFGs works in a very similar manner to HAMMER's approach.

To further illustrate the similarities between HAMMER and SCFGs, we dedicate the rest of this section to present a system for action recognition with both HAMMER and SCFGs. It will also serve as an example of how to use both of our libraries, the HAMMER middleware and SARTParser, in a HRI application.

We chose dancing as the sample activity to perform with the HAMMER middleware and the SARTParser library. To be precise, the task at hand is to recognise the arm movements of the well known song *Macarena*. Both implementations have to differentiate between ten arm positions, five for each arm. The poses recognised are: *resting*, *extended*, *crossed*, *to-head* and *to-hip* (see fig. 3.5a for a depiction of these poses). The robot must be able to recognise these actions from the user and act accordingly (fig. 3.5b).

We now describe the different parts of each system: the state representation which is common to both HAMMER and the SCFGs parser; the confidence function and inverse and forward models for HAMMER as well as their counterparts, the detectors; and finally the Finite State Machine (FSM) for HAMMER and the grammar for the SCFGs-based system.

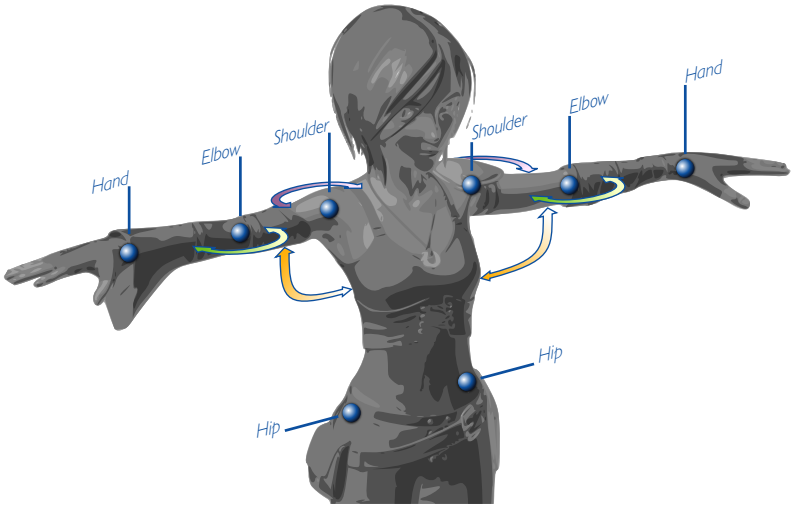


Figure 3.6: State representation for dancing NAO activity[6]. In *orange*: angle between *shoulder-shoulder* and *shoulder-hip* vectors. In *purple*: angle between *shoulder-elbow* and *shoulder-shoulder*. In *green*: angle between *elbow-hand* and *elbow-shoulder*.

3.5.1 State representation and acquisition

In order to avoid the correspondence problem —that is, the translation of actions across dissimilar embodiments (Alissandrakis et al., 2002)— we use an abstract state representation that can be easily calculated for both humans and NAO. This representation consists of six angles, three for each arm; namely the angle between the elbow and the shoulder (purple arrows in fig. 3.6), the angle between the shoulder and the hip (orange arrows in fig. 3.6), and the elbow angle (green arrows in fig. 3.6). This state representation clearly does not cover the whole space of arm positions in a human or a robot, however it does suffice to distinguish the end positions previously listed.

To generate the state representations, the Kinect depth sensor in conjunction with OpenNI[4] and NITE[5] are used. The Kinect sensor and associated software libraries were described in-depth in appendix R.7. This setup allows us to easily obtain a 15 point skeleton and apply geometric transformations to obtain the angles that make up the state representation. Once the angles are computed they are fed to HAMMER or to the SCFGs low-level detectors. This state representation is all the information about the user both of the dancing NAO implementations will receive.

4 OpenNI is a set of open source programmer interfaces for natural interaction devices such as the Kinect sensor.

5 NITE is a middleware built on top of OpenNI to generate skeleton and track users.

6 Character model is copyrighted by the Blender Foundation and licensed under the CC-BY-3.0 licence.

3.5.2 HAMMER's confidence function

HAMMER does not place any restrictions on the form of the confidence function. For convenience, we introduced two intermediate expressions to calculate the change in confidence: the error and the negative reward. The first one, the error, is computed as follows:

$$e_a = |s_a - \hat{s}_a| \bmod 2\pi \tag{3.17}$$

where $a \in 1 \ldots 6$ represents each of the six angles that represent the state, s_a is the observed value of angle a, and $\hat{s}_a$ is the predicted value. We further define the negative reward r_a in terms of the error:

$$r_a = \begin{cases} \dfrac{e_a}{\theta_a} & \text{if } e_a \leqslant \theta_a \\ 1 & \text{if } e_a > \theta_a \end{cases} \tag{3.18}$$

where θ_a are the angle specific thresholds, above which the negative reward will be maximal. These thresholds were fine-tuned by experimentation. Different values for different aspects be must allowed as the precision of the Kinect sensor varies for different parts of the body. For reference we used $\theta_a = \pi/2$ rad when a represented any of the elbow angles (green arrows in fig. 3.6), and $\theta_a = 0.6$ rad for all the other angles.

Finally, we define the confidence as,

$$\Delta c = \prod_{a=1}^{N} (1 - r_a) \tag{3.19}$$

Δc is the change of confidence for this step, and N is the total number of aspects of the state representation, ie. six. Multiplication is used here as it enforces that all the predicted angles are similar to those obtained from the user —if an angle differs too much its reward will be close to zero, thus bringing the total confidence towards zero.

Since Δc cannot be negative in eq. (3.19), the confidences are always rising. This means that at some point all confidences must be reset to allow for new behaviours to be detected. We found that periodically resetting all the models (inverse-forward pairs) produces good detection results. For our experiments we used a period of a second. This interval is sufficiently small to detect all movements by the user, yet large enough to average out any sensing error.

3.5.3 HAMMER's inverse and forward models

We instantiate an inverse model for every arm position to be detected (*resting*, *extended*, *crossed*, *to-head*, *to-hip*). For this particular application, every inverse model sends the desired end-effector position to the forward model, independently of the current world state. This is a simplification, as it assumes the robot will reach the end state in just one step; it is however sufficient for this applications as the 10 poses are easily differentiable. The output `Signals` of these inverse models contains the NAO motor commands required to reach those positions, which were recorded using kinaesthetics.

The forward models were implemented by applying forward kinematics and computing the angles that make up the state representation. These angles are calculated with the same function that is applied to the Kinect data input to generate the state representation, as in eqs. (3.17) to (3.19).

3.5.4 HAMMER's finite state machine for full dance detection

By taking advantage of the HAMMER middleware's dependencies feature (section 3.1), we were able to build a hierarchy that would detect the full *Macarena* dance using a new inverse model with dependencies to the 10 inverse-forward pairs introduced earlier. This higher-level inverse model kept track of the total confidence of every inverse-forward pair and detected which ones were being executed by the user. A Finite State Machine (FSM), whose states were the current position of the left and right arms, was coded and incorporated into the high-level inverse model. The only valid FSM transitions were those of the dance. This technique allows the robot to detect when has the user performed the full choreography and not just a few steps.

3.5.5 SCFG's low-level detectors

Thus far, we have described all the components necessary to operationalise a simple HAMMER architecture to recognise the steps of the *Macarena* and have them imitated by NAO. Now we describe an alternative, though similar, operationalisation based on SCFGs.

We do not use SCFGs to recognise low-level actions, as they are better suited for recognition of sequences. And although we could have used HAMMER to detect low-level actions, and then replace the FSM by a more powerful SCFG, this is not the approach we follow for reasons that will become clear at the end of this chapter.

Instead, we simplify the recognition of low-level actions by not splitting the models in inverse-forward pairs.

We denominate as *detectors* the structures that observe the state of the world (as defined in section 3.5.1) and output grammar terminals —with their associated probabilities. These detectors are effectively equivalent to the combination of an inverse-forward pairs and the confidence function already described in sections 3.5.2 and 3.5.3.

In our sample application, 10 low-level detectors were instantiated, one for each of the poses to be recognised (fig. 3.5a). Every time a new state is received, its angles (fig. 3.6) are compared to the internal angles computed from the pose that each detector is looking for. A score is then computed using eqs. (3.17) to (3.19) and stored in each detector.

These low-level detectors are reset every second. Subsequently, the detectors which had accumulated the highest score are sent as terminals to the SCFG parser.

3.5.6 Dance stochastic grammar

We manually designed a stochastic grammar to represent the *Macarena* dance[7]. The terminals of this grammar are made up of the combination of left arm and right arm poses (fig. 3.5a). The tuple (*resting*, *extended*) is an example of such terminals. The first element of the tuple represents the left arm position and the second, the right arm position. Since there are five poses for each arm, a total of 25 terminals were added to the grammar.

We designed the grammar so that the parser would cope with the following situations: parsing of spurious terminals, assigning partial credit for poses close to the expected one, iteration over the whole sequence and iteration over one step of the sequence. Let us examine how this was achieved.

The axiom of the grammar, S expands to five non-terminals, $S_0 \dots S_4$ and each of these expands to one of the terminals that represent the main steps of the dance. That is, S_0 expands to (*resting*, *resting*), S_1 to (*extended*, *extended*) and so on until S_4 which expands to (*to-hip*, *to-hip*).

A skip non-terminal was added to the grammar; skip non-terminals expand to each and every terminal with equal probability. For example, K is a skip non-terminal if the following rules are part of the grammar $\{K \to t_1, K \to t_2, \dots K \to t_{|\mathcal{T}|}\}$ with each rule having probability $|\mathcal{T}|^{-1}$ ($\forall t_1 \dots t_{|\mathcal{T}|} \in \mathcal{T}$, with $\mathcal{T}$ being the set of all terminals). This way, any non-terminal that expands to X may be skipped as it will accept all non-terminals in

7 Grammars may be learnt by demonstration following Lee et al.'s method (2013). However, since we are only considering 5 steps from the *Macarena* dance, manually designing the grammar allows us to explore the techniques for stochastic grammar specification

the grammar, albeit with a low probability. In the *Macarena* grammar, all non-terminals representing the dance steps, expand to the skip non-terminal ($S_0 \to K$, $S_1 \to K$, ... $S_4 \to K$).

In order to give partial credit for poses where one of arms was in the expected place, we create five non-terminals —one per pose— which expand to each terminal which has one arm in common with the expected pose. This is essentially a variation of the skip technique we have just described. Accordingly, for the first step (*resting*, *resting*), the partial credit non-terminal expands to (*resting*, *extended*), (*resting*, *to-hip*), ..., (*to-hip*, *resting*). There are 8 terminals with one common arm for every pose, and so each of the previous rules has a probability of 0.125. Each of the non-terminals representing a dance step, $S_0 \ldots S_4$, expands to a partial credit non-terminal.

We also wanted to allow for repetition of the whole dance as well as individual steps. To allow parsing of a non-terminal repeatedly, we just need to add a rule like this $X \to X\,X$. Such a rule ensures that the parser is always ready to accept another iteration of X. We added this style of rule for the grammar's axiom ($S \to S\,S$) and consequently, the whole dance may be parsed multiple times. We further add repetition rules for all the non-terminals representing the dance steps ($S_0 \to S_0\,S_0$, ... $S_4 \to S_4\,S_4$); therefore, the user may stay at the same pose for several cycles.

To recapitulate, the axiom also expands to the sequence of *Macarena* dance steps: $S \to S_0\,S_1\,S_2\,S_3\,S_4$. As we have seen each one of these step non-terminals expands to: the skip non-terminal, a partial credit non-terminal, themselves and the terminal which best represents theirs step. The grammar is not sensitive to the probabilities assigned to these rules; it simply requires that the right action is more likely than the wrong one, otherwise prediction will not work. Thus, every rule is assigned an equal probability of 0.25.

In summary, putting together detectors and a specially crafted SCFG allows us to create an architecture which is functionally similar to HAMMER and FSM to detect movements and imitate them with NAO. Both implementations have virtually identical functionality, but the SCFG-one is more robust to uncertainty.

Using SCFGs has three advantages over FSMs. Firstly, we can robustly predict the next step of the dance. Secondly, we can deal with spurious input terminals. Finally, we may use the Viterbi probability as a measure of how well did the user perform the dance.

3.6 CONCLUSION

The previous section justifies why at the beginning of the chapter we refer to HAMMER and SCFGs as the basis for user modelling. Using either algorithm we can detect actions from the user and adapt the interaction accordingly. The way we built the dancing NAO applications with HAMMER and SCFGs is extremely similar. The main difference being that HAMMER has distinct inverse forward models and confidence function whereas the detectors we use with SCFGs essentially group those three functions into a single module. As we have indicated earlier, it is possible to replace the detectors with HAMMER, as it is equally possible to create a high-level inverse model based on a SCFG parser. Indeed, we consider SCFGs as an alternative implementation of HAMMER's higher-level models.

For the rest of the thesis, however, we will focus on using both detectors and SCFGs, as we did with the SCFG implementation of the dancing NAO. The reason for this is that detectors do not split inverse and forward models; this split, whilst allowing for *reuse of the inverse model* to drive the actions of the robot, makes it more complicated to operationalise HAMMER in situations where the robot performs different actions to the ones it recognises, which is the case in all the user modelling scenarios we will present in chapter 5.

The combination of detectors and SCFGs fulfils the requirements for a user modelling framework we specified in section 2.3. This approach is *adaptive* as it has to continuously assimilate the new state of the world. Moreover, the accuracy of predictions is guaranteed as long as the grammar specification is an accurate reflection of the task at hand. Finally, we have already discussed SCFGs ability to manage *uncertainty*.

It is clear however that our approach is inspired by HAMMER, as we make extensive use of the HAMMER features for user modelling we have identified: *action recognition*, *next-step prediction* and *hierarchical organisation*. We have also described a few SCFG techniques that will re-surface later on the thesis. These are:

- Accepting spurious inputs in the grammar
- Iterating over non-terminals of the grammar
- Using the Viterbi probability as a proxy measure for the user's performance of an activity.
- Prediction of the next-step in the sequence.

Finally, this chapter has introduced two software libraries: the HAMMER middleware and SARTParser. Both are written in C++ and coded to be fast and easy-to-use. Im-

portantly both are open-source and have been used in a variety of other projects: the HAMMER middleware for active vision and SARTParser for the application that we will present in chapter 5.

Still, before applying these techniques, we have to introduce the application domain; which is precisely what we will do in chapter 4.

4

ROBOTIC COMPANIONS FOR WHEELCHAIR USERS AND HOSPITAL PATIENTS

The potential for robots to be human companions is often discussed in the robotics literature (Dautenhahn et al., 2005) as well as popular culture. An interesting and challenging related question is in what actual scenarios robotic companions might be useful. In this chapter, we explore two such scenarios: driving a power-wheelchair and being a patient at a hospital. The capabilities of the robot companion have to be tailored for each activity. For instance, the robot can point at obstacles and give directions as a wheelchair sidekick; whereas it can act more like a conversational agent in the role of hospital companion.

Both of the eventual target populations in each scenario —children with cognitive difficulty and socially isolated hospital patients— would greatly benefit from extra companionship, but may not be able to get it due to their carers already being overstretched.

There are also particular reasons as to why a companion would be useful in each scenario. For power wheelchairs users, a robotic companion can make driving —a potentially cognitively taxing task— feel more like a game. This is important as it may motivate users and allow them to drive for a longer period of time. Moreover, the robot has access to low level information about the wheelchair so it can give advice about things that are not obvious to the driver (eg. that obstacle is not letting the wheelchair move forward).

In hospitals, a robotic companion may be a useful tool fighting social isolation. What is more, it may do so with some of the most challenging patients, people with dementia.

This chapter features NAO as a companion in both roles. We will provide a description of the underlying hardware and software that allow NAO to act as a companion. Moreover, we will document our efforts to verify whether users would accept a robotic companion. Our studies and experiments show that robotic companions can appeal to people of all ages, from four year old children to a hundred year old hospitalised patients (fig. 4.1).

The results presented in the first half of this chapter were based on our previously published work (Sarabia and Demiris, 2013). In addition, the second part of this chapter has been submitted to an international robotics journal for peer review.

(a) Child driving ARTY with NAO as a wheelchair sidekick.

(b) Hospital patient interacting with NAO as a companion to fight social isolation.

Figure 4.1: Different roles for NAO as a companion with diverse users.

4.1 A ROBOTIC SIDEKICK FOR WHEELCHAIR USERS

Driving a wheelchair is a cognitively challenging task (Massengale et al., 2005). Users need to predict the behaviour of the wheelchair, and be spatially aware of their surroundings. Could there be a way of reducing the cognitive requirements driving a wheelchair demands? We hypothesise that a robotic companion may be able to do so. This way, we aim to lower a wheelchair's entry barrier for people with cognitive disabilities and children whose cognitive faculties may not be fully developed.

To that end we added NAO to a paediatric wheelchair, the Assistive Robotic Transport for Youngsters (ARTY), so that it can act as a companion for mobility-impaired persons. In this setting, we propose two tasks for NAO to carry out. In the first one, NAO points out the location of obstacles, explaining why the smart wheelchair may not be moving in the expected direction. For the second role, we set NAO to act as a driving aid giving directional instructions and compare it to more traditional driving aids (such as voice and on-screen arrows).

We were also interested in understanding whether there are any benefits to having a physical robot as a companion rather than a simulated one. Our results will show that, at least for adults, participants much preferred the physical robot over the simulated one, even if the performance differences were inconclusive.

4.1.1 Wheelchair sidekick system description

Our system, depicted in fig. 4.2, has two distinct subsystems: the Assistive Robotic Transport for Youngsters (ARTY) and Aldebaran's NAO. In-depth descriptions of both ARTY and NAO can be found in appendices R.3 and R.6 of this thesis. For both the children trials and the adult experiments that we will present later we made use of two NAOs. This way, when the battery of one of them ran out we could continue the experiments with the other.

Figure 4.2 highlights the main hardware features of the wheelchair sidekick system. Correspondingly, fig. 4.3 summarises the main software components of the system, all of which are written atop the Robot Operating System (ROS) which we introduce in appendix R.1.

The user studies we carried out with the wheelchair sidekick consisted of driving the wheelchair through a pre-determined path and, at the path junctions, checking and following the directions of one of four driving guides (NAO robot, NAO simulator, Wayfinder Software and Computer Voice). In another user study, participants drove the wheelchair freely and when they were near an obstacle the wheelchair would stop and NAO would point at the obstacle. We will describe these studies in greater detail in sections 4.1.2 and 4.1.3. For the moment, this serves as context for the discussion of the wheelchair sidekick software components.

Laser Combiner[1] takes the readings of the three laser rangers and combines them into a single coherent message (Soh and Demiris, 2012). Laser Scan Matcher[2] takes in this combined laser message and interpolates the wheelchair odometry using Censi's iterative closest point algorithm (2008). The Adaptive Monte-Carlo Localisation (AMCL)[3] node receives both the interpolated odometry and the combined laser message to localise the wheelchair on a pre-defined map. The Obstacle Avoidance[1] module moderates the input joystick signal according to the wheelchair proximity to obstacles using a dynamic window approach, thus avoiding potential collisions (Soh and Demiris, 2012).

We designed a few ROS nodes specifically for the wheelchair sidekick system. Navigation Reporter raises an alert whenever the user has deviated from a pre-recorded path on the map and indicates the direction the user should follow at the junctions of said path. Every time a new pose is generated by the AMCL node, Navigation Reporter tries to find its closest match in the list of poses that compose the pre-defined path. To achieve

1 Nodes written by Harold Soh.

2 Node written by Ivan Dryanovski and William Morris, available from http://www.ros.org/wiki/laser_scan_matcher.

3 Node written by Brian Gerkey and Andrew Howards, available from http://www.ros.org/wiki/amcl.

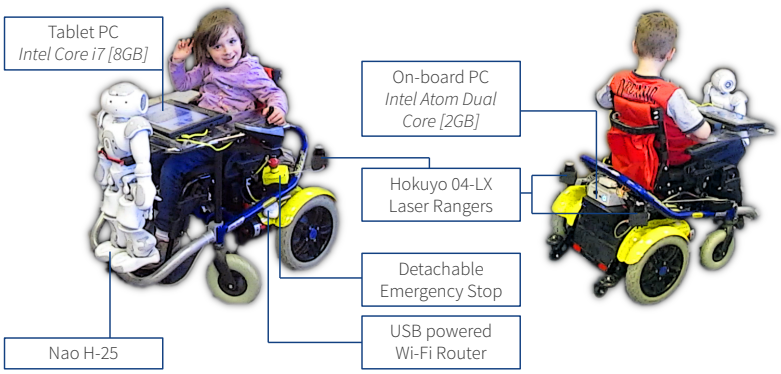


Figure 4.2: Hardware components of the wheelchair sidekick system.

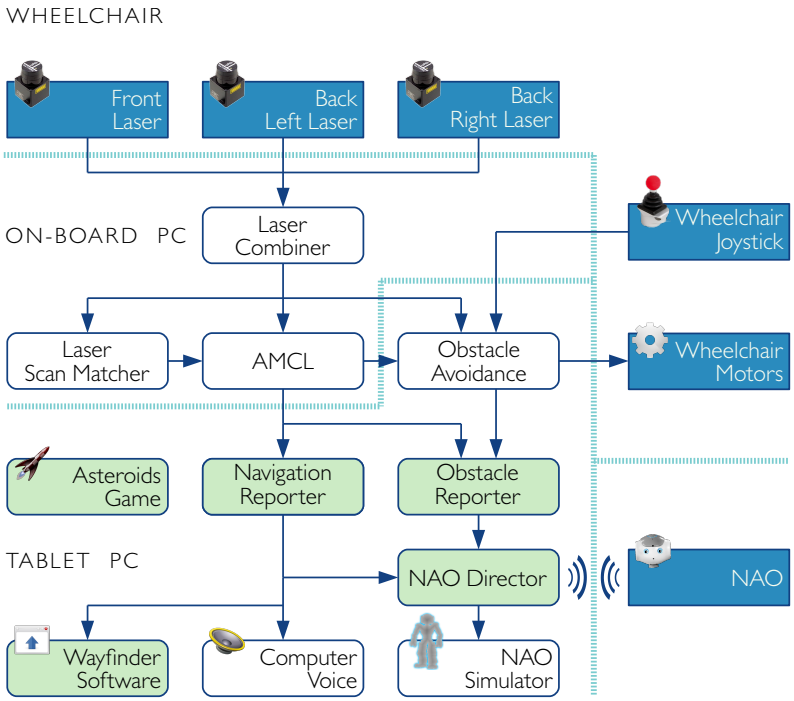


Figure 4.3: Software components of the wheelchair sidekick system. Square edges with blue background represent hardware, round edges represent software nodes. Green shaded nodes are bespoke nodes of the system. Discontinuous lines show the boundaries between the different computers. See section 4.1.1 for authorship of other nodes.

this, the algorithm evaluates all path poses within ±1.5 metres of the last known path pose and chooses the one with highest score. The scoring function is defined as:

$$\text{score} = e^{\left(\frac{\Delta}{\alpha} \ln 2\right)} + e^{\left(\frac{\Gamma}{\beta} \ln 2\right)} \tag{4.1}$$

where Δ represents the euclidean distance between the current wheelchair pose and the candidate path pose, Γ is the normalised angle difference between the current wheelchair

Figure 4.4: Asteroids game designed to serve as a secondary task to increase the cognitive load of participants in our experiments.

pose and the candidate path pose; α and β are adjusting variables to control what distance and angles difference yield half score (we set α =0.4m and $\beta = \pi/4$ rad, these values were determined empirically). If the score of all path poses considered was lower than a threshold (heuristically set to 1.3) no pose was chosen. This formulation is robust to errors in AMCL and can deal with paths that go over the same point repeatedly, thanks to the $\pm$1.5m sliding window. We determined the length of the sliding window heuristically and made sure it was smaller than the length of any of the loops in the pre-defined paths.

Obstacle Reporter takes the information from the Obstacle Avoidance node and raises an alert if the user is driving towards an obstacle. To prevent NAO from overwhelming the user with information, both Obstacle Reporter and Navigation Reporter suppress similar alerts that occur in a short period of time.

The asteroids game (fig. 4.4) was developed as a secondary task for the path driving experiments. Its role, as we will describe in section 4.1.3, is to increase the user cognitive load to simulate real wheelchair-driving conditions. The objective of asteroids game is to move the spaceship (triangle in the figure) *away* from the incoming asteroids (circles in the figure) using the up and down arrows. The game keeps track of the total number of impacts: the higher the impacts, the less attention the user was paying to the game.

There are several other bespoke nodes whose task is to communicate to the user the alerts raised by the wheelchair. We start with NAO Director, which coordinates all of NAO's movements and speech. It instructs NAO to execute wake-up and power-off animations and provides a background behaviour for NAO —randomly looking around and blinking. If the node receives any alert from either Navigation Reporter or Obstacle Reporter it will stop the background behaviour and command NAO to indicate the dir-

ection the user has to follow (in case of a navigation alert) or the location of the obstacle (in case of an obstacle alert). Thanks to NAO Director, NAO appears more human as it is continuously moving and relays information to the user in a coordinated fashion.

NAO Director can command both the actual NAO or an on-screen simulated NAO. The underlying code is the same. However, the simulated NAO cannot emit sounds and instead writes them on the display. To minimise the differences between the simulated NAO and the physical one, NAO Director did not provide new information through voice, instead giving vague messages (eg. "we have to go that way"). One has to look at the arms of the robot or the simulator to understand its instructions.

We also implemented two other, more traditional, driving aids. The first one is the Wayfinder Software, a simple computer window to show navigation alerts. Upon receiving an alert, an arrow indicating the direction the user should follow is shown on-screen for three seconds. The other traditional driving aid was the Computer Voice, a program which would speak out loud the instructions received from Navigation Reporter using ROS text-to-speech facilities. Example utterances are: "go straight" and "drive left". The sound of the ROS computer voice was easily distinguishable from that of NAO.

4.1.2 Pointing obstacles to children in wheelchairs

In order to evaluate the wheelchair sidekick system, we took advantage of our university's open-day in May 2013 and conducted trials of the system. Since we had very little control over the environment (it was estimated that 10,000 people attended the event) the main focus of the exercise was to study children attitudes towards the robotic companion.

Mapping the exhibition floor and pre-recording path for users to follow was challenging due to the uncontrolled and dynamic nature of the open-day. For this reason we limited the system capabilities to obstacle avoidance —hence NAO only reported the location of obstacles. NAO had its default behaviour enabled (start-up animation, saying hello and bye, blinking and looking around) as well as obstacle reporting (it would point in the direction of the obstacles).

We recorded 1 hour and 20 minutes of data. During this time a total of 566.96 metres were driven (as estimated by the Laser Scan Matcher module). The system detected 4,695 potential collisions and NAO warned the children 173 times. This disparity can be explained since near-collisions tend to be both spatially and temporally clustered and the Obstacle Reporter module only raises an warning when it considers the potential collision to be novel.

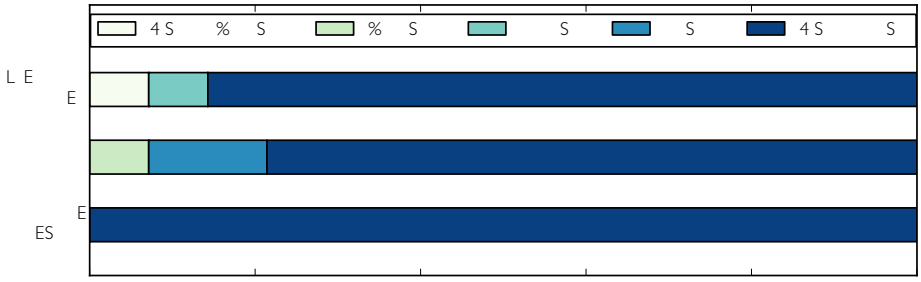


Figure 4.5: Questionnaire responses of 14 children who tried the wheelchair sidekick system.

We asked 14 participants, who had driven the wheelchair for about 5 minutes each, to fill in a questionnaire with three questions in five-point Likert-scale; though, to make the questionnaire friendlier we used smileys instead of numbers following Markopoulos et al. (2008). The responses came from 9 boys and 5 girls aged 4 to 12. Some of the younger participants were helped by their parents when filling in the questionnaire.

Children were generally very positive about the field-trials as can be seen from fig. 4.5. All 14 children strongly agreed they had enjoyed the wheelchair, 12 strongly agreed NAO had helped them not to crash and 11 participants strongly agreed they liked having NAO by their side when driving. The participants also had the opportunity to write comments about the experiment, which all were positive. We list below a few representative comments:

- ✎ "I want him as a pet or brother"
- ✎ "I liked that the robot spoke. It was like he was my friend"
- ✎ "I think that this is a really good project and that it will really help disabled children!"
- ✎ "I loved it when there was someone in front of me and NAO told me to stop"
- ✎ "NAO is really helpful"
- ✎ "I think HE IS AWESOME!"

Although there was an inherent pressure on the children to evaluate the wheelchair positively —it is hard to criticise the toy one has just played with— we consider the enthusiastic comments to be a proof that, at least in the short term, children really enjoy having NAO as a companion. This finding echoes similar ones in the literature (Belpaeme et al., 2012; Ros et al., 2014).

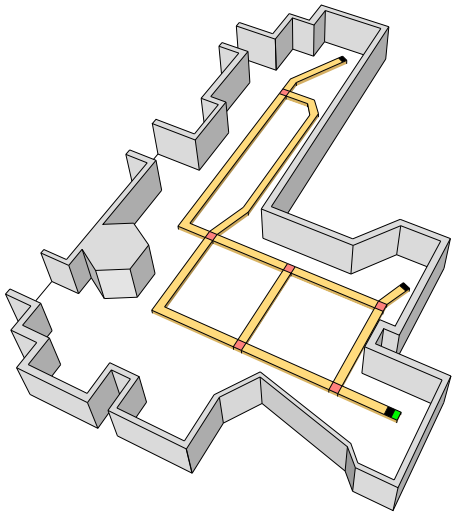


Figure 4.6: Path-driving environment with 6 junctions (in red) and 3 finishing points (in black). The starting point is indicated in green.

4.1.3 Giving driving directions to adults

Besides investigating whether children enjoyed interacting with NAO in a wheelchair, we wanted to test if NAO would be a useful driving aid. We also considered whether there was any advantage to having a physical robot rather than a simulator.

4.1.3.1 *Experimental set-up*

Participants were tasked with navigating through a pre-defined path. The basic make-up of each path was the same, all shared the starting point and driving segments, the difference came from the turns to be performed at each junction and the finishing point. There were a total of 6 junctions and 3 finishing points (fig. 4.6). All paths were designed so it would take around three minutes to complete them.

The participants did not know the exact directions to follow on the course, instead relying on different driving aids to guide them through the pre-defined paths. Specifically, driving aids told users to turn left, right or keep going straight at each junction. If participants did not follow the instructions the system would remember the last point where they had been on-course and instructed them to retrace their steps until they reached a known point.

To simulate real wheelchair driving conditions where a user may be distracted by some other task (eg. by having a conversation) and following the example set by Carlson

and Demiris (2012), we devised a secondary task: the asteroids game introduced in section 4.1.1.

Participants were asked to complete four paths. Each time, they were guided by one of the driving aids already described (NAO robot, NAO simulator, Wayfinder Software and Computer Voice). The order in which each participant used every driving aid was determined using a within-subjects design with conditions allocated randomly. At the end of each trial, they were asked to complete a questionnaire regarding the driving aid they had just used. Additionally, at the end of the experiment they were requested to chose their preferred driving aid.

Throughout these experiments Obstacle Reporter was disabled and NAO only acted as a driving aid. This was done as a control measure since neither of the traditional driving aids could communicate obstacle alerts. All data interchanged by the different ROS nodes of the system, including scores on the asteroids game, was recorded for subsequent analysis of user performance.

4.1.3.2 *Performance metrics and questionnaires*

The following metrics are used to compare the different driving aids:

AVERAGE DRIVING SPEED computed as the sum of the euclidean distances between the poses reported by AMCL over the time the trial lasted. A higher value indicates better overall performance in the task.

ASTEROIDS GAME IMPACTS defined as number of impacts in the secondary task. A high score indicates participants attention was occupied with driving. Accordingly, a lower value implies that driving the wheelchair is less taxing cognitively. Thus, a lower value points to higher overall performance.

TIME LOST computed as sum of the durations between a user receiving a *lost* instruction and the user receiving any other instruction, which only happens when the participant has driven back to the junction where she became lost. A lower value is suggestive of better overall performance.

We also collected subjective metrics through the use of questionnaires. The questions are listed below (note system was replaced by the actual driving aid):

- "I found driving the wheelchair whilst playing the game difficult." (*Difficulty*)
- "I felt safe in the wheelchair." (*Safety*)

- "I understood the instructions the system was trying to convey." (*Understanding*)
- "The system gave me accurate instructions." (*Accuracy*)
- "The system distracted me from driving." (*Distraction from driving*)
- "The system distracted me from playing the game." (*Distraction from game*)
- "I found the system to be a useful driving aid." (*Usefulness*)

4.1.3.3 *Results*

20 people (11 female and 9 male) aged between 21 and 38 completed the experiment. 40% of the participants declared to have worked with robots before. Participants were mostly university students, though no one was actually working in robotics. Due to a misaligned laser scanner we had to discard one of the trials as the AMCL module failed to localise properly.

Even though participants had a few minutes practice with the wheelchair before the actual experiment started, we found most of the learning occurred during the first attempt. This is evident from fig. 4.7 where the deviation from the average performance is greatest in the first attempt for all three metrics. Therefore, we report on data from *all* attempts as well as from the *last three* attempts.

Figure 4.8 shows the questionnaire results for the simulator and the robot across all attempts (N=19). These results are checked for statistical significance using the Wilcoxon signed-rank matched pairs test which is a non-parametric, within-subjects, two-tailed test. *Safety* and *accuracy* have equivalent scores for both the simulator and the robot. When asked about *difficulty*, *understanding* or *usefulness*, ratings are generally higher for the NAO robot but the differences do not reach statistical significance. In contrast, the robot has significantly lower ratings for *distraction from driving* ($p = 0.007$, $W = 11$) and *distraction from game* ($p = 0.032$, $W = 14.5$).

It is noteworthy that when considering only the last three attempts (N=9)[4] both *distraction from driving* and *distraction from game* were no longer statistically significant, though they still favoured the robot. *Difficulty* and *understanding* continued to have higher, non-significant, ratings for the robot, whereas *safety* and *accuracy* had very similar ratings. Importantly, 89% agreed or strongly agreed that the robot was *useful* compared to only 33% for the simulator; and this was found to be statistically significant ($p = 0.031$, $W = 21$).

4 This corresponds to the number of people who did not have either the NAO robot or the NAO simulator on the first attempt.

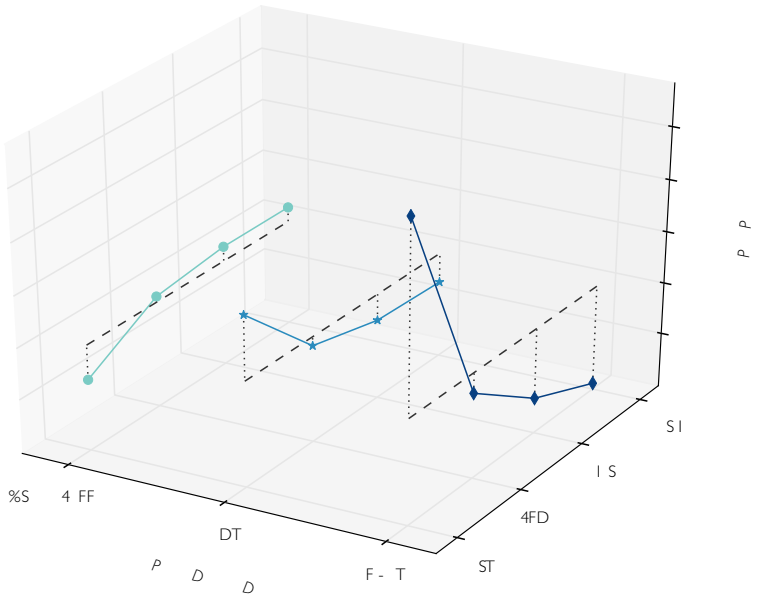


Figure 4.7: Deviation from average in driving performance across path-driving experiments. Results from different driving aids have been aggregated. From the figure, it is clear that users had lower performance with all aids on the first trial.

Recall that there are three performance measures: driving speed (v), asteroid game impacts (i) and time lost (τ) which we extracted from the recorded experimental data. Considering all trials, we found that, the median driving speed was $\tilde{v}_{robot}$ =0.19 m/s (IQR: 0.10 m/s) and $\tilde{v}_{sim}$ =0.21 m/s (IQR: 0.11 m/s) for the NAO robot and the NAO simulator. Similarly, the median number of impacts was $\tilde{i}_{robot}$ =69 (IQR: 45.75) and $\tilde{i}_{sim}$ =64 (IQR: 60.5); the time lost was $\tilde{\tau}_{robot}$ =21.91s (IQR: 82.69s), $\tilde{\tau}_{sim}$ =8.27s (IQR: 72.53s). If only the last three trials are considered (N=9) we find that $\tilde{v}_{robot}$ =0.24 m/s (IQR: 0.08 m/s), $\tilde{v}_{sim}$ =0.22 m/s (IQR: 0.07 m/s); $\tilde{i}_{robot}$ =52 (IQR: 55), $\tilde{i}_{sim}$ =56 (IQR: 43); $\tilde{\tau}_{robot}$ =0s (IQR: 15.86s), $\tilde{\tau}_{sim}$ =30.73s (IQR: 57.49s). None of these differences were found to be statistically significant using once more a Wilcoxon signed-rank matched pairs test.

We were also interested in whether either the NAO robot or the NAO simulator would yield better performance than the control driving aids. Our data shows that the wayfinder software had the best median performance metrics: $\tilde{v}_{wayfinder}$ =0.24 m/s (IQR: 0.06 m/s), $\tilde{i}_{wayfinder}$ =49 (IQR: 43.25), $\tilde{\tau}_{wayfinder}$ =5.57s (IQR: 30.64s). Followed by the computer voice: $\tilde{v}_{voice}$ =0.22 m/s (IQR: 0.08 m/s), $\tilde{i}_{voice}$ =52 (IQR: 43.25), $\tilde{\tau}_{voice}$ =7.23s (IQR: 23.93s). Followed by the NAO simulator and NAO robot (see above for medians and interquartile ranges). Performing Friedman tests reveals a statistically significant effect of the driving aid on all three performance metrics. We found T=10.45 and p=0.015

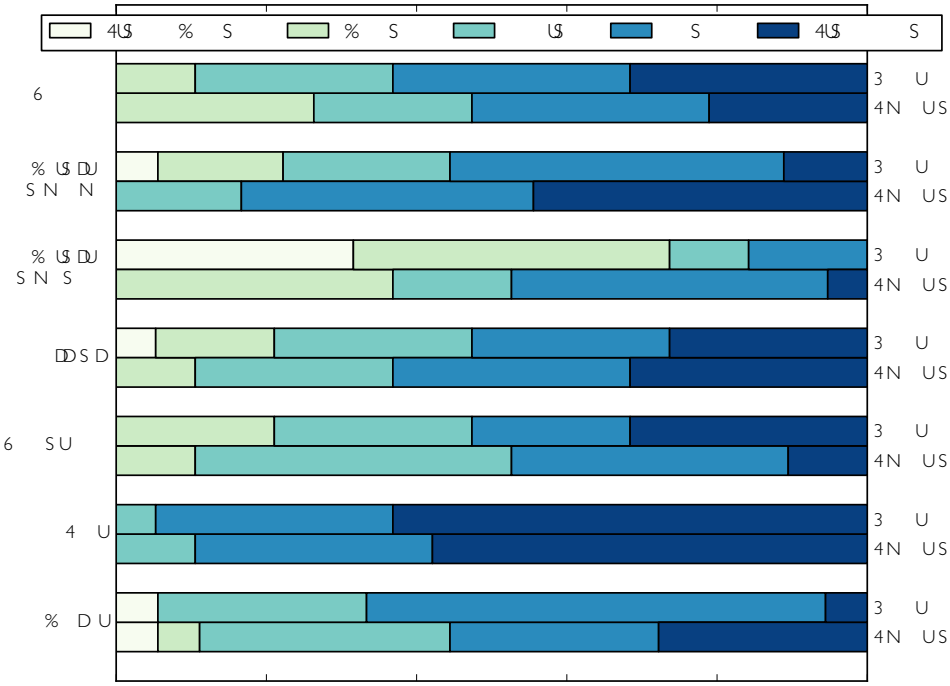


Figure 4.8: Questionnaire responses for Simulator and NAO considering all path-driving experiments attempts. Categories appended with * are statistically significant. See section 4.1.3.2 for actual questions.

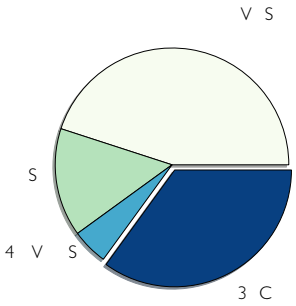


Figure 4.9: Preferred driving aids to give instructions to adults driving a wheelchair.

for driving speed; T=12.97 and p=0.005 for impacts; and T=9.41 and p=0.024 for time lost. Notice that, once more, N=19.

Figure 4.9 shows the preferred driving aids of participants. Remarkably, the robot was the second most preferred choice, with 30% more participants choosing it over the simulator.

4.1.4 Discussion

The robot and the simulator had very similar scores in *safety* and *accuracy*. This was expected as *safety* depends on the Obstacle Avoidance module which was active throughout the experiment. Likewise, the *accuracy* of instructions relies upon the Navigation Reporter which is shared by both simulator and robot.

In most other categories the robot had a slight advantage in ratings when compared to the simulator, though most of these differences did not reach statistical significance. The exceptions were *distraction* and *usefulness*.

Interestingly, the robot scored significantly lower in the *distraction from game* category than the simulator. This is surprising since the simulator shared the screen with the game and participants did not have to look away. Moreover, even if the *accuracy* and the actual task performance metrics were similar across both driving aids, users preferred the robot over the simulator (30% difference when it came to the favourite driving aid). One possible explanation is that the lack of embodiment may cause the simulator to be more distracting and less useful. Further research is needed to clarify this question.

Although the simulator presented higher driving speed, lower number of impacts and less time lost, none of the results were statistically significant. Furthermore, when considering the last three attempts we found the opposite was true: the robot had a slightly higher driving speed and a considerably lower time lost. This suggests more time may be required to habituate to NAO than to the simulator.

Taking into account the traditional driving aids it is clear that the robot and the simulator were not as effective as the control driving aids. More work is needed to ensure these robotic driving aids catch up with or even improve upon their traditional counterparts.

To be specific, we identified two issues which might explain why the performance of NAO was lower than that of the control driving aids. Firstly, NAO sometimes took too long to give instructions due to unpredictable network latency, which confused participants. Secondly, many participants were disappointed and mentioned in the comments that NAO never gave instructions by voice. By exploiting multi-modality NAO may become a more effective driving aid.

4.2 A COMPANION FOR HOSPITAL PATIENTS

Depression and cognitive decline are common sequelae of social isolation and disengagement of patients in hospitals (Lowenthal, 1964). As the population grows older, the problem of social isolation in hospitals becomes even more of a priority.

Previous studies illustrate the magnitude of these problems. Depression is common in hospitals: Shah et al. (1997) revealed that up to 46% of older patients are depressed using the Geriatric Depression Scale; yet, as many as 90% of these cases are not identified. Further, studies show that direct patient contact time makes up only 12% of the junior doctors time (Block et al., 2013). Similarly only 50% of nurses time is spent in direct contact with patients (Storfjell et al., 2008). These results suggest patients may spend the day with little or no social interaction. And with an expected shortage of healthcare workers due to ageing (European Commission, 2012), social isolation will only increase.

We now report the outcomes of an exploratory trial carried out during one week at a busy hospital in London. Though similar trials have been performed with children (Ros et al., 2014), the trials in this section represent the first of their kind with adult patients in an acute hospital setting.

In contrast with the wheelchair sidekick system where NAO was fully autonomous, for this trials we control NAO through a Wizard of Oz (WoZ) interface (Riek, 2012). This was to be able to properly react to unexpected situations in the hospital, a much less controlled environment than the lab where the previous experiments took place.

4.2.1 Hardware and software description

An overview of the software and hardware components described in this section is shown in fig. 4.10.

The trials were carried out with NAO (appendix R.6). NAO was programmed to send audio captures from its microphone and video from its top camera to the remote operator. However, the Wi-Fi antenna's low bandwidth meant that the sound was noisy with occasional fragments of sound missing and that the camera resolution was limited to 160x120 pixels.

Following our experience with the wheelchair sidekick system, we used two NAOs during trials, though only one at a time. Having two robots allowed us to run trials continuously without the risk of empty batteries or overheating robot joints.

NAO stood on top of a rolling table with its feet anchored to the table by velcro, this way the 58cm tall robot was able to interact with patients face to face. This set-up

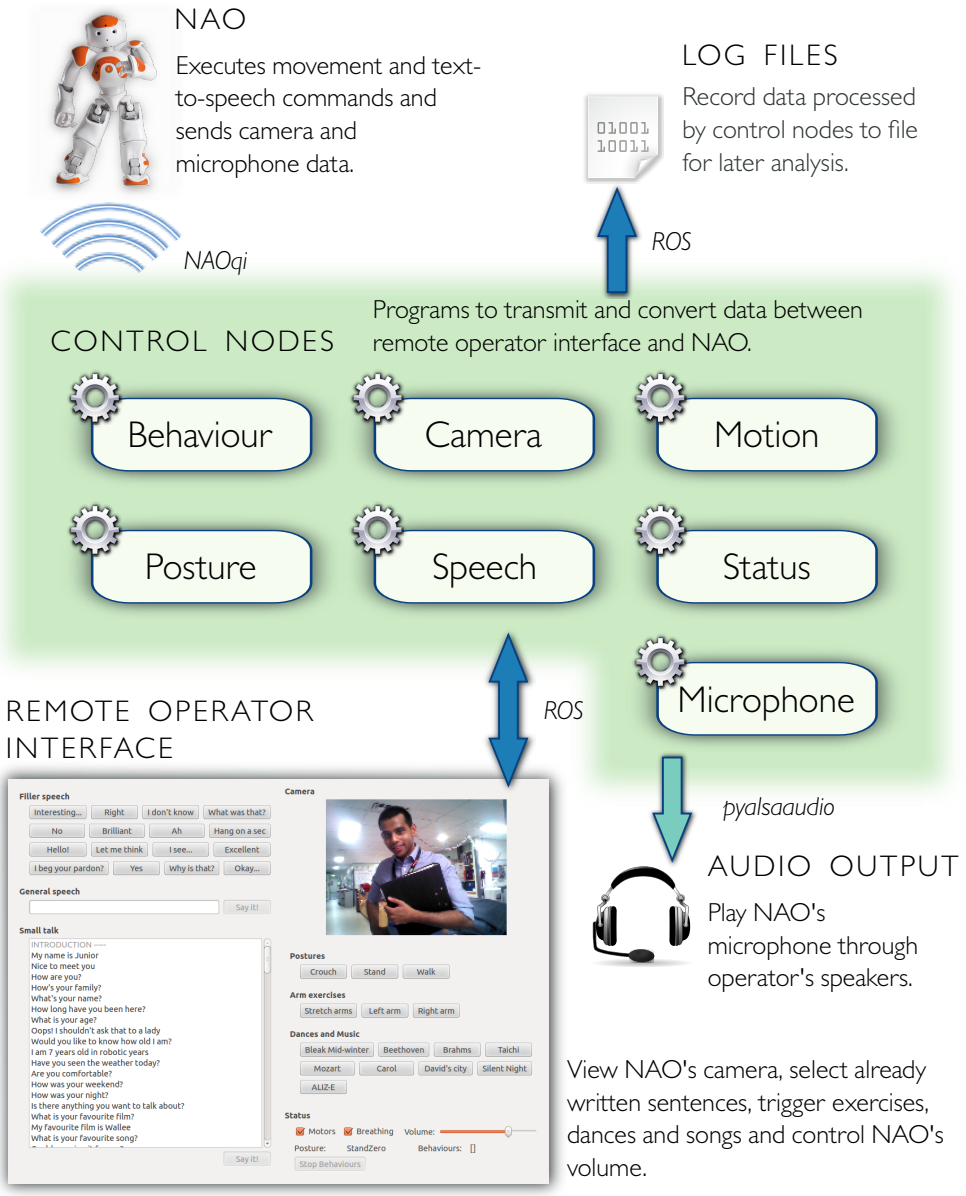


Figure 4.10: Overview of the hardware and software components of NAO as a hospital companion. Elements in italics are computer libraries. Rounded boxes are programs to relay information between NAO and the operator interface. See section 4.2.1 for a detailed explanation of each component.

Figure 4.11: Overview of the remote operator interface. Left-half of the interface is dedicated to NAO's speech with conversational fillers as well as common sentences. The right-half controls the pre-recorded behaviours and the camera of NAO.

also had the advantage of allowing us to move the robot from bed to bed quickly. The flexibility lost by anchoring the robot feet was balanced with the extra safety of stopping the robot from falling on the patient in the event the motors failed.

We wrote several programs to control different aspects of the robot (*control nodes* in fig. 4.10). All programs were written in Python and made use of the ROS library (appendix R.1) for internal communication. The NAOqi library was used for our programs to control NAO and PyAlsaAudio to output sound (*audio output* in fig. 4.10).

The remote operator interface allowed the user to trigger robot behaviours as well as to type text for NAO to say (*remote operator interface* in fig. 4.10). The interface was designed as to minimise the operator's reaction time, as can be seen from fig. 4.11. The final user interface is the result of several iterations of trial and testing. Originally, the interface only had an input box for NAO's speech, but we found that easily accessible fillers gave extra time to the operator to type a complete answer. Similarly, the pre-written sentences in small talk gave the operator some structure for the interaction. Controls for rapidly adjusting NAO's speaking volume were found to be important very early on, as was the ability to trigger stored behaviours on the robot.

ROS has the ability to save all the information that it transmits to a log file (*log files* in fig. 4.10). During trials we recorded interaction aspects such as robot's camera view and the robot activities into the log files. These were later used for analysis purposes.

Putting all these components together allowed us to program NAO so it was able to perform the following actions to socialise with patients:

- Ask basic questions about patients lives.
- Tell jokes.
- Read a verse of poetry.
- Read the news (pre-fetched on the morning).
- Play classical music pieces.
- Demonstrate arm flexing and stretching exercises and then request the patient repeats them.
- Perform two dances: one was Tai chi inspired and relaxing; the other was more energetic and based upon the work of Ros et al. (2014)

4.2.2 Trial procedure

Trials took place in the Chelsea and Westminster Hospital in London from the 15th to the 19th of December 2014. Sessions were carried out from 9.30am until 5.30pm every day. Patients were chosen from the following wards: the stroke and neurology rehabilitation unit; the orthopaedic, urological, general and plastic surgery unit; the intensive care unit; the gastroenterology, general surgery and bariatric surgery unit; the gynaecology unit; and the gastroenterology, endocrinology and haematology unit. Most patients were located in a bay with 6 beds, though a few had their own individual room. A total of 69 patients were offered to take part in the trials.

The introduction of robots at Chelsea and Westminster Hospital was approved by the Medical Devices Committee, Safety and Effectiveness Group, Clinical Engineer (Reference 266, v01.38) and the Research and Development Department at Chelsea and Westminster NHS Foundation Trust. These trials were granted Clinical Governance and Caldicott approval in September 2014 (CAPP 1087) and were conducted in accordance with the Patient Protection Act 1998.

For each session, a Clinical Research Fellow (CRF) carried NAO near the bed of a patient and obtained her consent for participation in the trials. If the patient refused, the CRF thanked her and moved on to another bed. If she accepted, the CRF proceeded to introduce NAO. At this point, the remote operator would start controlling NAO and the CRF would leave the patient's bed. Thus the patient was left to interact with the robot alone; the CRF remained in the ward —a few steps away— in case his intervention was necessary.

Subsequently, the robot tried to engage the patients with conversation, jokes, dance, music and exercises. The robot started by asking the patient how she was feeling, inquiring about the weather and her family. It would also offer to tell jokes or read the news. If all this failed to catch the attention of the patient, then the robot would start playing music or dancing. In general, the approach was to try a range of topics or activities until the user became engaged.

When all interaction possibilities had been exhausted the robot operator indicated the CRF to take the robot away and escort it to the next patient's bed.

Once the robot had been taken away, if the CRF —a trained medical doctor— had identified a patient as having dementia, this was noted down on the interactions log. The age and gender of the patients as well as whether they had made eye contact and talked to the robot were also annotated on the interaction logs. Neither the medical conditions of the patients nor the length of their stay were recorded.

To ensure consistency, the same person —the author of this thesis— operated the robot for throughout all the interactions.

After the second interaction, or if it became clear the patient would not be able to have a second interaction due to time constraints, the CRF would give the patient a questionnaire to fill in. In several occasions, patients that had interacted with the robot did not fill in a questionnaire, leading to the disparity between the number of questionnaires and interactions recorded. This situation mostly arose when we expected to have a second interaction with a patient who was later not present. The questionnaires asked for age, gender, comments as well the following questions:

- Have you enjoyed interaction with the robot during your time in the hospital? (*Enjoyed interaction*)
- Would you want to use the robot again? (*Would like to use the robot again*)
- Do you think the robot will be a useful tool in patient care in the future? (*Robot will be useful in the future*)

Note that the questionnaires were anonymous, which meant cross-correlating them with the recorded interactions was not always possible afterwards.

Most patients filled in the questionnaire by themselves. However, if the patient was not able to do so the CRF would then read the questions out loud and would write down the answers.

For analysis purposes, we split the participants into three age groups: *group <60* is composed of patients whose age is 59 years or less, *group 60–79* contains all participants aged 60–79, finally *group ⩾80* gathers everyone 80 years old or more. We chose these groupings to evenly split the study population.

The interaction duration metric was extracted automatically from the recorded logs and was defined as the time elapsed between the first and last time the robot spoke to the patient.

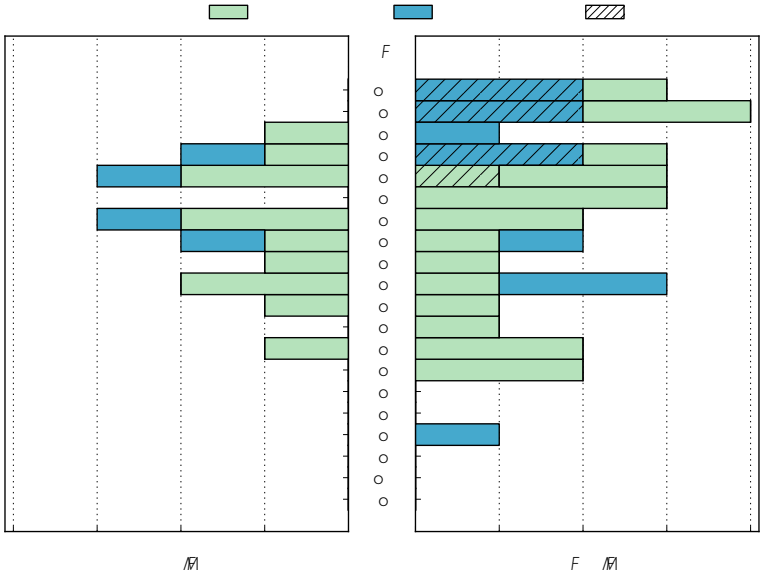
4.2.3 Results

In total, 49 patients interacted with the robot, 32 of them were female and 17 male (see fig. 4.12). The ages of subjects varied from 18 to 100 with 16 in each of the age groups. One of the participants' age is unknown and was excluded from the age analysis. 7 women aged 79–99 had dementia. Moreover, 15 patients (4 male, 11 female) had a second interaction with NAO.

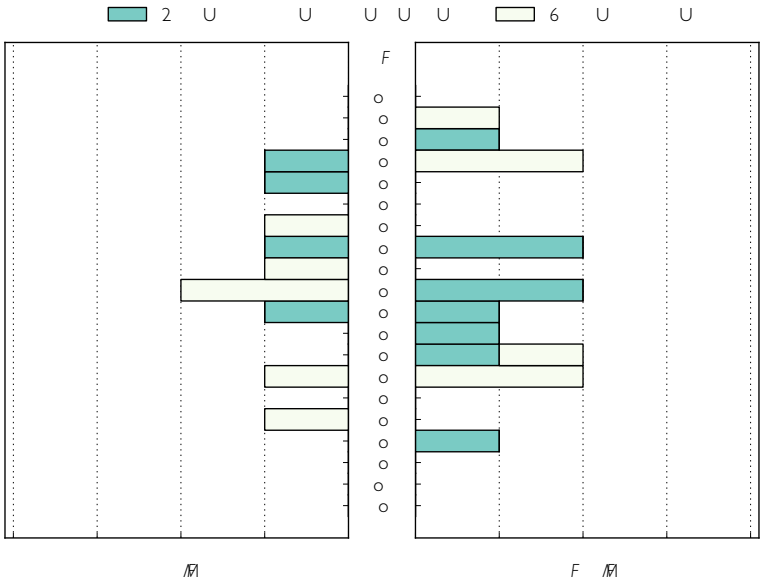
This adds up to a total of 64 interactions spanning 9 hours and 38 minutes, with the median interaction being 8min 39s, and the interquartile range (IQR): 4min 58s.

Additionally, 32 people filled in the questionnaire (12 male, 17 female, 3 undeclared), 15 in *group <60*, 6 in *group 60–79* and 7 in *group ⩾80*. 4 respondents did not declare their ages and were consequently excluded from the age analysis. As mentioned earlier, not all people who interacted with the robot filled in a questionnaire. Consequently, correlating the questionnaires to the interactions was challenging. Still, we were able to uniquely match 13 questionnaires and interactions by using both the age and gender.

Lastly, 20 patients (6 male, 14 female) refused to take part in the trials. Of these 5 belonged to *group <60*, 10 to *group 60–79* and 4 in *group ⩾80* (we were not able to establish the age of one of these patients). Eight patients mentioned being too tired as the reason for their refusal.

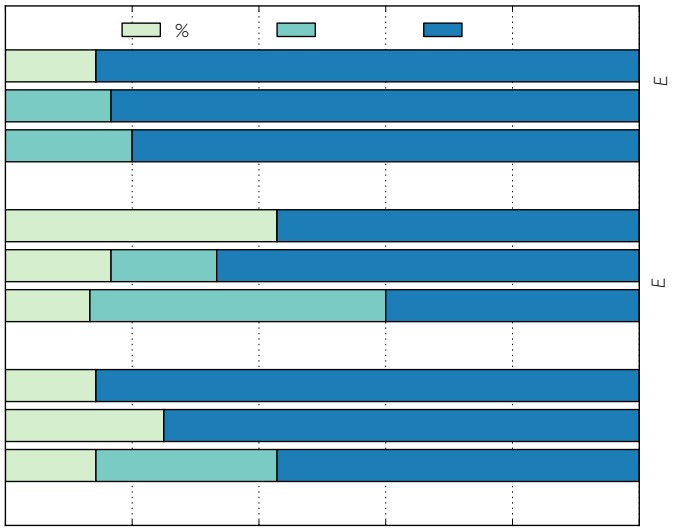


(a) Interaction participants by age and gender. The figure also shows whether participants had dementia and their number of interactions with NAO. Note, 1 male participant did not declare his age and has been excluded from the graph.

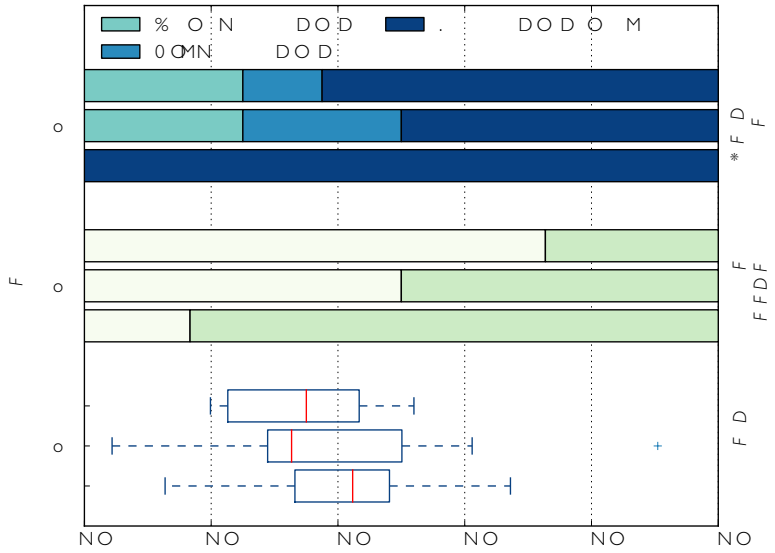


(b) Questionnaire respondents by age and gender. The figure also indicates whether a respondents questionnaire was matched to their interaction. Note, 7 respondents did not declare their age or gender and have been excluded from the graph.

Figure 4.12: Hospital trials population. Discrepancies between graphs are due to incomplete data and not all participants filling in a questionnaire.



(a) Questionnaire responses by age group.



(b) Age differences by type of interaction, response to exercises and first interaction duration.

Figure 4.13: Qualitative and quantitative results of interactions with hospital companion. * correlation between *exercise* and *age* is statistically significant. ** correlation between *made eye contact and talked* and *age* is statistically significant.

4.2.3.1 *Quantitative analysis*

We found that 84% of the respondents agreed with *enjoyed interaction*, 50% agreed with *would like to use the robot again* and 72% were in agreement with *robot will be useful in the future*. Figure 4.13a shows these results broken down by age groups. We checked the Spearman rank correlation between the age and answers of the participants for each question and found none of the correlations to be statistically significant ($p>0.10$). Likewise, a χ^2 test revealed no statistical significant differences in answers to the questionnaires by gender ($p>0.10$).

The pattern was different when analysing the actual interactions, as can be seen from fig. 4.13b. Regarding patient reactions to the robot, we found that all subjects of *group <60* made eye contact, whereas for both other groups 25% of patients did not look at the robot. 25% of those in *60–79* made eye contact but did not talk with the robot, this figure is 12.5% for *group ⩾80*. That leaves 50% of *group 60–79* and 62.5% of *group ⩾80* which both made eye contact and talked with the robot. A point-biserial statistical test suggests a negative correlation ($r= -0.255$) between a patient making eye-contact and increasing age, but does not reach statistical significance ($p=0.080$). Similarly, the point-biserial correlation between verbal interaction and age is $r= -0.356$ and this was found to be statistically significant ($p=0.013$).

24 patients were asked by the robot to perform some arm flexing and stretching exercises with it, 7 of them were in *group <60*, 6 in *group 60–79*, and 11 in *group ⩾80*. The percentages of patients that exercised when the robot asked them to do so were: 86% for *group <60*, 50% for *group 60–79* and 27% in groups for *group ⩾80* and 50% overall. The point-biserial correlation between exercising and age is $r= -0.45$ and this was statistically significant ($p=0.028$).

We also checked if age or dementia was a factor in whether patients made eye contact, talked and exercised, yet a χ^2 test revealed no statistical differences ($p>0.10$).

For *group <60* the median duration of the first interaction was 11min 8s (IQR: 3min 17s), for *group 60–79* the median interaction duration was 8min 10s (IQR: 5min 18s), and for the last age group it was 8min 45s (IQR: 5min 11s). These results did not reach statistical significance ($p>0.10$).

Looking only at those patients who had a second interaction ($N=15$) yields similar results. The median interaction duration for *group <60* was 12min 45s (IQR: 8min 58s, $N=3$), for *group 60–79* it was 6min 35s (IQR: 2min 48s, $N=3$), and for *group ⩾80* the median interaction duration was 6min 43s (IQR: 3min 44s, $N=9$). As before, the Spearman rank correlation was not statistically significant ($p>0.1$).

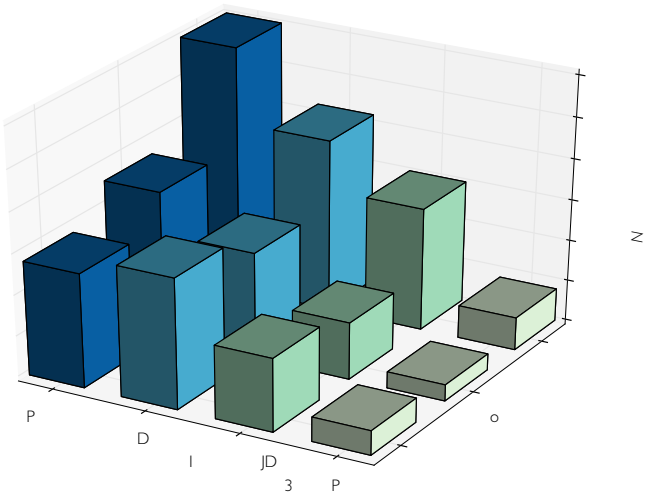


Figure 4.14: Hospital companion activities by age group. Differences in total number of activities by age are statistically significant.

Patients with dementia had significantly shorter interactions. The median first interaction duration for patients with dementia was 7min 10s (IQR: 3min 2s, N=7) whereas for the rest of the patients was 9min 58s (IQR: 4min 57s, N=42). Performing a Mann-Whitney U test shows these differences to be statistically significant (u=77.0, p=0.047). For the second interaction we found similar results: the median second interaction duration for patients with dementia was 3min 23s (IQR: 2min 16s, N=6); for other patients it was 7min 48s (IQR: 1min 25s, N=9). In this case, the results did not reach statistical significance (u=12.0, p=0.087).

Additionally, a Mann-Whitney U test revealed gender did not affect the interaction duration ($p>0.10$).

Interestingly, there were statistical differences between the duration of the first interaction (median: 9min 55s, IQR: 5min 27s) and the second (median: 6min 51s, IQR: 4min 45s) for patients that interacted twice with the robot across all age groups. Furthermore, a two-tailed Mann-Whitney U test showed these differences to be statistically significant (u=63.0, p=0.042).

Concerning NAO's activities during the trials, we found that the total number of activities performed by the robot for the patients correlates negatively with age ($\rho = -0.394$) and this correlation was statistically significant (p=0.012). For a breakdown of the activities by age and type of activity refer to fig. 4.14.

With regards to the 13 questionnaires which could be matched to interactions, we were not able to find any significant correlation between most of the activities the robot performed or the duration of the interaction. There was one exception in that the

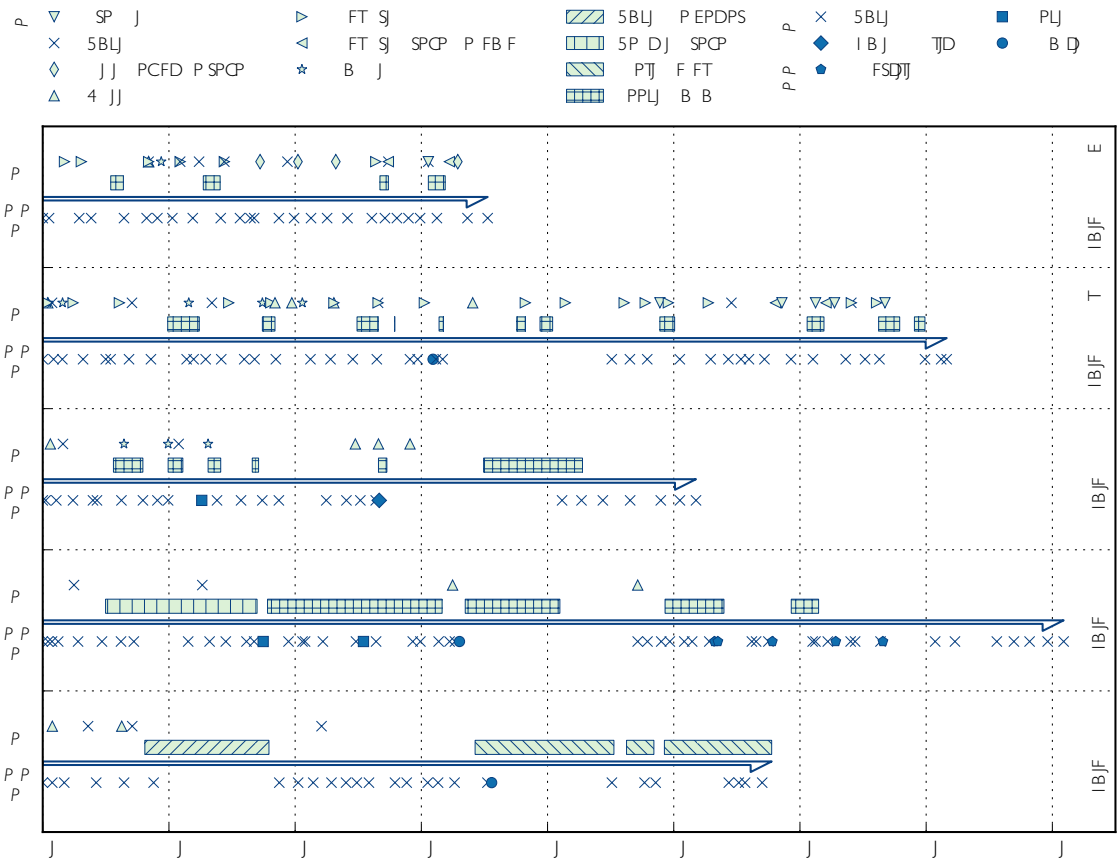


Figure 4.15: Timelines of interactions with 4 patients with dementia aged 84–99. Note patient D had two interactions. The graph shows the actions of the robot and the actions of the patient.

Spearman rank correlation was statistically significant between the number of times the robot danced and whether a patient answered affirmatively to *robot will be useful in the future*: $\rho = 0.629$, $p = 0.021$.

Our data did not show any effect of age or gender in whether patients agreed to participate in the trials ($p > 0.1$ in both cases). We checked the effect between age and participation with a point-biserial correlation, and the effect between gender and participation with a χ^2 test.

4.2.3.2 *Qualitative analysis: interaction analysis of patients with dementia*

In order to capture specific elements of the interaction with patients with dementia, five of their interactions were chosen at random and further analysed, as shown in fig. 4.15. Patient A only spoke to the doctor, and when NAO started dancing she closed her eyes, only to briefly reopen them when spoken to by the robot. Though Patient B did not

speak very much nor did the arm exercises with NAO, she did try to touch it and smiled when the robot spoke to her. Patient C only spoke twice but she smiled many times and even laughed. The interaction strained one minute into the song NAO was playing as she became disengaged, though she re-engaged with the robot once it started speaking again. Patient D was very engaged with NAO: she laughed, smiled, gestured and talked to the robot. Curiously, towards the end of the interaction she started gesturing the robot to go away and frowned when the robot did not do so immediately. Yet this seemed to have no effect on the next interaction when she laughed and smiled and even attempted to give NAO a £10 note. Note that at the end of the second interaction patient D started frowning again and gesturing NAO to go away.

4.2.4 Discussion

Subjective responses towards the quality of interaction with NAO were overwhelmingly positive. There were no statistical differences across age groups on questionnaire data. Notably, 84% of all respondents agreed that they enjoyed interacting with the robot.

In contrast, verbal interaction, engagement with exercise and the total number of activities NAO performed negatively correlate with increasing age in a statistically significant manner. The data suggests the same could be true of interaction duration and eye contact but these measures did not reach statistical significance.

In consequence, one of the challenges is to determine specific activities and interaction duration that can best engage older patients. For example, from fig. 4.14 we can see how joking was not something that older people appreciated NAO doing, in contrast to dancing and singing. This leads to the hypothesis that shorter more tailored interactions —such as music therapy (Wall and Duffy, 2010)— may be more beneficial for older people.

The statistically significant differences in duration between the first and second interactions as well as the fact that only 50% of questionnaire respondents agreed that they would like to use the robot again could be explained by the novelty factor. It is hypothesised that the first interaction with a robot is unusually positive due to novelty. Therefore, as interactions repeat and novelty wears off, users may be less positive towards the robot (Sung et al., 2009). Further research is required to clarify if this is indeed the case.

The strong and statistically significant correlation between the number of times the robot danced and participants agreeing with the *robot will be useful in the future* question may be explained by the fact that the dance was the most sophisticated of NAO's activities, where it had to move its arms, torso, head and played music in synchrony. This differs

WHOLLY POSITIVE (16 comments in this category)	✎ "Improvement in my mood." ✎ "I loved Junior." ✎ "This is a fun way to encourage a return to health." ✎ "[Interaction was] a bit too short, I prefer a longer time" [*93 year old woman*].	✎ "[Junior is] lively and amusing." ✎ "Funny robot." ✎ "Thank you for being chosen to meet Junior." ✎ "He is so good and amazing. Well done Junior."
POSITIVE, BUT SCEPTIC (10 comments in this category)	✎ "It was a bit of a novelty for the day—the idea of it made me laugh, but I wouldn't really need any more time with it." ✎ "[Conversation was] slightly stilted but a great diversion." ✎ "I can't say exactly if I have enjoyed it, but it was exciting." ✎ "I enjoyed but found it a bit intimidating."	
NEGATIVE (3 comments in this category)	✎ "Tried to ignore it but it didn't go away. Its limited vocabulary and intellect made it difficult to converse." ✎ "We have enough machines." ✎ "People want to speak to people."	

Table 4.1: Example of patients' comments classified by *wholly positive*, *positive but sceptic* and *negative*. In total, 29 patients provided comments. *Junior* is the name we gave NAO for the patients.

from conversational interaction which was slower due to delays related to the operator having to type NAO's speech and the noisy audio capture which meant the operator sometimes did not understand what had been said. Thus, patients who saw the robot performing a complex task may have been more agreeable to the future possibilities of using the robot in healthcare.

Results show that interactions with dementia patients were the shortest. Though most dementia patients were in *age group* $\geqslant 80$, the effect of age was not found to be significant in interaction duration. This was in contrast with dementia, which did have a significant effect in interaction duration. Despite engagement difficulties, we observed that many patients with dementia smiled even if they did not speak as much as other patients.

4.2.4.1 *Analysis of Patients' Comments*

Studying the patients' comments on the questionnaires reinforces our findings that patients were mostly positive about their time with NAO, but that interaction needs to be tailored. Example of these comments are shown in table 4.1.

Moreover, eight respondents commented that they thought NAO would be useful with children. This is not surprising since our own results in section 4.1 show how enthusiastic children are about NAO.

Overall, the comments were very positive and followed the answers to the questionnaires, with only 3 people writing decidedly negative comments about the interaction with NAO.

4.3 CONCLUSION

We have presented two roles for NAO as a companion: wheelchair sidekick and hospital companion.

Both approaches showed that most users, across all ages, rated their interactions with NAO positively. Even patients with dementia smiled and laughed with the robot. This was true across widely different scenarios: hospital, lab and university open-day; and with contrasting set-ups: the wheelchair sidekick was fully autonomous and worked in conjunction with ARTY whereas the hospital companion was the only robot but was teleoperated.

Moreover, the wheelchair sidekick path-finding experiments revealed that NAO can be an effective driving aid. Additionally, users found a physical robot less distracting than a simulator —even if no objective performance differences were found. Similarly, the trials carried out at the Chelsea and Westminster Hospital are a proof of concept —to our knowledge, the first of its kind— that robots can act as companions for hospital patients.

There were limitations to both studies. In the wheelchair sidekick's case, we established NAO was not as useful as a traditional driving aid. This was because the task at hand did not fully take advantage of NAO's abilities. Further research is needed to establish how well will our results apply with children with disabilities. For the hospital companion role, the WoZ interface meant the robot was less responsive than it could have otherwise been. The limiting factor lies at the speed the operator could type which usually translated in a 2-3 seconds delay for the robot to reply. Finally, in both roles, NAO could be a more effective companion if the interaction was tailored to each of the users.

The next chapter is dedicated to investigate how to take advantage of the algorithms described in chapter 3 to address these limitations by personalising the interaction with the robotic companions.

5

USER MODELLING APPLICATIONS FOR ROBOTICS

The previous chapter introduced several ways in which NAO could act as a companion for users. There was however a key weakness in the case of the wheelchair sidekick: the interaction was not personalised for the user. It seems unlikely that any social robot could sustain any sort of repeated interaction unless these interactions are personalised for the user (Leite et al., 2009).

We dedicate this chapter to tackle the problem of personalisation. We do so by applying the algorithms we introduced in chapter 3 to create a user model. We present two user modelling frameworks: the first one models the abilities of the user and the second her preferences. These frameworks make extensive use of the properties of Stochastic Context-Free Grammars (SCFGs) such as next-step prediction and using Viterbi probabilities as a metric for task performance.

The frameworks introduced in this chapter may, at first, seem vastly different but as we will explain in the conclusion they are both operationalisations on the Hierarchical Attentive Multiple-Models for Execution and Recognition (HAMMER) architecture.

We will study each framework with a different scenario. We model users abilities in the context of a robotic tutor for wheelchair users, thus merging our wheelchair sidekick with the principles of Intelligent Tutoring Systems (ITSs). The second scenario is human-robot musical collaboration where Baxter creates a drum accompaniment for the user's melody. In both cases the use of SCFGs is justified as both scenarios are made of structured tasks: driving through waypoints for the robotic tutor and selecting musical segments for the human-robot musical collaboration.

The research presented in the second half of this chapter has already been published (Sarabia et al., 2015).

5.1 MODELLING USERS ABILITIES: A ROBOTIC TUTOR FOR WHEELCHAIR USERS

The field of ITSs has many examples of tutors for a variety of domains, from physics (Gertner and VanLehn, 2000) to the Haskell programming language (López et al., 2002). However, to the best of our knowledge, robotic tutors such as Robovie (Kanda et al.,

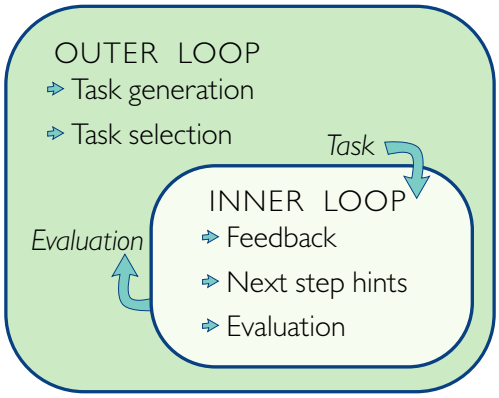


Figure 5.1: Main components of an Intelligent Tutoring System, adapted from VanLehn (2006). The outer loop holds the user model and is in charge of selecting the next task for the user to perform. The inner loop evaluates the user performance for the tasks and additionally provides feedback to the user as well as hints if the user does not know how to continue.

2004) and NAO (Kennedy et al., 2015) do not take advantage of the research carried out in the field of ITSs.

In this section, we present a framework based on Stochastic Context-Free Grammars (SCFGs) which implements the components of an ITS. We further apply this framework to the wheelchair sidekick scenario we presented in section 4.1. This allowed us to avoid some of the pitfalls we previously identified and benefit from the work already carried out for the robotic sidekick.

The framework allows NAO, now a robotic tutor, to give hints and evaluate users performance. These user performance evaluations effectively constitute a model of the user abilities. Having such a model of the users abilities allows the robot to determine whether the user should progress to the next task or try once more the current task.

Our study with adults demonstrates that users without a tutor made incorrect choices with respect to their performance 39% of the times. More importantly, users that were instructed by NAO to repeat a task saw a moderate improvement in their performance.

5.1.1 Implementing an ITS with SCFGs

VanLehn (2006) summarised the main components that any Intelligent Tutoring System (ITS) must have: the *outer loop* and the *inner loop*. Figure 5.1 shows the duties and interconnections of each of these loops. Briefly, the *outer loop* is in charge of selecting the most adequate task for the user, it may even generate such a task if necessary. The

inner loop is executed once the task is chosen, and it evaluates the performance of the user whilst she carries out the task, giving hints and feedback.

VanLehn lists the general requirements of an ITS, but leaves the specifics to the implementer. We now look to implement an ITS with multiple Stochastic Context-Free Grammar (SCFG) parsers. In particular, our focus is to implement the *next step hints* and *evaluation* roles of the *inner loop*.

As we have previously mentioned, SCFGs encode the structure of a task in an intuitive and succinct manner. Furthermore, since SCFGs are probabilistic they can deal with user errors. For this section, we have hand-written the grammars that represent each of the tasks users will have to perform —shown in table 5.1, and discussed in detail later on. We envision that, eventually, these grammars would be manually specified by an specialist. Alternatively, these grammars could also be learnt from demonstration as Lee et al. (2013) did.

The *outer loop* of our system selects a task from the pool of pre-defined tasks and then requests the user carries that task out. After the user has finished, her performance is evaluated and if it exceeds a certain threshold (set empirically) the next task in the pool is selected. This process is repeated until all tasks have been completed.

Under this set-up, the steps of a task are represented as terminals of the stochastic grammar. Hence, whenever a step of the task is detected (through the use of *detectors*, as with the dancing NAO architecture in section 3.5) this terminal/step is fed to the parser representing the currently selected task.

5.1.1.1 *Next-step hints*

SCFGs can be used to predict the most likely next step of a task. Following the formulation in section 3.3.5, we can obtain the terminal probability distribution which gives us the probability of each of the steps of the currently selected task happening next. We denote the terminal probability distribution as $\mathbf{P}(\mathcal{T}_x)$, where $\mathcal{T}_x$ is the set of all terminals/steps in x, which is the currently selected task. If the system is requested a hint, it then recommends the terminal with highest probability in the predicted probability distribution, $\arg\max_{t \in T_x} (\mathbf{P}(T_x)\{t\})$.

Now that we know *what* next-step to recommend to the user, we are left with determining *when* to do so. In our study where users are tasked to drive around a series of waypoints, the system fives a next-step hint whenever the user has not moved the joystick for longer than a second. Incidentally, recognition of the release of the joystick

for a second also happens through a *detector*, though this output is not fed to the SCFG parsers since none of the steps of the stochastic grammars are to release the joystick.

5.1.1.2 *Task evaluation*

The Viterbi probability provides an intuitive measure of how well the user performed the task (section 3.3.4). The higher the Viterbi probability, the closer the steps performed were to the stochastic grammar specification. This way, should the user carried out unlikely or unnecessary steps, its associated parsing probability will be lower.

Viterbi probabilities are not, by themselves, a good evaluation metric. The reasons for this are twofold. Firstly, Viterbi probabilities cannot be compared against different tasks, as the maximum Viterbi probability depends on each grammar specification. Note that since we are using scaled Viterbi probabilities, we can compare probabilities within a task. Secondly, SCFGs do not track time, just sequences of terminals. As a result, a very slowly performed task with no missteps may yield a higher Viterbi probability than a quickly performed task with a few missteps.

In order to address the first disadvantage, we compare the user Viterbi probability against the Viterbi probability of an expert who performed the same task. This way, Viterbi probabilities are normalised and may be compared across different tasks.

In our study, users have a maximum time span to complete each task, thus we can take the number of steps performed as a proxy metric for the task speed. As with the Viterbi probabilities the maximum number of steps is task dependent, so we compare it against the number of steps of an expert.

Thus, the final normalised score is defined as follows:

$$\text{normalised score} = \frac{\bar{\nu}_{\text{user}} \times n_{\text{user}}}{\bar{\nu}_{\text{expert}} \times n_{\text{expert}}} \tag{5.1}$$

where $\bar{\nu}$ is the scaled Viterbi probability and n the number of steps completed in the allocated time. A normalised score of 1 implies the same performance as the expert. n could be replaced in the equation above by the average speed of the wheelchair if we wanted to compare against tasks with different durations.

5.1.2 A robotic tutor for wheelchair users

Now that we have presented a possible implementation of an ITS based on SCFGs, we will apply this framework to the companion for power-wheelchair users scenario. We will

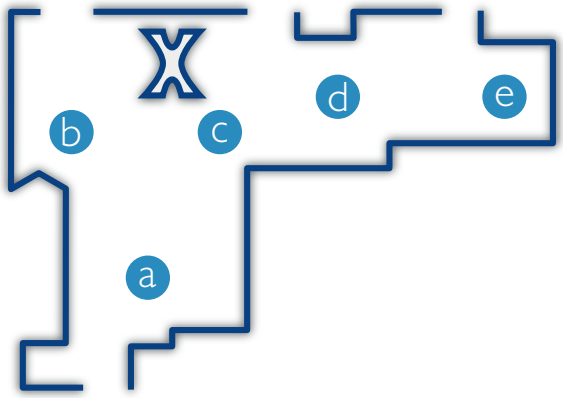


Figure 5.2: Map of the driving environment for the robotic tutor study showing the 5 waypoints (a–e). The X in the middle of the map is an obstacle for participants.

also describe a pilot study of our robotic tutor system with 18 participants and report on its results.

5.1.2.1 *Tasks specification*

The tasks of the study consisted in driving through a series of pre-determined waypoints in a limited time span. A map of the driving environment with the five waypoints is shown in fig. 5.2.

We devised three tasks as waypoint patterns for the users to drive through. Each task was represented by a different stochastic grammar (as listed in table 5.1). The grammars deal with recursion and spurious inputs using the techniques already presented in section 3.5.6, particularly the use of *skip rules* —which use the K non-terminal— to allow the parser to accept user mistakes. Moreover, participants were expected to repeat the task as many times as they could in three minutes. Consequently, the stochastic grammars have extra rules to deal with recursion (eg. S→SS).

Participants did not have access to the grammar specifications of the tasks. Instead, they were read a textual description of each of the tasks grammars. Listed below are the textual descriptions of each stochastic grammar:

TASK A "Do a triangle starting at a, then go to b, and c, and finally a. And then another triangle in the other direction. That is: first c, next b, and lastly a."

TASK B "Start at e, go to d, and then back to e. Next go to c, and then back to e. Finally go to b, and then back to e."

TASK C "Start at e, then go to d, next c, and a. Then do two triangles. That is: b, c, a, b, and c; and then go to d."

	TASK A	TASK B	TASK C
NON-TERMINALS: $\mathcal{N}$ [$\mathcal{S}$]	{A, B, C, D, E, X, Y, K, S}	{A, B, C, D, E, X, Y, Z, K, S}	{A, B, C, D, E, X, Y, K, S}
AXIOM: $\mathcal{S}$	⟶ S ⟵		
TERMINALS: $\mathcal{T}$	⟶ a, b, c, d, e ⟵		
COMMON STOCHASTIC RULES: $\mathcal{R}$ [$\mathcal{P}$]	{A⟶ a [0.7], A⟶ K [0.3], B⟶ b [0.7], B⟶ K [0.3], C⟶ c [0.7], C⟶ K [0.3], D⟶ d [0.7], D⟶ K [0.3], E⟶ e [0.7], E⟶ K [0.3], K⟶ a [0.2], K⟶ b [0.2], K⟶ c [0.2], K⟶ d [0.2], K⟶ e [0.2] }		
UNIQUE STOCHASTIC RULES: $\mathcal{R}$ [$\mathcal{P}$]	{S⟶ SS [0.5], S⟶ K [0.25], S⟶ XY [0.25], X⟶ ABC [0.5], X⟶ K [0.5], Y⟶ ACB [0.5], Y⟶ K [0.5]}	{S⟶ SS [0.5], S⟶ K [0.25], S⟶ XYZ [0.25], X⟶ ED [0.5], X⟶ K [0.5], Y⟶ EDCD [0.5], Y⟶ K [0.5], Z⟶ K[0.5], Z⟶ EDCBCD [0.5]}	{S⟶ SS [0.5], S⟶ K [0.25], S⟶ XYYD [0.25], X⟶ EDC [0.5], X⟶ K [0.5], Y⟶ ABC [0.5], Y⟶ K [0.5] }

Table 5.1: Grammar definitions for each of the three wheelchair driving tasks.

5.1.2.2 *Robots*

We combined NAO and the Assistive Robotic Transport for Adults (ARTA) wheelchair as depicted in fig. 5.3. NAO is described in appendix R.6 whereas ARTA is introduced in appendix R.2.

We settled on a robotic tutor over a virtual tutor for several reasons. Firstly, as Fasola and Mataric (2013) reports, robots are more persuasive than virtual avatars. Secondly, a physical robot can point in the direction the user has to go. Finally, we intended that our work would also be applicable to children with disabilities, and children really enjoy driving with NAO as we verified in our wheelchair sidekick trials back in section 4.1.2.

5.1.2.3 *Robotic tutor framework implementation*

The software that controlled the robotic tutor was an implementation of the SCFG-based framework for ITSs described in the previous section. We combine this framework, with the dancing NAO architecture presented in section 3.5 —a visual overview of the robotic tutor framework will be presented at the end of the chapter in fig. 5.12b. The modules to control and navigate the wheelchair are identical to those described in section 4.1. As in our previous systems, all the communication was handled by the Robot Operating System (ROS) which we describe in appendix R.1.

We initialise one detector per waypoint. Each detector takes the pose generated by the Adaptive Monte-Carlo Localisation (AMCL) module and outputs the probability of the user being at the waypoint. We additionally added another detector which takes the

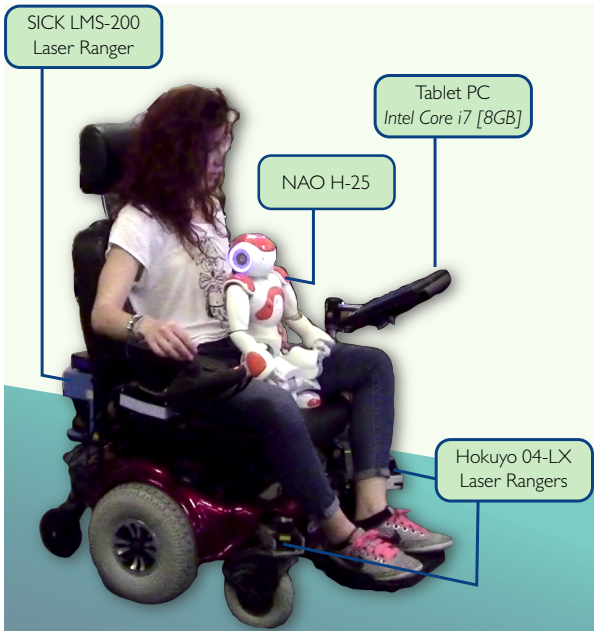


Figure 5.3: Hardware components of our robotic tutoring system.

joystick position and triggers a signal when the user has not given any joystick input in a second. All detectors are executed continuously and concurrently.

The outputs of the waypoint detectors, though not that of the joystick release detector, were fed to the currently selected SCFG parser —recall there is a parser per task in the activity. The parser then calculated the next most likely step and if the joystick release detector triggered then NAO would hint to the user what the next step was.

At the end of a task, the parser would send the normalised score to the tutor module which would in turn determine whether the user should repeat the task or progress onto the next one. We heuristically set the tutor to repeat the task if the normalised score was lower than 0.7.

After evaluation, the system waited until the user was ready to start again (indicated by touching NAO's head), at which point all detectors and parsers would be reset to analyse the user performance and provide hints for the upcoming task.

5.1.2.4 *Comparison with the wheelchair sidekick experiments*

There are many commonalities between this robotic tutor and the path driving experiments we carried out with the wheelchair sidekick system in section 4.1.3. In both cases NAO accompanies and gives directions to the user whilst she drives a smart power-wheelchair.

There are however some key differences. Most strikingly, we used an adult wheelchair rather than a children's wheelchair, as we expected that adult participants would be more comfortable whilst driving. As a consequence of this, NAO needs to be placed in the user's lap since ARTA, unlike ARTY, is not equipped with a hook from where to hang NAO. In any case, it is possible to run the same code in Assistive Robotic Transport for Youngsters (ARTY) with only minimal changes.

Another difference is that for this study we only consider waypoints rather paths, which offers two advantages: detecting waypoints is simpler and more robust than detecting a full path; and, more importantly, it gives users more freedom to drive however they chose between waypoints, potentially driving sub-optimally but also leaving more space for learning.

The focus of the study differs too. Whereas previously we wanted to compare NAO with other driving aids, here we are interested in whether NAO and ARTA could be used as a robotic tutor. Consequently, NAO speaks and points at the same time, thus addressing one of the short-comings of the previous study, ie. the lack of multi-modality in NAO.

5.1.2.5 *Set-up*

We designed two conditions for the study, the baseline and driving with the robotic tutor. In both cases, users were supposed to drive following the patterns of Task A, B and C (table 5.1) in that order.

For the baseline, a researcher read the task description to the user. Then, when the user indicated she was ready, the researcher started recording the data and the participant had three minutes to complete the task. Once the three minutes had elapsed the researcher would indicate this to the user and ask her whether she wanted to progress or repeat the task —unless the user had already repeated the current task in which case she would automatically progress to the next task. This process was repeated until the user had progressed through the three tasks. Participants in this condition could not request either next-step hints nor performance evaluations.

The robotic tutor condition was very similar but the robot would describe on its own the task and ask the user to start the timer by touching its head. The robot would also give hints if the user released the joystick for a second. And, at the end of the task, it was the robotic tutor which decided whether the participant progressed onto the next task. This is in contrast to the baseline condition where it was the user who decided to progress or repeat.

Since we were interested in the different attitudes about the robotic tutor, we assigned two participants to the robotic tutor condition for every participant in the baseline.

In both conditions, users had to fill in a questionnaire after they had completed the tasks. Most of these questions requested that the user rated her agreement with a statement in a 5-point Likert scale. A few questions however were binary and some of them had another question in Likert scale as a follow-up. In our analysis, we denote as N/A when the user did not answer affirmatively to the first binary question. All participants, regardless of condition, were asked: their gender, age, whether they had worked with robots or driven a wheelchair before as well as the following questions:

- The driving tasks were difficult (*difficult*).
- I understood the initial instructions for each task (*understood instructions*).

Additionally, users in the baseline were asked:

- I chose to repeat a task [Y/N] $\xrightarrow{\text{if yes}}$ I performed better the second time around (*second better*).
- I would have preferred a driving tutor for theses tasks [Y/N].

Users in the tutor group were asked instead:

- NAO asked me to repeat a task [Y/N] $\xrightarrow{\text{if yes}}$ I performed better the second time around (*second better*).
- NAO gave me hints whilst I was driving [Y/N] $\xrightarrow{\text{if yes}}$ NAO's hints were useful (*useful hints*).
- NAO was a useful driving tutor (*useful tutor*).
- I like having NAO with me when I was driving the wheelchair (*liked NAO*).

5.1.2.6 *Results*

A total of 18 people (7 females) took part in the study. Participants ages varied from 19 to 35. 9 people reported having worked with robots before and 5 to have driven a power wheelchair previously. 12 people performed the experiment with NAO as the wheelchair tutor (of which 6 females) and 6 drove by themselves as part of the baseline (1 female). For the baseline group, 3 people had worked with robots and 2 driven a

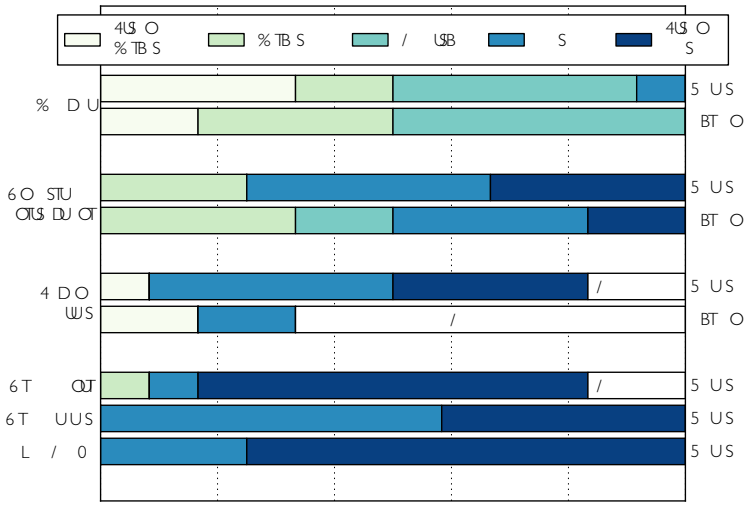


Figure 5.4: User study's questionnaire responses for users in both categories.

power wheelchair, whereas 6 people had worked with robots and 3 driven a wheelchair in the robotic tutor group. None of the participants was a roboticist.

Participants answers to the questionnaires are shown in fig. 5.4. Two-tailed Mann Whitney U tests revealed no statistically significant differences between participants answers to *difficult*, *understood instructions* and *second better*. However, the proportion of answers to *second better* for baseline is much smaller since —as we will analyse in more detail later— participants in the baseline repeated the tasks significantly less than participants with the tutor.

Regarding the subjective assessment of the robotic tutor, 75% of users agreed or strongly agreed that the robot gave *useful hints*, and everybody agreed or strongly agreed that NAO was a *useful tutor* and that they *liked* driving with it. When asked if they would have preferred a robotic tutor for the tasks, 5 out of 6 participants in the baseline condition answered negatively.

Looking at the objective metrics, the data suggests participants in the baseline group obtained a higher normalised score averaged across tasks, median: 0.91, interquartile range (IQR): 0.26, than participants with the robotic tutor (median: 0.64, IQR: 0.13); still this result was not found to be statistically significant ($p = 0.067$). Similarly, the Viterbi score averaged across tasks for participants in the baseline was significantly higher ($u = 13$, $p = 0.035$) than for participants with the robotic tutor: the median Viterbi score averaged across tasks was: 0.150 (IQR: 0.0190) for the baseline in contrast to 0.134 (IQR:

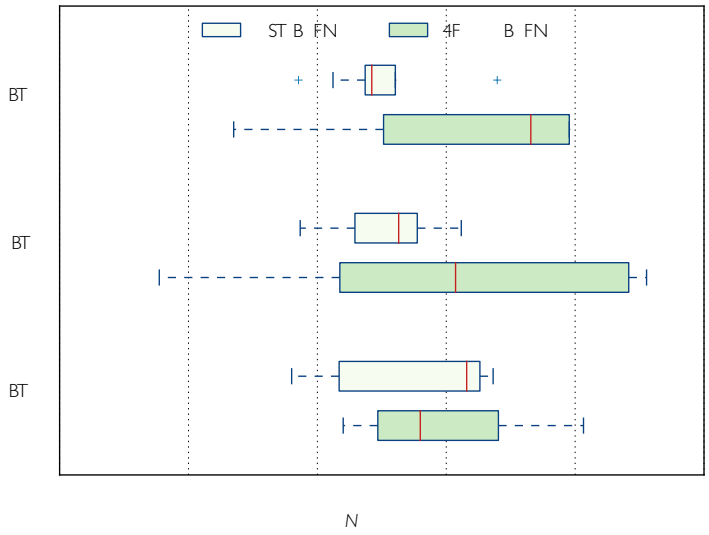


Figure 5.5: Boxplots showing driving scores for each user who was instructed to repeat a path by the robotic tutor.

0.020) for the robotic tutor group. For reference, the average Viterbi score of the expert was: 0.166.

As we mentioned earlier there was a statistically significant difference in the number of task repetitions between participants with the robotic tutor (median: 2.0, IQR: 1.0) and the baseline (median: 0.0, IQR: 1.5). For this test, the u statistic was found to be 15 and the p-value 0.044.

For those participants in the robotic tutor category who did repeat their task, we found that their second attempt had a significantly higher normalised score (median: 0.63, IQR: 0.10, N = 26) than their first (median: 0.49, IQR: 0.32, N = 26) with u=214.5 and p=0.024. Figure 5.5 showcases these differences. No such difference was found for the baseline, though the number of repetitions for that group was very small (N=5).

On the other hand, there were no statistically significant differences in the number of hints the tutor gave between the first and second attempts of a task in the robotic tutor group.

Furthermore, a Friedman test revealed no statistically significant differences in either score or hints given by the tutor across the three different tasks.

We were interested in whether the number of hints the tutor gave or the final score affected the way users rated their interactions with the robot. The Spearman correlation between the number of hints the users received and their rating of the usefulness of the hints was not statistically significant ($p > 0.10$). On the other hand, the data suggest

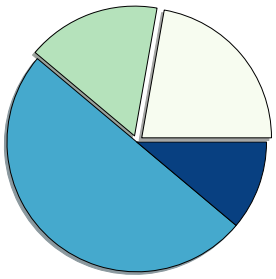


Figure 5.6: Correct and incorrect evaluations made by baseline users of their own task performance.

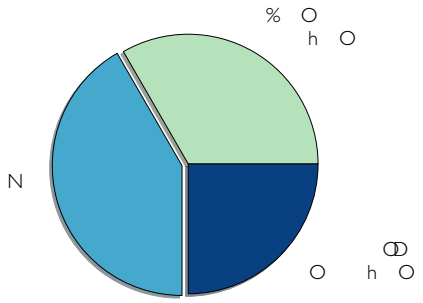


Figure 5.7: Classification of participants usage of the robotic tutor hints.

there may be a negative correlation ($\rho = -0.51$) between the average score and the robot being a useful tutor (p=0.087, N=12).

Baseline users were given the choice of repeating the task or progressing to the next one. Comparing their choices to what the robotic tutor would have decided shows that users repeated needlessly a task 17% of the time and advanced on to the next task incorrectly 22% of the time (fig. 5.6).

Finally, during the study we observed some participants never requested hints from the robotic tutor whereas other participants requested a hint at every step of the task. Figure 5.7 quantifies this behaviours: 33% of users did not request hints from NAO, whereas 25% of users requested a hint for more than 85% of the steps they completed.

5.1.3 Discussion

The questionnaires made it clear that participants in the robotic tutor condition enjoyed driving with the NAO and thought it was useful. This was in contrast to participants in the baseline, only one of whom would have preferred to have a driving tutor alongside her. We postulate this preference may be explained by the fact the baseline participants had not seen the robotic tutor in action and could not imagine its abilities.

Participants in the baseline group obtained a significantly higher Viterbi score, suggesting they followed the instructors better than participants who drove with the tutor. The small size of the baseline group makes it difficult to determine whether this finding is generalisable, but it could be that people pay closer attention when driving on their own than when they can ask for hints to someone else.

Considering only those who participated with the robotic tutor, we found that their scores were significantly higher after repeating a task. Despite not being clear if the tutor's hints had any influence on the higher score for the second attempt (we found no differences in the number of hints between original and repeated tasks), NAO's ability to evaluate users did translate into better task performance. The scores were not significantly different between the three tasks, suggesting that participants got better at a task, rather than at driving the wheelchair. It would be interesting to see if this holds true over more long-term sessions.

The data suggests that there may be a negative correlation between the perceived usefulness of the robotic tutor and the final score of a user. This may be due to users with more driving difficulties scoring lower which made them appreciate the robotic tutor more than those with high scores.

We identified two patterns regarding the hints requested to the robot. Firstly, 33% of participants in the robotic tutor group did not request any hint from the robot. This may be explained by users not understanding the robot when it announced its hinting capabilities; perhaps NAO should remind users they may request a hint if their progress is not adequate. We also identified a subset of users —25% of all participants in the tutor group— who requested hints for practically every step in of the tasks. In this case, the solution could be to have a limit on the number of hints per task and program NAO to instruct them when they are reaching this limit. By refusing to help, participants may actually learn more.

The number of task repetitions was significantly higher for the tutor group than for the baseline. Notwithstanding baseline participants higher score, a detailed analysis of their choices reveals that 39% made the wrong choice and would have benefited from either repeating the task —if their score was too low— or from progressing to the next task —when their score was high enough. This finding further highlights the importance of the evaluation step for ITSs.

Some users reported problems understanding NAO's speech in the comments section of the questionnaire. Indeed, a difference between the baseline and tutor groups was that in the baseline it was the researcher who read the instructions and this may help explain the higher score of the baseline group. Other users mentioned that they would have liked reassurance from the tutor when they were performing well, this could be added to the current system by checking the slope of the normalised score.

No user reported being confused by the robot's instructions and this translated into very high agreement with NAO being a *useful tutor*. This is a marked improvement over the appraisal of NAO as a driving companion where network delays and a heuristic al-

gorithm sometimes caused NAO to give contradictory instructions (section 4.1.4). We attribute this improvement to the use of a robust and principled algorithm (SCFGs) and having users request hints themselves.

This study represents a successful proof-of-concept of the architecture we have developed across this thesis (sections 3.5 and 5.1.1). The system uses SCFG properties to provide hints about what step should perform next, and it uses a HAMMER-like architecture to detect which steps has the user already performed. The user model is effectively the normalised scores of the users across the paths and this was used to determine if participants should progress on to the next task or repeat the current task.

In the next section, we focus on modelling the user preferences rather than the user abilities. And even if the scenario is quite different —synchronisation of musical tasks— many of the algorithmic components we use to model the user preferences are the same as those we used for the robotic tutor.

5.2 MODELLING USERS PREFERENCES: THE SYNCHRONISED GRAMMARS FRAMEWORK

Everybody enjoys music. It is not surprising then that there is intense research in the field of human-robot musical collaboration (Cicconet et al., 2013; Crick et al., 2006; Hoffman and Weinberg, 2010; Kitani and Koike, 2010; Michalowski et al., 2007; Quick and Hudak, 2013). However, there are many challenges to having a robot playing music alongside a human: there are small-scale problems like actuation speed and accuracy, as well as large-scale issues such as the progression of the musical pattern. There is also the question of adaptation, how can a robot adapt to each user's musical taste? In this section, we focus on tackling two of these problems: *synchronisation with the musical pattern* and *adaptation to the user's musical taste*.

We tackle the *synchronisation* problem by making use of Stochastic Context-Free Grammars (SCFGs). SCFGs can account for the underlying structure of musical compositions by representing the music primitives as the base symbols of the grammar (terminals) and the musical structure as a set of hierarchical rules. We attempt to solve the *adaptation to the user* problem by defining and updating a model of the user's musical taste.

That is the approach we follow. Using a tangible music interface (Reactable) we designed a human-robot collaboration system where a user and a robot create music together. In this set-up, the user is the conductor and is in charge of the melody while the robot (Baxter) produces the drums accompaniment. Having Baxter in the interaction

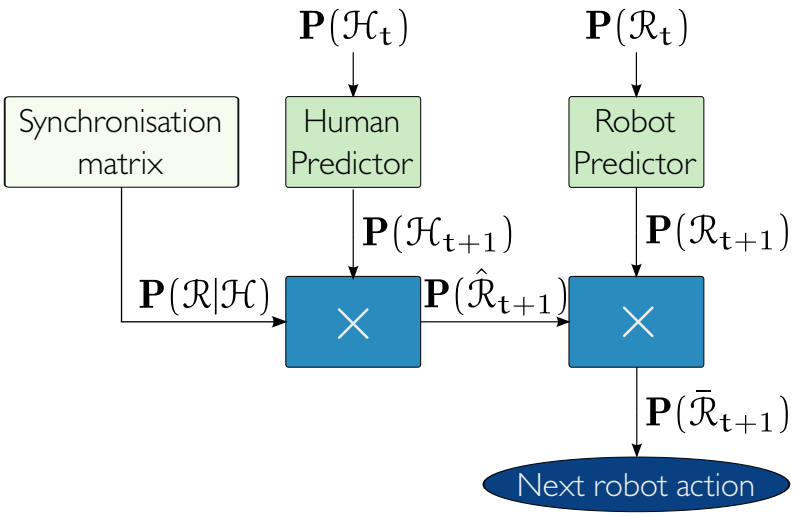


Figure 5.8: Flowchart representation of the synchronised grammars framework. $\mathbf{P}(\mathcal{H}_t)$ and $\mathbf{P}(\mathcal{R}_t)$ are the input probability distributions for the human and the robot predictors respectively, similarly $\mathbf{P}(\mathcal{H}_{t+1})$ and $\mathbf{P}(\mathcal{R}_{t+1})$ are the expected terminal probability distributions for the human and robot. $\mathbf{P}(\mathcal{R}|\mathcal{H})$ is the conditional probability between the terminals of the robot and the human. $\mathbf{P}(\hat{\mathcal{R}}_{t+1})$ is the influencing robot probability distribution. Finally, $\mathbf{P}(\bar{\mathcal{R}}_{t+1})$ is the distribution from which the robot draws its next action.

loop ensures that our algorithms are robust enough to deal with unexpected errors by the robot.

By using the synchronised grammars framework —which we will present next— the robot can predict the next most likely action of the user and act accordingly. Our framework works with probability distributions over all musical primitives (terminals). This is essential as Baxter draws its next action from the computed probability distribution rather than selecting the action with highest probability. Consequently, our system may choose an unexpected action, but we argue this adds a creativity aspect to the collaboration.

5.2.1 Synchronised grammars framework

We will now present our framework to synchronise two independent SCFG parsers. Specifically, we are interested in using the predictions of one parser to influence a second one. A summarised version of our algorithm is shown in fig. 5.8.

We denote $\mathcal{H}$ and $\mathcal{R}$ as the set of terminals of the first and second grammars respectively. Note that in our forthcoming study $\mathcal{H}$ represents the terminals of the human performer, and $\mathcal{R}$ those of the robot performer; despite this, the analysis in this section can

be applied to any two SCFG parsers. Following the next input prediction algorithm in section 3.3.5, $\mathbf{P}(\mathcal{H}_t)$ and $\mathbf{P}(\mathcal{R}_t)$ represent the terminal probability distributions at step t for the first and second parsers.

By feeding the terminals to the parser and performing a full parsing step (i.e. executing *scan*, *complete* and *predict*) as well as predicting the next input, we can obtain the expected terminal probability distributions for each parser: $\mathbf{P}(\mathcal{H}_{t+1})$ and $\mathbf{P}(\mathcal{R}_{t+1})$.

We are looking for a way of influencing the second grammar. Therefore, we need to transform the first expected terminal probability distribution into a probability distribution over the terminals of the second grammar, which we will call the influence probability distribution, $\mathbf{P}(\hat{\mathcal{R}}_t)$. To achieve this we use a matrix whose elements denote the conditional probability between terminals in the first and second grammars. We designate this matrix as the synchronisation matrix, $\mathbf{S}(\mathcal{H}, \mathcal{R})$:

$$\mathbf{S}(\mathcal{H}, \mathcal{R}) = \begin{bmatrix} P(r_1|h_1) & P(r_1|h_2) & \cdots & P(r_1|h_n) \\ P(r_2|h_1) & P(r_2|h_2) & \cdots & P(r_2|h_n) \\ \vdots & \vdots & \ddots & \vdots \\ P(r_m|h_1) & P(r_m|h_2) & \cdots & P(r_m|h_n) \end{bmatrix} \tag{5.2}$$

$$\text{with } \mathcal{H} = \{h_{1\ldots n}\} \text{ and } \mathcal{R} = \{r_{1\ldots m}\}$$

With the synchronisation matrix we can convert the predicted terminal probability distribution of the first grammar into the influence probability distribution using the law of total probability. This is equivalent to taking the dot product of the synchronisation matrix and the expected terminal probability distribution:

$$\mathbf{P}(\hat{\mathcal{R}}_{t+1}) = \text{normalise}\left(\mathbf{S}(\mathcal{H}, \mathcal{R}) \cdot \mathbf{P}(\mathcal{H}_{t+1})\right) \tag{5.3}$$

The final step is to obtain the final probability distribution for the second grammar from both the influencing grammar and the terminal probability distribution for the second grammar. Though any method to combine two distributions into a mixture could work here, we have chosen element-wise multiplication and normalisation since: it does not require any extra parameters, it favours elements which have high probabilities in both distributions and it punishes elements with a low probability in either distribution:

$$\mathbf{P}(\bar{\mathcal{R}}_{t+1}) = \text{normalise}\left(\mathbf{P}(\mathcal{R}_{t+1}) \odot \mathbf{P}(\hat{\mathcal{R}}_{t+1})\right) \tag{5.4}$$

Multiplying two probability distributions as above implies both of the factors have the same relative importance. Accordingly, one has to take care to define the second grammar

in a way allows for it to be influenced. If a grammar determines its terminals with very high confidence at all times, it will not be influenced very much by other parsers.

The synchronisation matrix is the structure we use to introduce adaptivity in the framework. By modifying the elements of this matrix we can change the resulting influence probability distribution according to the preferences of each user. In other words, the synchronisation matrix represents the model of the user preferences.

Though there are several ways of altering the synchronisation matrix, we chose a punishing method whereby the user can indicate it does not like the currently selected terminals r_i, h_j. In such instances, we divide $\mathbf{S}(\mathcal{H},\mathcal{R})\{r_i, h_j\}$ by a constant factor (heuristically set to *2.0* in our study) and renormalise the i^{th} row to add up to 1 again. This way we effectively increase the probability of all other terminals in $\mathcal{R}$ with respect to h_j.

Note that our algorithm outputs a probability distribution over the terminals of the second grammar: $\mathbf{P}(\bar{\mathcal{R}}_{t+1})$. Though taking the terminal with the highest probability as the next input would work, we choose to draw a random terminal according to the $\mathbf{P}(\bar{\mathcal{R}}_{t+1})$ distribution.

5.2.2 Synthetic analysis of the synchronised grammars framework

We now synthetically analyse the synchronisation and adaptivity of our framework. Before that, however, we will verify that the parsing probability of randomly generated grammars does not change significantly.

For these experiments we use two different grammars: $\mathcal{G}_{\mathcal{H}}$ and $\mathcal{G}_{\mathcal{R}}$. See table 5.2 for their respective definition. $\mathcal{G}_{\mathcal{H}}$ encodes a sequence of terminals: {a, b, c^n, d^n, b, a} and it accepts repetition of a given terminal any number of times (with rules like A→a and A→AA). To add robustness, $\mathcal{G}_{\mathcal{H}}$ can also *skip* any terminal through the use of rules such as A→K, though this is left as a low-probability option. Meanwhile, $\mathcal{G}_{\mathcal{R}}$ is set to chose a random terminal with equal probability.

First, we verify that there are no statistically significant differences between two sets of sequences randomly generated from the same grammar. To test this we generated two sets of a thousand independent sequences each with 60 characters spawned from $\mathcal{G}_{\mathcal{H}}$ and obtained their Viterbi parsing probability against $\mathcal{G}_{\mathcal{H}}$. By Viterbi parsing probability, we are referring to the *scaled Viterbi probability* introduced back in section 3.3.4. Spawning characters is achieved by iteratively performing a parsing step (*scan*, *complete* and *predict*); obtaining the expected probability distribution, $\mathbf{P}(\mathcal{T}_i)$; and drawing a random character from $\mathbf{P}(\mathcal{T}_i)$.

	USER GRAMMAR: $\mathcal{G}_{\mathcal{H}}$	ROBOT GRAMMAR: $\mathcal{G}_{\mathcal{R}}$
NON-TERMINALS & AXIOM: $\mathcal{N}$ [$\mathcal{S}$]	{A, B, C, D, X, S, K} [S]	{A, B, C, D, S} [S]
TERMINALS: $\mathcal{T}$	{a, b, c, d}	{a, b, c, d}
STOCHASTIC RULES: $\mathcal{R}$ [$\mathcal{P}$]	{S→ABXBA [1.00], X→CD [0.50], X→CXD [0.50], A→a [0.45], A→AA [0.45], A→K [0.10], B→b [0.45], B→BB [0.45], B→K [0.10], C→c [0.45], C→CC [0.45], C→K [0.10], D→d [0.45], D→DD [0.45], D→K [0.10], K→a [0.20], K→b [0.20], K →c [0.20], K→d[0.20], K→KK [0.20] }	{S→S [0.2], S→AS [0.2], S→BS [0.2], S→CS [0.2], S→DS [0.2], A→a [0.8], A→AA [0.2], B→b [0.8], B→BB [0.2], C→c [0.8], C→CC [0.2], D→d [0.8], D→DD [0.2] }

Table 5.2: Grammar definitions for: influencing (user) grammar and base (robot) grammar.

We then performed a two-tailed Mann-Whitney U test on both sets of Viterbi probabilities and, as expected, found the differences *not* to be statistically significant. The median Viterbi parsing probability for the first set of sequences was 0.1395 and their interquartile range (IQR): 0.0292, whereas for the second set we found a median of 0.1414 and IQR of 0.0261.

5.2.2.1 *Synchronisation test*

Subsequently, we verified whether one parser can influence the state of another parser. To do so, we generated two sets of sequences from $\mathcal{G}_{\mathcal{R}}$; one of them by drawing the input terminal from the expected probability distribution as before. The other sequence was obtained by influencing $\mathcal{G}_{\mathcal{R}}$ with $\mathcal{G}_{\mathcal{H}}$ using our framework. This effectively requires spawning an independent sequence from $\mathcal{G}_{\mathcal{H}}$ which again we do by drawing randomly from the expected probability distribution. The synchronisation matrix chosen here is the identity matrix (or which is the same, $\mathcal{H} \equiv \mathcal{R}$). We then parse these sequences (generated from $\mathcal{G}_{\mathcal{R}}$) against $\mathcal{G}_{\mathcal{H}}$ and obtain their scaled Viterbi probabilities, thus measuring how much did $\mathcal{G}_{\mathcal{H}}$ influence each sequence. We expect that the higher the influence, the higher the resulting Viterbi parsing probability and check for differences using a two-tailed Mann-Whitney U test once more.

We ran this test with two sets of a thousand sequences, each sequence being 60 characters long and found the differences in Viterbi probability to be statistically significant ($p \ll 0.01$). The median Viterbi probability of the uninfluenced sequences was 0.0762 and the IQR: 0.0114, whereas for the influenced sequences the median probability was 0.1538 and the IQR: 0.0315.

This confirms our claim that, at least in synthetic environments, our framework allows the state of one parser to influence the state of another parser.

5.2.2.2 *Adaptivity test*

Finally, we tested whether the synchronisation matrix can be adapted to the user's preferences. To do so, we randomly create *preferred mappings* from the $\mathcal{H}$ terminals to the $\mathcal{R}$ terminals (eg. $\{ a \to b, b \to d, c \to c, d \to a \}$). The *preferred mapping* is meant to represent the different combinations of $\mathcal{H}$ and $\mathcal{R}$ a user would prefer. We then count the number of times the user would give negative feedback with adaptivity and without it.

Two 60 characters long sequences are generated from $\mathcal{G}_{\mathcal{R}}$ influenced by $\mathcal{G}_{\mathcal{H}}$ following our framework. The first sequence is generated by updating the synchronisation matrix (this is done by dividing the corresponding entry in the synchronisation matrix by 2 and renormalising the row, as in section 5.2.1) when the terminals do not correspond to the *preferred mapping*. For the second sequence —the control sequence— the synchronisation matrix is not updated. Note that, in both cases the synchronisation matrix is 4x4 and initialised with all its entries to 0.25, effectively representing a random mapping between $\mathcal{H}$ and $\mathcal{R}$.

As the sentences are generated, we record the number of mismatches, that is the number of times the terminals did not correspond to the *preferred mapping*.

Repeating this process a thousand times and performing a two-tailed Mann-Whitney U test reveals that sequences generated with synchronisation matrix adaptivity had fewer mismatches from the *preferred mappings* (median: 33, IQR: 6.0) compared to the control (median: 45, IQR: 5.25) and these results were statistically significant with $p \ll 0.01$.

This proves that, in a synthetic set-up, updating the synchronisation matrix leads to significantly fewer mismatches between the preference of the user and what the robot chooses to do.

5.2.3 Musical human robot collaboration study

The synchronised grammars framework can be readily applied to human-robot collaboration to generate music. In the following section we describe the pilot implementation of such a system.

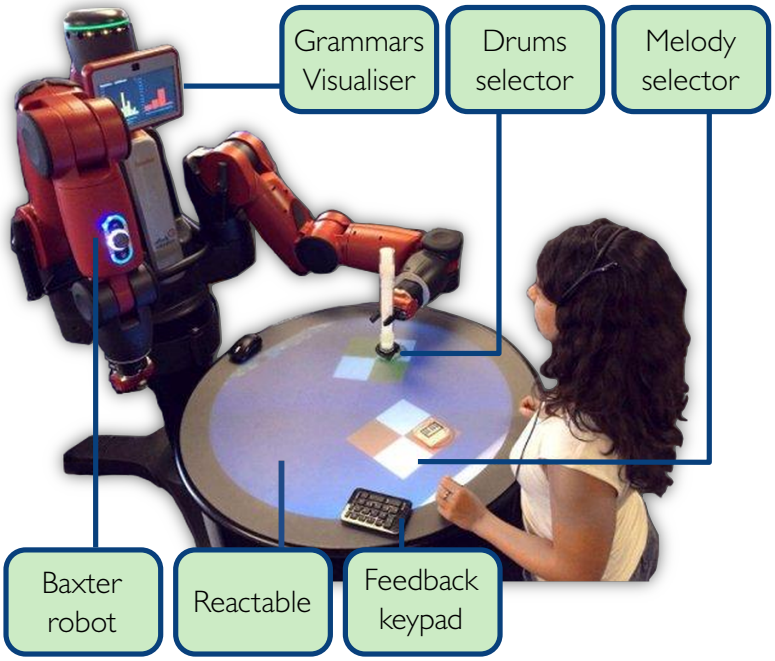


Figure 5.9: Main components of our musical collaboration system with Baxter and Reactable.

5.2.3.1 *Set-up*

Figure 5.9 shows a picture the main components of our system. At its core, we have a Reactable, the tangible music table we will describe in appendix R.5. We use a custom built application which uses the information provided by reacTIvision to drive two virtual chequerboards. Each of these chequerboards represents a musical instrument, either melody or drums. For every chequerboard there is an associated fiducial, and depending on the fiducial's position with respect to its owning chequerboard one track of music or another is played. The system is currently programmed with 4 drum patterns and 4 melody patterns. All patterns are 4 seconds long and do not change as the trials progress. Note these patterns constitute the musical primitives (or terminals) of our grammars.

Baxter is a 1.90 metres tall robot with two 7 degree-of-freedom arms, grippers on each hand and a programmable display (appendix R.4). Baxter also receives the position of the fiducial with respect to its keyboard. By showing Baxter the kinematic configuration of the positions where each of the tracks is active, we can direct Baxter to play a specific pattern with quick movements. Additionally, Baxter displays the status of the current music session on its display.

The system needs three computers, one for the Reactable, another on-board Baxter and a final one running our framework. Communication between Baxter and the main computer is done through ROS (appendix R.1) whereas the Reactable uses a bespoke

library built with Python, Unix sockets and JavaScript Object Notation (JSON) for data interchange.

The role of the Stochastic Context-Free Grammars (SCFGs) is to represent the structure of the music about to be played. For this study we defined the grammars by hand (table 5.2 shows the user and robot grammars respectively), but it is also possible to generate these grammars from expert demonstrations using the method presented by Lee et al. (2013).

The initial synchronisation matrix was obtained by asking a participant to generate music with our system during 5 minutes and then counting the number of co-occurrences for each user and robot terminal. The participant controlled both the melody and the drums and thus there was no robot involvement. The resulting synchronisation matrix was the starting matrix for all participants and is shown below:

$$\begin{bmatrix} 0.37 & 0.05 & 0.27 & 0.31 \\ 0.23 & 0.04 & 0.46 & 0.27 \\ 0.03 & 0.08 & 0.83 & 0.06 \\ 0.08 & 0.82 & 0.06 & 0.04 \end{bmatrix} \tag{5.5}$$

Eight participants, two of them female, aged 21–34 took part in our pilot study. Each participant had 4 trials to create 3 minutes of music with Baxter. The first of these trials was discarded as training. Participants could also indicate the system the track Baxter had selected was not an appropriate match for their own choice of track. This assessment was made by pressing the Enter key on the feedback keypad situated next to them. Note the changes made to the synchronisation matrix were kept across trials (except for the training trial).

At the end of the trials, participants were asked to complete a questionnaire with the following questions in a 5-point Likert scale:

- Performing music with Baxter was difficult (*difficult*).
- Performing music with Baxter was engaging (*engaging*).
- Baxter's actions conformed to my expectations (*conformed to expectations*).
- Baxter became a better accompanist as trials progressed (*progressed*).
- Baxter reacted quickly to my changes in the melody (*reacted quickly*).

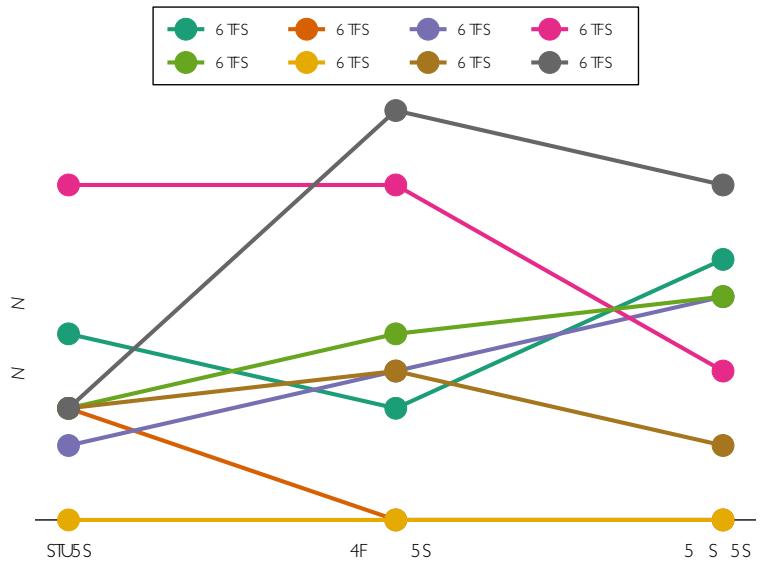


Figure 5.10: Number of times users gave feedback to the musical collaboration system by participant.

5.2.3.2 *Results*

Figure 5.10 shows the amount of negative feedback per participant across the three trials. It can be seen in the figure that 4 participants (S2, S4, S6 and S7) decreased the overall amount of negative feedback. In contrast, participants S1, S3, S5 and S8 increased the overall amount of negative feedback. S6 did not provide feedback at all.

The final synchronisation matrices from two participants are shown below. We remark that both matrices are different from each other as well as from the original synchronisation matrix.

$$\begin{bmatrix} 0.53 & 0.06 & 0.19 & 0.22 \\ 0.37 & 0.01 & 0.18 & 0.44 \\ 0.01 & 0.08 & 0.80 & 0.11 \\ 0.05 & 0.54 & 0.32 & 0.09 \end{bmatrix} \quad (5.6) \qquad \begin{bmatrix} 0.18 & 0.01 & 0.52 & 0.29 \\ 0.24 & 0.00 & 0.47 & 0.28 \\ 0.01 & 0.08 & 0.89 & 0.02 \\ 0.19 & 0.47 & 0.03 & 0.31 \end{bmatrix} \quad (5.7)$$

Figure 5.11 shows the results of the questionnaires. 3 people agreed the robot *reacted quickly* while 3 people disagreed or strongly disagreed with the statement. 6 people agreed the robot improved as trials *progressed*. 2 people disagreed that the robot *conformed to their expectations* whereas 3 agreed that was the case. 6 people agreed or strongly agreed the task was engaging. 3 people were neutral about whether they had found the task *difficult*, the rest disagreed or strongly disagreed.

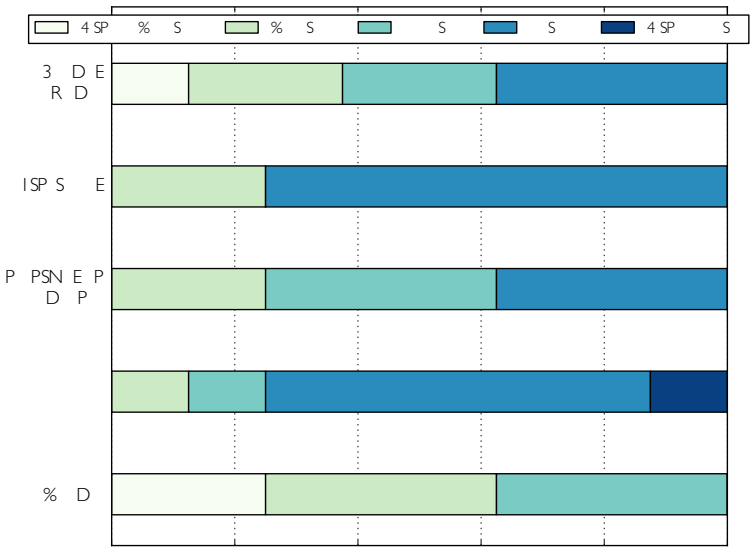


Figure 5.11: Results of the questionnaire filled in by participants after the musical collaboration trials.

5.2.4 Discussion

We posit the differences in the amount of negative feedback stem from the fact that every participant has a different expectation of the system's learning curve. This is substantiated by the questionnaires where the answers to whether the robot *conformed to expectations* were evenly spread. It is possible as well that different users require different updating constants in the synchronisation matrix.

The sample final synchronisation matrices confirm that our framework can adapt to the preferences of different users. For instance, the first robot terminal in eq. (5.6) gives most of the probability mass to the first human terminal, whereas eq. (5.7) gives it to the third human terminal. Moreover, observe how these preferences are not present in the original synchronisation matrix: eq. (5.5).

With respect to the questionnaires, we were not surprised to find that most users agreed the task was engaging since, as we mentioned in the beginning of this section, most people enjoy music. More interesting was the fact that most people (6 out of 8 participants) felt the robot had become a better accompanist as the trials progressed. We remark that this is a similar ratio to the number of people which decreased their overall negative feedback (4 out of 8). This similarity encourages us to carry out further experimentation to establish statistical significance.

There is no clear consensus with respect to whether the robot reacted quickly to user's actions and our data shows that there were network-induced delays in Baxter's actions. This may also explain the divergent answers to whether the robot conformed to the user's expectations.

5.3 CONCLUSION

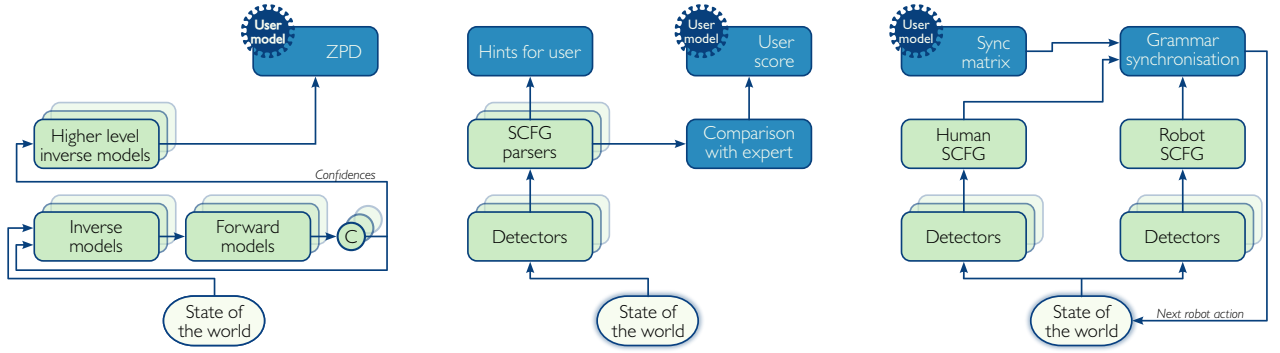
We have presented two frameworks to model the abilities and preferences of users. The first one, the robotic tutoring framework, is an implementation of an ITS and allows the robot to propose tasks, evaluate them and give hints to the user. The use of SCFGs makes the robotic tutoring framework robust to noise and unpredictable user steps. The framework recognises the low-level actions via *detectors* and then feeds the output to a SCFG parser, just as we did for dancing NAO in section 3.5. The output of the parser allows the tutor to determine what is the next step the user should perform and to evaluate the user's overall task performance.

A study with two conditions, baseline and robotic tutor, has shown that users who were told to repeat a task, mostly improved upon doing so. Our results also highlight the difficulty of evaluating your own performance when you are not an expert in the task, as 39% of baseline participants decisions when deciding whether to progress onto the next task were incorrect. Moreover, the results of the questionnaires show that *all* participants agreed or strongly agreed that the robotic tutor was useful and that they liked driving with NAO by their side.

The second framework we have presented allows to account for the values of another parser. This can be useful in many scenarios as it provides a formal method to combine the constraints given by the structure of one task and the needs of another independent —but simultaneous— task. With our framework it is possible to incorporate external feedback or ignore it due to task constraints. The synchronised grammars framework further allows for personalisation. This is achieved through the synchronisation matrix. The synchronisation matrix is the link between the two parsers and changing it can give rise to a wide array of behaviours.

Our synthetic experiments confirm both the ability to let one parser influence the other as well as the adaptivity properties of the synchronisation matrix.

All the algorithms we used are probabilistic, which lets the synchronised grammars framework provide a natural approach (by randomly drawing from a probability distribution) for a robot to play music with variations which reflect the user's preference. This



(a) Using HAMMER to model the user's ZPD, ie. the tasks which she can potentially accomplish —adapted from Demiris (2009).

(b) Modelling the user abilities following the robotic tutor framework presented in section 5.1.

(c) Synchronisation of two grammars using the synchronised grammars framework introduced in section 5.2.

Figure 5.12: Visual comparison between the three different architectures for user modelling we have discussed in this thesis.

is relevant to a number of domains. Amongst them is music, where small variations are vital for the music not to sound artificial.

Music usually has an internal structure and users have very varied tastes. With its ability to synchronise to external patterns and adapt to users, our framework fits well with musical human-robot collaboration. To test this, we built a system with a tangible music table and a two-armed robot. From the results it was observed that the robot could be influenced by the music the users were generating as well as to adapt to feedback by users.

Specifically, the results of our pilot study suggest that, given enough time, the synchronisation matrix will converge to the preferences of the user. This was corroborated by the user's questionnaires —where 6 out of 8 participants agreed the robot had improved across the trials— as well as the results from the synthetic experiments —where the mismatch rate was significantly lower with an adaptive synchronisation matrix.

5.3.1 Comparison between user modelling frameworks

At this point, it may seem that the two frameworks we introduced in this chapter are very different. Though they have been applied to diverse domains, they share more than the use of SCFGs, as we will discuss now. Figure 5.12 shows the a diagrammatic representation of the two frameworks we introduced as well as an adaptation of how Demiris (2009) described the HAMMER architecture for user modelling.

Recall that in section 3.5.5 we characterised a detector as a combination of an inverse model, a forward model and a confidence function. Moreover the SCFG parser in the robotic tutor framework could in fact be interpreted as higher level inverse models. Finally, the user score is a metric of how well did the user perform a task and consequently could be used to compute the Zone of Proximal Development (the Zone of Proximal Development (ZPD) corresponds to the set of tasks that the user has the potential to accomplish but has not yet). In conclusion, fig. 5.12b are essentially equivalent can be interpreted as an operationalisation of fig. 5.12a. Hence, we interpret the robotic tutor framework as being an specialisation of the HAMMER architecture.

The synchronised grammars framework is also very similar. This framework is the combination of two sets of detectors and SCFG parsers with the algorithmic machinery to synchronise them. The synchronised grammars framework is then another operationalisation of the HAMMER architecture.

It is not surprising that these three frameworks are similar, they are solving many of the same problems as we have summarised in table 5.3. Firstly, in any setting it is necessary to recognise the atomic actions that make up a task —what we have represented as terminals in the thesis. All three frameworks do so by running either inverse-forward pairs or detectors concurrently and checking which one has the highest confidence or score.

Secondly, the systems need to detect the tasks, which we have represented as a sequence of atomic actions and specified in the form of a SCFG. Note that HAMMER places no constraints on how to recognise these higher-level actions.

Lastly, by analysing the differences in how users perform tasks we obtain a user model. The user model could be of the user abilities —by measuring their performance at different tasks— or of her preferences —by checking her preferred mappings between different terminals (or musical segments).

	HAMMER	ROBOTIC TUTOR FRAMEWORK	SYNCHRONISED GRAMMARS FRAMEWORK
LOW-LEVEL ACTION RECOGNITION	*Pairs of inverse and forward models* implemented with proportional-integral-derivative (PID) controllers (Demiris and Hayes, 2002), low-level motor commands (Johnson and Demiris, 2004) or Kalman filters (Ognibene et al., 2013).	*Heuristic detectors* which take the wheelchair position and output the corresponding waypoint.	*Heuristic detectors* which take a fiducial position on the Reactable and output the corresponding melody/accompaniment track.
COMPOSITE ACTION RECOGNITION (HIGH-LEVEL)	*Hierarchies of inverse-forward pairs* implemented as graphs (Johnson and Demiris, 2004) or finite-state machines (Sarabia et al., 2011).	*Stochastic Context-Free Grammars* for recognition of complex, hierarchically organised sequences of low-level actions.	
USER MODEL	Extracted from tasks successfully performed by user and the tasks the user has the potential to achieve; that is, the ZPD (Demiris, 2009)	Derived from performance difference with expert across different driving tasks.	Obtained from different user preferences with regards to synchronisation of music and accompaniment tracks.

Table 5.3: Approaches to low-level action recognition, composite action recognition (high-level action recognition) and user modelling by each of the frameworks discussed in this thesis.

6

CONCLUSIONS AND FUTURE WORK

Throughout this thesis we have presented frameworks to effectively model the abilities and preferences of users in structured scenarios. We have done so by leveraging the representational power of Stochastic Context-Free Grammars (SCFGs) and drawing inspiration from the multiple model competition of the Hierarchical Attentive Multiple-Models for Execution and Recognition (HAMMER) architecture. We have further provided optimised implementations of these algorithms in the hope that the community may adopt and adapt them to their problems.

Moreover, we have described two distinct scenarios which benefit from these user modelling frameworks: a wheelchair sidekick and a musical collaborator. We have further detailed as a further third scenario —the hospital companion— to which we will apply these frameworks in the future.

6.1 SUMMARY OF CONTRIBUTIONS

The main contribution of this thesis is a flexible framework to model the abilities of users in structured tasks. The framework provides a principled way to evaluate the user abilities at a certain task. In our case, the task was to drive a wheelchair through several waypoints, though this could be extended to other situations. This framework uses SCFGs to represent the tasks, which allows for robust detection of sequences of actions while at the same time letting researchers to easily express these tasks. Further, SCFGs provide a principled basis for task performance evaluation as well as prediction of the most likely next step. Low-level action recognition in this framework is done through a HAMMER-inspired architecture with multiple competing models. The application of the user model to the interaction is taken from research in the field of Intelligent Tutoring Systems (ITSs).

We have also contributed a framework to synchronise two SCFG parsers, thus allowing the states of a parser to influence another parser. This was applied to the field of human-robot musical collaboration, but it may be a useful approach to other situations such as human-robot assembly or recognition of events by two agents. Moreover, the cornerstone of this framework —the synchronisation matrix— can be used to model and encode the user preferences.

These contributions are not only theoretical, the work for this thesis has resulted in optimised and open-source developer kits for both the HAMMER architecture and a SCFG parser. Both of these libraries are generic and could be adapted for a multitude of problems beyond the scope of this thesis.

From a more applied point-of-view, this thesis presents a novel scenario for Human-Robot Interaction (HRI): a wheelchair sidekick. We have presented several possible roles for a humanoid accompanying a power-wheelchair user: pointing out obstacles, giving directions, and tutoring users so that they can improve their driving skills. We have trialled the system in many situations, with two different wheelchairs and with adults and children. And though the actual objective improvements of having such a sidekick are yet to be proven, it is clear that most users enjoy the experience of having a robot by their side —85.7% of children strongly agreed they liked driving with NAO by their side and most adult users preferred the physical robot over a simulated one.

Finally, we have described a Wizard of Oz (WoZ) controlled robotic companion for patients in an acute hospital setting. Our contribution lies in the breadth of our study. We reported 64 interactions with patients including a 18 year old teenager and five nonagenarian patients, all with different reasons for being in the hospital. Yet our results showed that most patients enjoyed their time with NAO and that the robot made several patients with dementia smile. This positive attitude of patients towards the companion is perhaps best highlighted by the overwhelmingly positive comments of the patients themselves with 84% of them agreeing they had enjoyed interacting with NAO.

6.2 LIMITATIONS

Both the robotic tutor and the synchronised grammar frameworks —introduced in sections 5.1 and 5.2 respectively— had some limitations which we discuss in what follows.

TIME REPRESENTATION As we have indicated, SCFGs do not track the timing of events. In this thesis, we compensate for this by comparing against the speed of an expert, but this may not be applicable to other scenarios. Another possible approach, is to generate different terminals for the same events with different timing, however this approach does not scale if it is necessary to model complex timing relationships between events. Zhang et al. (2011) provides a time-aware SCFG parser, but that comes at the expense of greater parsing complexity.

FORMAL CONCEPT DRIFT FORMULATION User models have to balance between old information about the user, which may be outdated; and new information, which may be noisy. The user modelling frameworks presented in this thesis are meant to be continuously updated and can indeed balance between old and new information. Nonetheless, we did not investigate a formal way to temporally discount old observations.

TASK STRUCTURE SPECIFICATION Both of the scenarios where we applied our user modelling frameworks had an underlying structured task definition. We specified manually the tasks, but they could have been learnt from demonstration (Lee et al., 2013). Despite that, if the task to model does not have a clearly defined structured, specifying —whether manually or by demonstration— the task structure will be very challenging and, as a consequence, may not benefit from our frameworks.

EXPERT DEMONSTRATIONS This limitation only applies to the robotic tutor framework, which requires an expert demonstration of the task as a baseline to evaluate other users. As mitigation, particularly for automatically generated tasks, the system could simulate the task and obtain a virtual expert task demonstration to compare against.

6.3 TOWARDS THE FUTURE

In addition to addressing the previously listed limitations, there are naturally several directions into which the research we have presented in this thesis could be taken. In what follows, we outline some of the most promising ones.

6.3.1 Long-term studies

The studies we carried out for this thesis were relatively short, with up to 30 minutes of interaction between the participant and the robot. What would happen if the interaction was repeated over a long period of time? It is not at all clear that users would not get bored of the current capabilities of the robotic companions. Humans interactions evolve over time and interactions with robots should too.

A possible approach to tackle this problem is to make use of the framework we have presented in this thesis to model users abilities. The robot could lock some of its abilities and only unlock them once the user has demonstrated sufficiently proficient interactions with the robot.

Long-term studies would require a greater number of tasks, which may prove to onerous to specify manually. Instead, the robotic tutor could generate new tasks following the research of the ITS community (Martin and Mitrovic, 2002).

6.3.2 Studies with disabled children

As we have already stated the wheelchair sidekick scenario was devised with children who have physical and cognitive disabilities in mind. However, the complexities of organising such as study have prevented us from doing so thus far. For example, children with disabilities often need very specialised seating systems that normally are not compatible with ARTY's. Moreover, some children may not have sufficient cognitive capabilities to recognise what NAO is trying to tell them.

Of course, there are still many children who may gain from a robotic sidekick and we look forward to long-term studies with the target population that definitely establish the benefits of having a robotic companion in the wheelchair, once the more practical problems like seating and creating friendly interfaces for the occupational therapists are solved.

6.3.3 Improving the tutoring skills by using the user model

The current abilities of the robotic tutor could be enhanced by taking further advantage of the information from the user model. For example, the number of hints the robot gives or whether the robot gives hints pro-actively should be modulated by previous performance scores. Lower user scores would result in a higher degree of help from the robot, and as the scores increase the robot would refuse to help the user.

Even within the scope of a single task execution the robot may give feedback to the user about how they are progressing: praising the user if the score derivative is going up or recommending her to request more hints if the derivative is negative.

Furthermore, the robotic tutor would likely be more useful if it was equipped with a hierarchy of tasks which are unlocked depending on the user abilities determined by the user model —akin to the concept of Zone of Proximal Development (ZPD) which Demiris (2009) adapted from Vygotsky (1978). With such abilities a robotic tutor may indeed be useful in many other fields besides that of power-wheelchair tutoring.

6.3.4 Synchronisation between multiple SCFG parsers

Our synchronisation grammars framework allows for the state of an stochastic grammar to affect that of another stochastic grammar. This could be extended to synchronise multiple stochastic grammars by growing the synchronisation matrix to an n-dimensional matrix and learning from the co-occurrences of the terminals of the various stochastic grammars.

Nonetheless, as the number of terminals and stochastic grammars grows so too will the synchronisation matrix. Further research is required to develop sparse representations of the relationships between the terminals of the different stochastic grammars.

6.3.5 A graphical interface for specification of stochastic tasks

Throughout this thesis we have had to specify SCFGs of tasks manually: from the *Macarena* dance, to Baxter's musical sequence. This is very simple once one learns how to perform recursion or to skip a step or a series of steps. In spite of this, the process of specifying SCFGs manually remains tedious and error-prone.

A computer program which allows to visually define steps and their relationships as well as recursion, step skipping, repeated execution would greatly simplify the specification of SCFGs. The program could further assign probabilities automatically based on user constraints, eg. "step A is twice as likely to happen as step B" or "step C has a 10% chance of not appearing".

Moreover, a graphical visualisation would be useful not only for generation of stochastic grammars, but also for debugging of already created grammars.

6.4 EPILOGUE

Personal robots need to learn from their users and adapt to them, just like humans do. But it is not only robots that may benefit from maintaining models about their users: computer programs, websites and smartphones may benefit from tailoring their interfaces to the specifics of each user as well. This thesis represents a step into a future where all of our interactions with machines are personalised.

P

TOWARDS A PARALLEL STOLCKE-EARLEY PARSING ALGORITHM

As we saw in section 3.4, quickly parsing Stochastic Context-Free Grammars (SCFGs) allows us to deal with more complex sequences. Sakakibara et al. (1994) echo this finding by stating that "the main difficulty of [using SCFGs] to other families of RNA is [...] the computational cost of parsing longer sequences."

This appendix aims to achieve even higher parsing speeds than those already presented in the thesis and thus is dedicated to parallelising the Stolcke-Earley parsing algorithm and implementing the parallel version in a Graphics Processing Unit (GPU) with OpenCL[1].

There are other parallel parsers in the literature. Most notably, Canny et al. (2013) implemented a SCFG parser in a GPU achieving very high performance for dense grammars —that is grammars where most productions can generate most tokens. Hall et al. (2014) followed up on this approach and added pruning so the parser could deal with sparse grammars.

The methods introduced by Canny et al. (2013) and Hall et al. (2014) represent the state-of-the-art in terms of raw performance. However, their implementations are based on the Cocke-Kasami-Younger (CKY) bottom-up algorithm and were tested with input of up to 40 words. Our focus is to improve the maximal parsing length rather than the sentences parsed per second.

There have also been attempts at parallelising Earley's algorithm. In particular, Chiang and Fu (1984) describe a parallelisation scheme for a non-stochastic Context-Free Grammar to be printed in an integrated circuit. Their key insight is to replace Earley's prediction step by a prediction of all possible non-terminals thus freeing a point of contention and allowing more parallel work. Unfortunately, this approach is not viable for stochastic grammars as the algorithm requires keeping track of prefix probabilities which do depend on previous inputs. Chiang and Fu also use bit-vectors to efficiently represent the grammar and its states. Whilst this is a very efficient representation of a grammar with few terminals, it does not match well with the architecture of a GPU where types are limited to 8, 16, 32 and 64 bits. Moreover, such representation will likely not scale

1 OpenCL is a vendor-neutral language for general computation on a Graphics Processing Unit.

well with the number of symbols in the grammar —a thousand terminals would require a thousand bits just to store a single terminal.

It is clear that a new parallel implementation customised for GPUs would be desirable. The Stolcke-Earley is irremediably serial in the application of the *scan*, *complete*, *predict*. Our approach is, in consequence, to parallelise each of these functions. We provide working algorithms for scan and predict. We also compare the performance of the GPU scan and predict with SARTParser's performance (cf. section 3.4). Finally, we sketch out how to parallelise complete and discuss why doing so is not trivial.

P.1 PARALLEL ALGORITHM DESCRIPTION

At this point, we recommend the reader to revise section 3.3 as we will assume familiarity with the serial version of the Stolcke-Earley algorithm.

P.1.1 Efficient parsing data structures

OpenCL does not allow the declaration of complex C++-style classes in the GPU, though it does permit using C-like structs. Since reading or writing to the memory of the GPU is an expensive operation it is also preferable if the structures are as small as possible.

There are several data structures which require our attention: the stochastic grammar itself as well as the parsing states and state sets that the parser will generate. We start by dividing stochastic grammars into three different OpenCL structures: the list of symbols ($\mathcal{T} \cup \mathcal{N}$), the rules ($\mathcal{R}$) and the rule probabilities ($\mathcal{P}$).

The grammar symbols —non-terminals and terminals— are simply represented by an unsigned number which acts as their identifier. In order to verify if a symbol is a terminal or a non-terminal we require:

$$\forall_{n \in \mathcal{N}} \quad 0 < \mathrm{id}(n) \leqslant |\mathcal{N}| \tag{P.1}$$

$$\forall_{t \in \mathcal{T}} \quad |\mathcal{N}| < \mathrm{id}(t) \leqslant |\mathcal{N} + \mathcal{T}| \tag{P.2}$$

This way, it is trivial to establish whether a symbol is a terminal or a non-terminal by comparing it with the number of non-terminals. The 0 id is reserved to the empty symbol which expands to the axiom on the initial rule ($0 : \quad_0 \rightarrow .\mathcal{S} \quad [1.0, 1.0, 1.0]$).

Rule probabilities are modelled as an array of floats where the array index is the id of the rule and the array value is the probability of said rule.

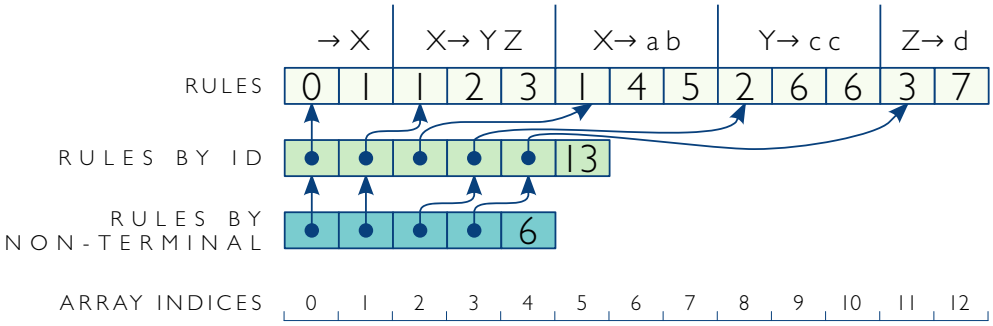


Figure P.1: Visual representation of the GPU encoding of stochastic grammar rules. The last entry in *rules by id* is the size of *rules*. Similarly the last entry in *rules by non-terminal* is the size of *rules by id*. This arrangement allows us to quickly compute the number of terminals in a rule as well as the number of rules with $X \to \ldots$ with $X \in \mathcal{N}$.

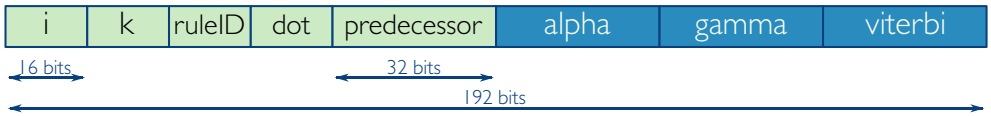


Figure P.2: Visual representation of a parsing state in a GPU. Boxes shaded in light green represent unsigned integer types, whereas boxes shaded in dark blue represent floating point types.

Representing the actual grammar rules is more complicated, as shown in fig. P.1. We create three distinct arrays, the first one (*rules* in the figure) contains all the rules in flattened form; that is, just the ids of the symbols of which the rules are composed. The second array contains pointers to the start of each of the rules (*rules by id* in the figure), this way it is possible to differentiate between the beginning and the end of a rule. To simplify computation the last entry of the *rules by id* array contains the size of the *rules* array. Note that the indices of this array are effectively the ids of each rule. Thirdly, the last array groups the rules by the non-terminal they have on the left-hand side (*rules by non-terminal* in the figure). Similar to the previous array, the last entry contains the size of the *rules by id* array.

This representation, though complicated at first sight, allows us to obtain the next element after the dot with just a look-up and a sum (`rules[rulesByID[ruleID]+dot+1]`). Further it allows to compute the number of symbols in a rule (`rulesByID[ruleID+1] - rulesByID[ruleID]`) as well as the number of rules that have a non-terminal on the left-hand side of the expression (`rulesByNonTerminal[nonTerminalID+1] - rulesByNonTerminal[nonTerminalID]`). The latter is useful for the predict operation as we will explain shortly.

The parser will generate many states, and therefore a compact structure to represent these parsing states is required. The structure we designed is shown in fig. P.2. It takes only 192 bits to represent a parsing state independently of the grammar specification.

States are stored in an array in the GPU and synchronised with the CPU at the end of each scan, predict and complete operation. The CPU keeps track of the state set boundaries and passes the appropriate offsets to the GPU.

P.1.2 Probability representation

We are using 32-bit floats to store probabilities following convention in GPU programming. Since the Stolcke-Earley algorithm multiplies probabilities often, the floats will become very small rapidly and will eventually underflow. The solution is to store the forward, inner and Viterbi probabilities in log-space. Though this entails some loss of precision, it is not important as we are often more interested in the relative probabilities of parsing states with respect to each other.

Using log-space has four ramifications in the implementation: full probabilities are represented as 0, zero probabilities are stored as $-\infty$, multiplication becomes addition, and addition needs to be managed to explicitly deal with underflows. We utilise the code shown in listing P.1 to add two numbers in log-space.

```
float logAdd(float logA, float logB)
{
    // if a or b are nan return nan
    // otherwise return logA + log( 1.0 + exp(logB - logA) )
    // unless the second part of the expression fails (logA >> logB) in
    // which case return logA (assuming logA > logB)
    float bigger = std::max(logA, logB);
    float smaller = std::min(logA, logB);

    float negDiff = smaller - bigger;
    float toAdd = log(1.f + exp(negDiff) );

    bool isNaN = ( isnan(logA) ) || ( isnan(logB) );
    bool invalidToAdd = ( isnan(toAdd) );

    if (isNaN) return NAN;
    else if (invalidToAdd) return bigger;
    else return bigger + toAdd;
}
```

Listing P.1: C++ code to add two numbers in log-space together.

Now that we have devised the basic structures of the parallel parsing algorithm, let us focus on the GPU implementations of scan and predict.

P.1.3 Scan

The role of scan in the Stolcke-Earley algorithm is to incorporate inputs into the parser. It does so by advancing the dot in those parsing states whose next symbol is a terminal and matches any of the input terminals (refer to section 3.3.1 for a detailed explanation).

In order to parallelise this function, we divide scan into two GPU kernels: pre-scan and scan. Pseudo-code for each of these kernels is shown in fig. P.3.

Pre-scan determines the states that need to advance their dot. We use a GPU thread for every state in the previous state-set and for every input terminal. Each thread then checks if their assigned input terminal would advance the dot of their assigned state. If so, this is noted down on the `binaryMask` array.

By applying the well-known exclusive sum-scan algorithm (Harris et al., 2007) to `binaryMask` we can obtain the output destination of the yet-to-be created scanned states. A visual depiction of sum-scan is shown in fig. P.4. Note that since sum-scan is a standard GPU algorithm we do not include it in our pseudo-code. The array `allocation` is the result of applying sum-scan to `binaryMask` and contains the memory offsets for every state that will be generated.

Lastly, we execute the scan kernel which has one thread for every state and for every input terminal. Each thread then checks if their state-terminal combination spawns a new scanned state (by looking up the `binaryMask`) in which case a new state is generated by multiplying (adding in log-space) the α, γ and ν probabilities times the input terminal probability and advancing the dot. The newly scanned state is stored in the GPU's memory according to the position indicated by the `allocation` array.

P.1.4 Predict

The job of predict is to expand states which are waiting for a non-terminal to be completed (cf. section 3.3.3). Our approach to parallelise this operation is similar to scan's: we split the task into two kernels: pre-predict and predict (pseudo-code for these kernels is shown in fig. P.5).

Pre-predict requires a thread for every non-terminal of the grammar and for every state in the current state-set. Each thread then obtains the non-terminal after the dot on their assigned state and checks if new rules with their assigned non-terminal must be created. This check consist on verifying that the entry in the probabilistic, reflexive, transitive left-corner relation ($\mathbf{R}_L$) for both non-terminals (the next symbol in the thread's parsing state and the thread's non-terminal) is different from 0. If new states must be

```
kernel Pre-scan
    ✎ This is a 2D kernel, the first dimension is for the states to analyse and the second for the terminals.
  stateId ⟵ threadIndex[0]
  stateSize ⟵ threadSize[0]
  terminalId ⟵ threadIndex[1]
  destination ⟵ stateId + terminalId × stateSize
  state ⟵ getState(stateId) ✎ get the state represented by stateId.
  next ⟵ nextSymbol(stateId) ✎ get the terminal/non-terminal after the dot.
  terminal ⟵ getTerminal(terminalId) ✎ get the terminal represented by terminalId.
  if  validState(stateId) and next = terminal  then
    binaryMask[destination] ⟵ true
  end if
end kernel

kernel Scan(binaryMask, allocation)
  ✎ 2D kernel, the first dimension is for the states to analyse and the second for the terminals.
  ✎ allocation contains this thread's output destination and is the output of sum-scan.
  stateId ⟵ threadIndex[0]
  stateSize ⟵ threadSize[0]
  terminalId ⟵ threadIndex[1]
  destination ⟵ allocation[(stateId + terminalId × stateSize)]
  prob ⟵ getProbability(terminalId) ✎ Get the probability associated with the input terminal.
  if binaryMask[destination] then
    state ⟵ getState(stateId)
    state.i ⟵ state.i + 1
    state.dot ⟵ state.dot + 1
    state.alpha ⟵ state.alpha + prob
    state.gamma ⟵ state.gamma + prob
    state.v ⟵ state.v + prob ✎ Note k and predecessor are not modified.
    states[allocation[dest]] ⟵ state
  end if
end kernel
```

Figure P.3: Pseudo-code for the parallel scan algorithm divided in two different kernels. Variables highlighted in red represent the output of each kernel.

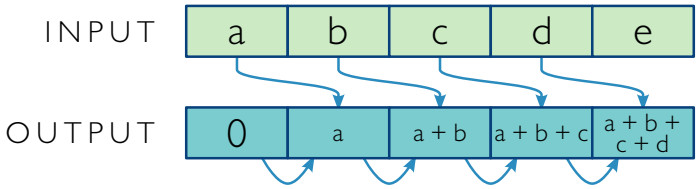


Figure P.4: Visual description of sum-scan. The output array may be used as the offsets for threads to safely write the number of elements in the input array.

```
kernel Pre-predict
   ✎ 2D kernel, the first dimension is for states to analyse and the second for the non-terminals.
   stateId ⟵ threadIndex[0]
   stateSize ⟵ threadSize[0]
   nonTerminalId ⟵ threadIndex[1]
   destination ⟵ stateId + nonterminalId × stateSize
   if validState(stateId) then
      next ⟵ nextSymbol(stateId) ✎ get terminal/non-terminal after the dot.
      nonTerminal ⟵ getNonTerminal(nonTerminalId)
      ✎ get non-terminal represeted by nonTerminalId.
      rl ⟵ getFromRLMatrix(next, nonTerminalId)
      ✎ get the entry in the matrix R_l{next, nonTerminal}.
      if  isNonterminal(next) and rl ≠ −∞  then
         probMatrix[destination] ⟵ state.alpha + rl
         countVector[nonTerminalId] ⟵ rulesStartingWith(nonTerminal)
         ✎ Save the number of rules starting with nonTerminal
      end if
   end if
end kernel

kernel Predict(alphaProbs, countVector, allocation, i)
   ✎ 2D kernel, the second dimension is for the non-terminals whilst the first is for each of the rules.
   n^thRule ⟵ threadIndex[0]
   nonTerminalId ⟵ threadIndex[1]
   destination ⟵ allocation[nonTerminalId] + n^thRule
   alphaProb ⟵ alphaProbs[nonTerminalId]
   ruleProb = getRuleProb(nonTerminalId, n^thRule)
   haveToExpand ⟵ ( countVector[nonTerminalId] ≠ 0 )
   if  n^thRule < rulesStartingWith(nonTerminalId) and haveToExpand  then
      state ⟵ newState()
      state.i ⟵ i
      state.k ⟵ i
      state.dot ⟵ 0
      state.predecessor ⟵ nil
      state.alpha ⟵ alphaProb + ruleProb
      state.gamma ⟵ ruleProb
      state.v ⟵ ruleProb
      states[destination] ⟵ state
   end if
end kernel
```

Figure P.5: Pseudo-code for the parallel scan algorithm divided in two different kernels. Variables highlighted in red represent the output of each kernel.

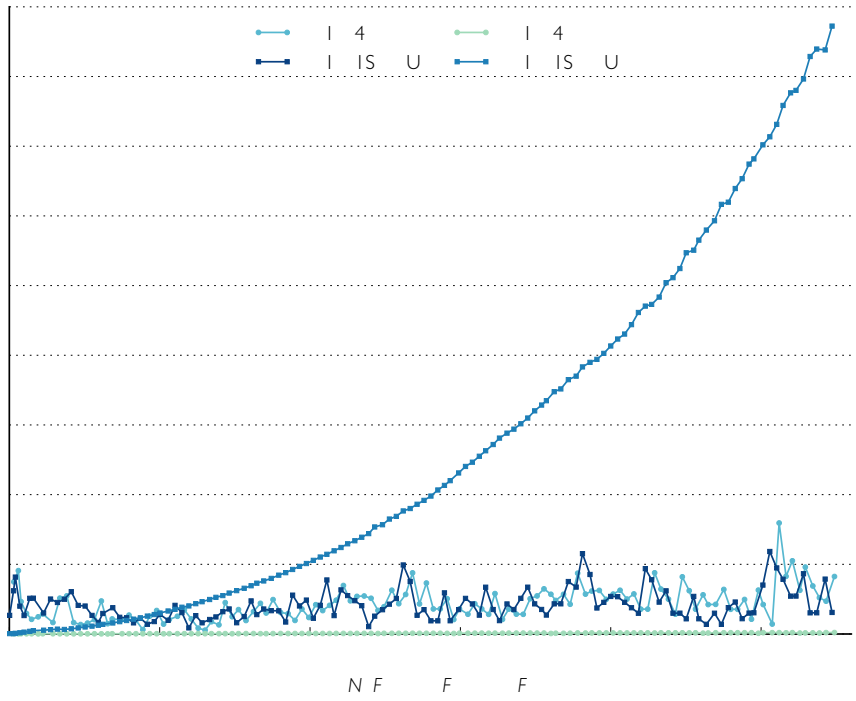


Figure P.6: Speed comparison between the CPU and the GPU implementations of scan and predict. Lines are the obtained average running times of 250 rounds.

generated, the thread will store the number of new states in `countVector`, one per rule in the grammar with the thread's assigned non-terminal on the left-hand side. Additionally, the α probability multiplied by the probability from $\mathbf{R}_l$ is saved into `probMatrix`.

As with scan, a sum-scan operation is then applied to `countVector` to determine the positions in memory of the yet-to-be created predicted states in memory. Additionally, we also sum-scan `probMatrix` to add the probabilities from all cases where more than one state may predict a rule. We define `allocation` and `alphaProbs` as the results of the sum-scans of `countVector` and `probMatrix`.

The last step is to execute the predict kernel which spawns a thread for every non-terminal and for every rule with that non-terminal in the left-hand side. Each thread then checks if it needs to add a rule by checking if the entry `countVector` for their non-terminal is non-zero. If so they will create a new state with their assigned rule and using `alphaProbs` to keep track of the α probabilities. The state is stored in memory according to the results of `allocation`.

P.2 PRELIMINARY RESULTS

We implemented predict and scan in OpenCL on an Intel Core i7-2600 with 16GiB of RAM and a Nvidia Quadro 600 with 1GiB of RAM and 96 CUDA cores running Ubuntu 12.04 with Nvidia's driver 352.30. Whilst the CPU we used for our tests is still competitive with current CPUs that is not the case with the GPU —modern Nvidia GPU usually have around 2048 CUDA cores as well as more and faster memory.

We tested the GPU implementations of scan and predict against those of the sped-up version of SARTParser. Both versions were built with compiler optimisations enabled. 250 parsing rounds were performed with the same sequence we used in section 3.4 as input. As expected, the output results of each parser were identical every time. The results are shown in fig. P.6.

Notwithstanding the relative advantage of the CPU, we found that the GPU version of predict was significantly faster, whereas the reverse was true for scan. Focusing only on the final parsing step highlights these differences: GPU predict finished in 1.54ms on average (σ =8.92ms) while CPU predict took an average 43.62ms (σ =2.25ms). For scan, the GPU version took 4.12ms on average (σ =17.58ms) and the CPU: 0.10ms (σ =0.006ms).

It must be noted that the variability in the GPU performance is quite large, in the worst case scenario the GPU took 100.88ms to complete the last predict operation and 88.48ms for the last scan operation. We will leave it for the discussion to explain the reason for this variability.

P.3 TOWARDS A PARALLEL COMPLETE

The role of complete is detecting when a state is finished (ie. the dot has reached the end of the rule) and advancing the dot on all the terminals that have been waiting on the non-terminal on the finished state's non-terminal. A detailed explanation can be found in section 3.3.2.

A GPU implementation of complete would need three kernels. The first one would gather the states that have been finished and annotate the non-terminals on the left-hand side of these states (akin to what pre-predict does). The second one would find which of the states may need have to advance the dot (similar to the operation of pre-scan). The last kernel would generate the new states (as scan and predict do).

There is one key difference with scan and predict. Whereas with the other operations, all states will be generated after one iteration[2], with complete we need to recursively execute the above three kernels.

A further difficulty arises from the fact that the same state may be generated by two different paths. Whilst theoretically the parsing algorithm works with duplicated states, in practice not merging them means that the GPU will run out of memory due to the exponential complexity of the algorithm. The only solution is then detecting the duplicates and merging them. Duplicate detection is paralellisable in GPUs (Forchhammer et al., 2013), but merging of two states in parallel is extremely difficult with current GPU architectures. We experimented with implementing mutexes in OpenCL —something which goes counter the principles of GPU programming— but we found the implementation to be neither reliable nor portable.

P.4 DISCUSSION AND CONCLUSION

Our results have shown that the GPU version of predict is, on average, much faster than the CPU version of the same function. The reason for this speed-up lies in both the optimised representation of the parser structures as well as the parallelisation of the work. Though scan is slower on the GPU than on the CPU we speculate this is due to low occupancy in the GPU and that, given enough states to process, the parallel version will be faster than the serial version.

The main disadvantage of the GPU versions of the scan and predict operations come from their big variability, in the worst case it may take up to 100ms to complete a single operation. This variability is not altogether surprising since executing a kernel in a GPU requires sending data to the GPU, executing the kernel itself and retrieving data from the GPU. Each time, operating system calls need to be made which may take a long time to be serviced depending on the operating system's current load.

However, the difficulty of implementing complete in the GPU is more significant. At the moment, merging duplicate states is hard. Yet, we expect that it will be possible in the future, be it by some extra hardware capability in GPUs or some new algorithm developed by the community or a combination or both.

A GPU implementation of the Stolcke-Earley could not only significantly speed-up the parsing of SCFGs, but also allow us to run the algorithms in systems that have average CPUs but very powerful GPUs, such as mobile phones.

2 Predict, technically, generates states recursively but the probabilistic, reflexive, transitive left-corner relation flattens all recursions.

R

ROBOTS & SENSORS

This thesis features several robots and other specialised hardware, which we now describe for reference.

R.1 ROS

We have extensively used Robot Operating System (ROS) —as described by Quigley et al. (2009)— throughout this thesis to control the robots. Specifically, we made use of the two most recent versions: *hydro* (released in September 2013) and *indigo* (released in July 2014).

Contrary to what its name may indicate, ROS is not an operating system, it actually requires an underlying operating system such as Ubuntu Linux. Instead, ROS is a collection of libraries for robotics.

As its core, ROS is a communication library that allows many-to-many message passing between different processes through the network. It also contains tools for standardisation of the message to be passed as well as the ability to execute remote code in another machine. The robotics community has largely embraced ROS and as a result it is now easy to download and set-up state-of-the-art algorithms such as Adaptive Monte-Carlo Localisation (AMCL) navigation and inverse kinematics amongst others. ROS works with most robots and sensors used in robotics research.

ROS has binding for many languages, but in this thesis we solely rely on its C++ native library and its Python bindings.

The Python bindings had a bug that could lead to a deadlock under a certain conditions. In our case, it triggered very often with the wheelchair sidekick (section 4.1). We analysed the code, and found the reason was an unprotected use of a shared variable. We wrote code to fix the error as well as a test case so the code would not regress. Both the fix and the regression test are now part of the official ROS distribution.

(a) ARTA (b) ARTY (c) Baxter

(d) Kinect (e) NAO (f) Reactable

Figure R.1: Robots and other hardware used for this thesis.

R.2 ARTA

The Assistive Robotic Transport for Adults (ARTA) is an electrically powered indoor-outdoor wheelchair for adults (fig. R.1a). ARTA was built from a DMA Escape mid-wheel-drive with four caster wheels: two at the back and two at the front. The wheelchair is fully analogue and features a VSI Anderson d50962 controller which translates the joystick signals from the user to motor commands. We tap into the VSI controller with a bespoke circuit board that permits that an Arduino UNO to control all the signals to the motors. The computer, an Intel Core i7 laptop with 8GiB RAM and a touch screen, communicates with the Arduino board with a custom protocol via serial-over-USB. We developed both the driver for a Linux machine to control the wheelchair as well as the module to control of the wheelchair from ROS topics.

The wheelchair is also equipped with 2 Houkuyo URG-04LX-UG01 and a SICK LMS 200 laser range finders which detect obstacles in a 360° plane. A Phidgets Spatial 3/3/3

1056_0 3-axis accelerometer, gyroscope and compass was also added to help with navigation. We use the standard ROS drivers for all these accessories.

ARTA was used in chapter 5.

R.3 ARTY

The Assistive Robotic Transport for Youngsters (ARTY) is an electrically powered indoor outdoor wheelchair designed for children (fig. R.1b) which was originally developed by Soh and Demiris (2012). ARTY was developed from an Ottobock Skippi wheelchair which is a rear-wheel drive wheelchair with two caster wheels at the front. The wheelchair is digitally controlled through a CAN bus controller with which we interface with through a PEAK PCAN-USB adapter. The wheelchair has an on-board Intel Atom dual-core computer with 2GiB, though for our experiments we also used a laptop with an Intel Core i7 processor, 8GiB RAM and a touch screen.

Additionally, ARTY is fitted with 3 Hokuyo URG-04LX-UG01 and a PhidgetSpatial 3/3/3 1056_0. This way, ARTY perceives all the obstacles around itself in a plane of 360°.

The software is also based on ROS and much of it is shared with ARTA so that code for one wheelchair should work with the other. This was not the case originally, we ported the code-base to the more recent versions of ROS and made sure it was compatible with the ARTA.

Note that ARTA was not finished till late in our research, and consequently we have also used ARTY with adults. ARTY was used in chapter 4.

R.4 BAXTER

Baxter is humanoid robot commercially developed by Rethink Robotics with two 7 degrees of freedom arms, each with a payload to 2.3kg (fig. R.1c). Baxter is a large robot, up to 1.9m of height depending on pedestal set-up and 1.1m arm length. It additionally features a display which serves as the head of the robot, and three cameras: one in each arm and another in the head. Baxter contains an Intel Core i7 processor and its drivers are written natively in ROS. On the other hand it has no legs, nor a mobile base.

We worked with Baxter in chapter 5.

R.5 REACTABLE

Reactable is an electronic musical instrument composed of a projector, an infrared camera and a translucent surface atop a table which shows the projected image. Using the reacTIvision framework (Kaltenbrunner and Bencina, 2007), the infrared camera can detect any fiducials placed on top of the translucent surface. Through calibration of the projected image, a computer program can establish where the fiducials are placed with respect to its window and react accordingly by, for instance, emitting different sound segments.

We used Reactable in chapter 5.

R.6 NAO

NAO is a 58cm tall humanoid robot with 25 degrees of freedom developed by Aldebaran (fig. R.1e). NAO is able to walk on its own, though we do not take advantage of this feature in this thesis. It has many sensors: two cameras (one looking at the front and the other looking down), four microphones, capacitive buttons, sonar range finders on the chest and bumpers on its feet. Moreover, it has two speakers which can reproduce speech or music as required.

Its on-board processor is an Intel Atom at 1.6Ghz and therefore most of the processing happens remotely. NAO is usually programmed through NAOqi, an event-driven Application Programming Interface for Python and C++. However to integrate it better with all other system we make use of an open-source ROS wrapper. We contributed to this wrapper the modules to control NAO behaviours, the speech engine and the build system. The wrapper is available from: `http://wiki.ros.org/naoqi_driver`.

NAO was featured in chapters 3 to 5.

R.7 KINECT

Kinect is an infrared-based depth sensor with an RGB camera (fig. R.1d). It can be used alongside OpenNI and NITE to perform real-time skeleton detection. Note we use the term Kinect to refer to both to the Microsoft Kinect sensor and the almost identical Asus Xtion PRO. Kinects were used in Chapter 3.

BIBLIOGRAPHY

Aggarwal, J. and Ryoo, M. S. (2011), 'Human activity analysis', *ACM Computing Surveys* **43**(3), 1–43. doi: 10.1145/1922649.1922653

Alissandrakis, A., Nehaniv, C. L. and Dautenhahn, K. (2002), 'Imitation with ALICE: learning to imitate corresponding actions across dissimilar embodiments', *IEEE Transactions on Systems, Man, and Cybernetics* **32**(4), 482–496. doi: 10.1109/TSMCA.2002.804820

Allen, J. F. (1983), 'Maintaining knowledge about temporal intervals', *Communications of the ACM* **26**(11), 832–843. doi: 10.1145/182.358434

Bainbridge, W. A., Hart, J. W., Kim, E. S. and Scassellati, B. (2011), 'The Benefits of Interactions with Physically Present Robots over Video-Displayed Agents', *International Journal of Social Robotics* **3**(1), 41–52. doi: 10.1007/s12369-010-0082-7

Baroni, I., Nalin, M., Baxter, P., Pozzi, C., Oleari, E., Sanna, A. and Belpaeme, T. (2014), What a Robotic Companion Could Do for a Diabetic Child, *in* 'Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication', pp. 936–941. doi: 10.1109/ROMAN.2014.6926373

Belpaeme, T., Baxter, P. E., Read, R., Wood, R., Cuayáhuitl, H., Kiefer, B., Racioppa, S., Kruijff-Korbayová, I., Athanasopoulos, G., Enescu, V., Looije, R., Neerincx, M., Demiris, Y., Ros, R., Beck, A., Cañamero, L., Hiolle, A., Lewis, M., Baroni, I., Nalin, M., Cosi, P., Paci, G., Tesser, F., Sommavilla, G. and Humbert, R. (2012), 'Multimodal Child-Robot Interaction: Building Social Bonds', *Journal of Human-Robot Interaction* **1**(2), 33–53. doi: 10.5898/JHRI.1.2.Belpaeme

Block, L., Habicht, R., Wu, A. W., Desai, S. V., Wang, K., Silva, K. N., Niessen, T., Oliver, N. and Feldman, L. (2013), 'In the wake of the 2003 and 2011 duty hours regulations, how do internal medicine interns spend their time?', *Journal of General Internal Medicine* **28**(8), 1042–1047. doi: 10.1007/s11606-013-2376-6

Butler, S. and Demiris, Y. (2010), Partial Observability During Predictions of the Opponent's Movements in an RTS Game, *in* 'Proceedings of the IEEE Conference on Computational Intelligence and Games', pp. 46–53. doi: 10.1109/ITW.2010.5593374

Canny, J., Hall, D. and Klein, D. (2013), A multi-Teraflop Constituency Parser using GPUs, *in* 'Proceedings of the Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, pp. 1898–1907.

Carlson, T. and del R. Millán, J. (2013), 'Brain-Controlled Wheelchairs: A Robotic Architecture', *IEEE Robotics & Automation Magazine* **20**(1), 65–73. doi: 10.1109/MRA.2012.2229936

Carlson, T. and Demiris, Y. (2010), Robotic Wheelchairs: Scientific Experimentation or Social Intervention?, *in* 'Proceedings of the Workshop on the Role of Experiments in Robotics Research at ICRA'.

Carlson, T. and Demiris, Y. (2012), 'Collaborative control for a robotic wheelchair: evaluation of performance, attention, and workload', *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics* **42**(3), 876–888. doi: 10.1109/TSMCB.2011.2181833

Censi, A. (2008), An ICP variant using a point-to-line metric, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', pp. 19–25. doi: 10.1109/ROBOT.2008.4543181

Chiang, Y. T. and Fu, K. S. (1984), 'Parallel Parsing Algorithms and VLSI Implementations for Syntactic Pattern Recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(3), 302–314. doi: 10.1109/TPAMI.1984.4767522

Chomsky, N. (1956), 'Three models for the description of language', *IEEE Transactions on Information Theory* **2**(3), 113–124. doi: 10.1109/TIT.1956.1056813

Chrysafiadi, K. and Virvou, M. (2013), 'Student modeling approaches: A literature review for the last decade', *Expert Systems with Applications* **40**(11), 4715–4729. doi: 10.1016/j.eswa.2013.02.007

Cicconet, M., Bretan, M. and Weinberg, G. (2013), 'Human-Robot Percussion Ensemble: Anticipation on the Basis of Visual Cues', *IEEE Robotics & Automation Magazine* **20**(4), 105–110. doi: 10.1109/MRA.2013.2256323

Crick, C., Munz, M. and Scassellati, B. (2006), Synchronization in Social Tasks: Robotic Drumming, *in* 'Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication', pp. 97–102. doi: 10.1109/ROMAN.2006.314401

Dautenhahn, K., Woods, S., Kaouri, C., Walters, M. L., Lee Koay, K. and Werry, I. (2005), What is a robot companion - friend, assistant or butler?, *in* 'Proceedings of

the IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 1192–1197. doi: 10.1109/IROS.2005.1545189

Demiris, Y. (2007), 'Prediction of intent in robotics and multi-agent systems', *Cognitive processing* **8**(3), 151–158. doi: 10.1007/s10339-007-0168-9

Demiris, Y. (2009), Knowing when to assist: Developmental issues in lifelong assistive robotics, *in* 'Proceedings of the Annual Conference of the IEEE Engineering in Medicine and Biology Society', pp. 3357–3360. doi: 10.1109/IEMBS.2009.5333182

Demiris, Y. and Hayes, G. M. (2002), Imitation as a Dual Route Process Featuring Predictive and Learning Components: a Biologically-Plausible Computational Model, *in* 'Imitation in Animals and Artifacs', MIT Press, chapter 13, pp. 327–361.

Demiris, Y. and Khadhouri, B. (2006), 'Hierarchical attentive multiple models for execution and recognition of actions', *Robotics and Autonomous Systems* **54**(5), 361–369. doi: 10.1016/j.robot.2006.02.003

Demiris, Y. and Meltzoff, A. (2008), 'The robot in the crib: a developmental analysis of imitation skills in infants and robots', *Infant and Child Development* **17**, 43–53. doi: 10.1002/icd.543

D'Mello, S., Olney, A., Williams, C. and Hays, P. (2012), 'Gaze tutor: A gaze-reactive intelligent tutoring system', *International Journal of Human Computer Studies* **70**(5), 377–398. doi: 10.1016/j.ijhcs.2012.01.004

Earley, J. (1970), 'An efficient context-free parsing algorithm', *Communications of the ACM* **13**(2), 94–102. doi: 10.1145/362007.362035

Ebcioglu, K. (1986), An Expert System for Chorale Harmonization, *in* 'Proceedings of the AAAI', pp. 784–788.

European Commission (2012), 'Action Plan for the EU Health Workforce'.

Evans, S., Neophytou, C., de Souza, L. and Frank, A. O. (2007), 'Young people's experiences using electric powered indoor - outdoor wheelchairs (EPIOCs): potential for enhancing users' development?', *Disability and rehabilitation* **29**(16), 1281–1294. doi: 10.1080/09638280600964406

Fasola, J. and Mataric, M. J. (2013), 'A Socially Assistive Robot Exercise Coach for the Elderly', *Journal of Human-Robot Interaction* **2**(2), 3–32. doi: 10.5898/JHRI.2.2.Fasola

Ferrari, E., Robins, B. and Dautenhahn, K. (2009), Therapeutic and educational objectives in robot assisted play for children with autism, *in* 'Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication', pp. 108–114. doi: 10.1109/ROMAN.2009.5326251

Fong, T., Nourbakhsh, I. and Dautenhahn, K. (2003), 'A survey of socially interactive robots', *Robotics and Autonomous Systems* **42**(3-4), 143–166. doi: 10.1016/S0921-8890(02)00372-X

Fong, T., Thorpe, C. and Baur, C. (2003), Collaboration, dialogue, human-robot interaction, *in* 'Robotics Research', Vol. 6 of *Tracts in Advanced Robotics*, Springer, pp. 255–266. doi: 10.1007/3-540-36460-9_17

Forchhammer, B., Papenbrock, T., Stening, T., Viehmeier, S., Draisbach, U. and Naumann, F. (2013), Duplicate Detection on GPUs, *in* 'Proceedings of the Conference on Database Systems for Business, Technology, and Web', pp. 165–184.

Frennert, S., Östlund, B. and Eftring, H. (2012), Would Granny Let an Assistive Robot into Her Home?, *in* 'Social Robotics', Vol. 7621 of *Lecture Notes in Computer Science*, Springer, pp. 128–137. doi: 10.1007/978-3-642-34103-8_13

Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M. and Weld, D. S. (2008), Predictability and accuracy in adaptive user interfaces, *in* 'Proceeding of Annual Conference on Human Factors in Computing Systems', ACM, pp. 1271–1274. doi: 10.1145/1357054.1357252

Gertner, A. S. and VanLehn, K. (2000), Andes: A coached problem solving environment for physics, *in* 'Intelligent Tutoring Systems', Vol. 1839 of *Lecture Notes in Computer Science*, Springer, pp. 133–142. doi: 10.1007/3-540-45108-0_17

Goetz, J., Kiesler, S. and Powers, A. (2003), Matching robot appearance and behavior to tasks to improve human-robot cooperation, *in* 'Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication', pp. 55–60. doi: 10.1109/ROMAN.2003.1251796

Graesser, A. C., Chipman, P., Haynes, B. C. and Olney, A. (2005), 'AutoTutor: an intelligent tutoring system with mixed-initiative dialogue', *IEEE Transactions on Education* **48**(4), 612–618. doi: 10.1109/TE.2005.856149

Gross, H.-M., Schroeter, C., Mueller, S., Volkhardt, M., Einhorn, E., Bley, A., Langner, T., Martin, C. and Merten, M. (2011), I'll keep an eye on you: Home robot companion for elderly people with cognitive impairment, *in* 'Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics', pp. 2481–2488. doi: 10.1109/ICSMC.2011.6084050

Hall, D., Berg-Kirkpatrick, T., Canny, J. and Klein, D. (2014), Sparser, Better, Faster GPU Parsing, *in* 'Proceedings of the Annual Meeting of the Association for Computational Linguistics'.

Harris, M., Sengupta, S. and Owens, J. D. (2007), Parallel Preix Sum (Scan) with CUDA, *in* 'GPU Gems 3', Addison-Wesley, chapter 39.

Hoffman, G. and Breazeal, C. (2007), 'Cost-based anticipatory action selection for human-robot fluency', *IEEE Transactions on Robotics* **23**(5), 952–961. doi: 10.1109/TRO.2007.907483

Hoffman, G. and Weinberg, G. (2010), Gesture-based human-robot jazz improvisation, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', pp. 582–587. doi: 10.1109/ROBOT.2010.5509182

Huntemann, A., Demeester, E., Vander Poorten, E., Van Brussel, H. and De Schutter, J. (2013), Probabilistc approach to recognize local navigation plans by fusing past driving information with a personalized user model, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', pp. 4376–4383. doi: 10.1109/ICRA.2013.6631197

Ioannidis, J. P. A. (2005), 'Why most published research findings are false', *PLoS Medicine* **2**(8), 0696–0701. doi: 10.1371/journal.pmed.0020124

Ivanov, Y. A. and Bobick, A. (2000), 'Recognition of Visual Activities and Interactions by Stochastic Parsing', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(8), 852–872. doi: 10.1109/34.868686

Johnson, M. and Demiris, Y. (2004), Abstraction in Recognition to Solve the Correspondence Problem for Robot Imitation, *in* 'Proceedings of the Conference Towards Autonomous Robotics Systems (TAROS)', pp. 63–70.

Kaltenbrunner, M. and Bencina, R. (2007), reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction, *in* 'Proceedings of the Inter-

national Conference on Tangible and Embedded Interaction', ACM, pp. 69–74. doi: 10.1145/1226969.1226983

Kanda, T., Hirano, T., Eaton, D. and Ishiguro, H. (2004), 'Interactive Robots as Social Partners and Peer Tutors for Children: A Field Trial', *Human-Computer Interaction* **19**(1), 61–84.

Karniel, A. (2002), 'Three creatures named forward model', *Neural Networks* **15**(3), 305–307. doi: 10.1016/S0893-6080(02)00020-5

Keates, S., Langdon, P., Clarkson, P. and Robinson, P. (2002), 'User Models and User Physical Capability', *User Modeling and User-Adapted Interaction* **12**(2-3), 139–169. doi: 10.1023/A:1015047002796

Kennedy, J., Baxter, P. and Belpaeme, T. (2015), The Robot Who Tried Too Hard: Social Behaviour of a Robot Tutor Can Negatively Affect Child Learning, *in* 'Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction', pp. 67–74. doi: 10.1145/2696454.2696457

Kidd, C. D. and Breazeal, C. (2004), Effect of a robot on user perceptions, *in* 'Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3559–3564. doi: 10.1109/IROS.2004.1389967

Kitani, K. M. and Koike, H. (2010), ImprovGenerator : Online Grammatical Induction for On-the-Fly Improvisation Accompaniment, *in* 'Proceedings of the Conference on New Interfaces for Musical Expression', pp. 469–472.

Klein, D. and Manning, C. D. (2003), Accurate Unlexicalized Parsing, *in* 'Proceedings of the Meeting of the Association for Computational Linguistics', ACM, pp. 423–430. doi: 10.3115/1075096.1075150

Klein, T., Gelderblom, G. J., de Witte, L. and Vanstipelen, S. (2011), Evaluation of short term effects of the IROMEC robotic toy for children with developmental disabilities, *in* 'Proceedings of the IEEE International Conference on Rehabilitation Robotics', pp. 1–5. doi: 10.1109/ICORR.2011.5975406

Krüger, V., Kragic, D., Ude, A. and Geib, C. (2007), 'The meaning of action: a review on action recognition and mapping', *Advanced Robotics* **21**(13), 1473–1501. doi: 10.1163/156855307782148578

Kruijff-Korbayová, I., Oleari, E., Baroni, I., Kiefer, B., Zelati, M. C., Pozzi, C. and Sanna, A. (2014), Effects of Off-Activity Talk in Human-Robot Interaction with Diabetic Children, *in* 'Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication', pp. 649–654. doi: 10.1109/ROMAN.2014.6926326

Lee, K., Kim, T.-K. and Demiris, Y. (2012), Learning Reusable Task Components using Hierarchical Activity Grammars with Uncertainties, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', pp. 1994–1999. doi: 10.1109/ICRA.2012.6224667

Lee, K., Su, Y., Kim, T.-K. and Demiris, Y. (2013), 'A syntactic approach to robot imitation learning using probabilistic activity grammars', *Robotics and Autonomous Systems* **61**(12), 1323–1334. doi: 10.1016/j.robot.2013.08.003

Leite, I., Martinho, C., Pereira, A. and Paiva, A. (2009), As Time goes by: Long-term evaluation of social presence in robotic companions, *in* 'Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication', pp. 669–674. doi: 10.1109/ROMAN.2009.5326256

López, N., Nuñez, M., Rodríguez, I. and Rubio, F. (2002), WHAT: Web-based Haskell adaptive tutor, *in* 'Artificial Intelligence: Methodology, Systems and Applications', Vol. 2443 of *Lecture Notes in Computer Science*, Springer, pp. 71–80. doi: 10.1007/3-540-46148-5_8

Lowenthal, M. F. (1964), 'Social Isolation and Mental Illness in Old Age', *American Sociological Review* **29**, 54. doi: 10.2307/2094641

Lu, S. C., Blackwell, N. and Do, E. Y.-L. (2011), mediRobbi: An interactive companion for pediatric patients during hospital visit, *in* 'Human-Computer Interaction. Interaction Techniques and Environments', Vol. 6762 of *Lecture Notes in Computer Science*, Springer, pp. 547–556. doi: 10.1007/978-3-642-21605-3_60

Marchal-Crespo, L., Furumasu, J. and Reinkensmeyer, D. J. (2010), 'A robotic wheelchair trainer: design overview and a feasibility study', *Journal of Neuroengineering and Rehabilitation* **7**(40). doi: 10.1186/1743-0003-7-40

Markopoulos, P., Read, J. C., MacFarlane, S. and Hoysniemi, J. (2008), *Evaluating Children's Interactive Products: Principles and Practices for Interaction Designers*, ACM.

Marti, P. and Giusti, L. (2010), A Robot Companion for Inclusive Games: a user-centred design perspective, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', pp. 4348–4353. doi: 10.1109/ROBOT.2010.5509385

Martin, B. and Mitrovic, A. (2002), Automatic problem generation in constraint-based tutors, *in* 'Intelligent Tutoring Systems', Vol. 2363 of *Lecture Notes in Computer Science*, Springer, pp. 388–398. doi: 10.1007/3-540-47987-2_42

Martins, M. F. and Demiris, Y. (2010), Learning multirobot joint action plans from simultaneous task execution demonstrations, *in* 'Proceedings of the International Conference on Autonomous Agents and Multiagent Systems', pp. 931–938.

Massengale, S., Folden, D., McConnell, P., Stratton, L. and Whitehead, V. (2005), 'Effect of visual perception, visual function, cognition, and personality on power wheelchair use in adults', *Assistive Technology* **17**(2), 108–21. doi: 10.1080/10400435.2005.10132101

McColl, D. and Nejat, G. (2013), 'Meal-Time with a Socially Assistive Robot and Older Adults at a Long-term Care Facility', *Journal of Human-Robot Interaction* **2**(1), 152–171. doi: 10.5898/JHRI.2.1.McColl

Michalowski, M. P., Sabanovic, S. and Kozima, H. (2007), A dancing robot for rhythmic social interaction, *in* 'Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction', pp. 89–96. doi: 10.1145/1228716.1228729

Minnen, D., Essa, I. and Starner, T. (2003), Expectation Grammars: Leveraging High-Level Expectations for Activity Recognition, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 626–632. doi: 10.1109/CVPR.2003.1211525

Mitrovic, A. (2003), 'An Intelligent SQL Tutor on the Web', *International Journal of Artificial Intelligence in Education* **13**(2-4), 173–197.

Montesano, L., Díaz, M., Bhaskar, S. and Minguez, J. (2010), 'Towards an intelligent wheelchair system for users with cerebral palsy', *IEEE Transactions on Neural Systems and Rehabilitation Engineering* **18**(2), 193–202. doi: 10.1109/TNSRE.2009.2039592

Morales, Y., Kallakuri, N., Shinozawa, K., Miyashita, T. and Hagita, N. (2013), Human-comfortable navigation for an autonomous robotic wheelchair, *in* 'Proceedings of the IEEE International Conference on Intelligent Robots and Systems', pp. 2737–2743. doi: 10.1109/IROS.2013.6696743

Nadrag, P., Temzi, L., Arioui, H. and Hoppenot, P. (2011), Remote control of an assistive robot using force feedback, *in* 'Proceedings of the International Conference on Advanced Robotics', pp. 211–216. doi: 10.1109/ICAR.2011.6088570

Nikolaidis, S., Ramakrishnan, R., Gu, K. and Shah, J. (2015), Efficient Model Learning for Human-Robot Collaborative Tasks, *in* 'Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction', pp. 189–196. doi: 10.1145/2696454.2696455

Nuzzo, R. (2014), 'Scientific method: Statistical errors', *Nature* **506**(7487), 150–152. doi: 10.1038/506150a

Ogale, A., Karapurkar, A. and Aloimonos, Y. (2007), View-invariant modeling and recognition of human actions using grammars, *in* 'Dynamical Vision', Vol. 4358 of *Lecture Notes in Computer Science*, Springer, pp. 115–126. doi: 10.1007/978-3-540-70932-9_9

Ognibene, D., Chinellato, E., Sarabia, M. and Demiris, Y. (2012), Towards contextual action recognition and target localization with active allocation of attention, *in* 'Biomimetic and Biohybrid Systems', Vol. 7375 of *Lecture Notes in Computer Science*, Springer, pp. 192–203. doi: 10.1007/978-3-642-31525-1_17

Ognibene, D., Chinellato, E., Sarabia, M. and Demiris, Y. (2013), Contextual action recognition and target localization with active allocation of attention, *in* 'Bioinspiration & Biomimetics', Vol. 8, IOP Publishing, p. 035002. doi: 10.1088/1748-3182/8/3/035002

Park, E., Kim, K. J. and del Pobil, A. P. (2011), The Effects of a Robot Instructor's Positive vs Negative Feedbacks on Attraction and Acceptance towards the Robot in Classroom, *in* 'Social Robotics', Vol. 7072 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 135–141. doi: 10.1007/978-3-642-25504-5_14

Plaisant, C., Druin, A., Lathan, C., Dakhane, K., Edwards, K., Vice, J. M. and Montemayor, J. (2000), A storytelling robot for pediatric rehabilitation, *in* 'Proceedings of the ACM International Conference on Assistive Technologies', pp. 50–55. doi: 10.1145/354324.354338

Quick, D. and Hudak, P. (2013), A Temporal Generative Graph Grammar for Harmonic and Metrical Structure, *in* 'Proceedings of the International Computer Music Conference', International Computer Music Association, pp. 177–184.

Quigley, M., Gerkey, B. P., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng, A. (2009), ROS : an open-source Robot Operating System, *in* 'Proceedings of the Open Source Software Workshop at the International Conference of Robotics and Automation'.

Riek, L. (2012), 'Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines', *Journal of Human-Robot Interaction* **1**(1), 119–136. doi: 10.5898/JHRI.1.1.Riek

Rizzolatti, G. and Craighero, L. (2004), 'The mirror-neuron system', *Annual Review of Neuroscience* **27**, 169–92. doi: 10.1146/annurev.neuro.27.070203.144230

Robinson, H., Broadbent, E. and MacDonald, B. (2015), 'Group sessions with Paro in a nursing home: Structure, observations and interviews', *Australasian Journal on Ageing* . doi: 10.1111/ajag.12199

Robinson, H., MacDonald, B., Kerse, N. and Broadbent, E. (2013), 'The Psychosocial Effects of a Companion Robot: A Randomized Controlled Trial', *Journal of the American Medical Directors Association* **14**(9), 661–667. doi: 10.1016/j.jamda.2013.02.007

Ros, R., Baroni, I. and Demiris, Y. (2014), 'Adaptive human-robot interaction in sensorimotor task instruction: From human to robot dance tutors', *Robotics and Autonomous Systems* **62**(6), 707–720. doi: 10.1016/j.robot.2014.03.005

Rosen, L., Arva, J., Furumasu, J., Harris, M., Lange, M. L., McCarthy, E., Kermoian, R., Pinkerton, H., Plummer, T., Roos, J., Sabet, A., Vander Schaaf, P. and Wonsettler, T. (2009), 'RESNA position on the application of power wheelchairs for pediatric users.', *Assistive Technology* **21**(4), 218–225. doi: 10.1080/10400430903246076

Ryoo, M. and Aggarwal, J. (2006), Recognition of Composite Human Activities through Context-Free Grammar based Representation, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 1709–1718. doi: 10.1109/CVPR.2006.242

Sakakibara, Y., Brown, M., Hughey, R., Mian, I. S., Sjölander, K., Underwood, R. C. and Haussler, D. (1994), 'Stochastic context-free grammars for tRNA modeling', *Nucleic Acids Research* **22**(23), 5112–5120. doi: 10.1093/nar/22.23.5112

Sanderson, D. and Pitt, J. (2011), An Affective Anticipatory Agent Architecture, *in* 'Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology', pp. 93–96. doi: 10.1109/WI-IAT.2011.178

Sarabia, M. and Demiris, Y. (2013), A Humanoid Robot Companion for Wheelchair Users, *in* 'Social Robotics', Vol. 8239 of *Lecture Notes in Computer Science*, Springer, pp. 432–441. doi: 10.1007/978-3-319-02675-6_43

Sarabia, M., Lee, K. and Demiris, Y. (2015), Towards a Synchronised Grammars Framework for Adaptive Musical Human-Robot Collaboration, *in* 'Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication', pp. 715–721. doi: 10.1109/ROMAN.2015.7333649

Sarabia, M., Ros, R. and Demiris, Y. (2011), Towards an open-source social middleware for humanoid robots, *in* 'Proceedings of the IEEE/RAS International Conference on Humanoid Robots', pp. 670–675. doi: 10.1109/Humanoids.2011.6100883

Schneider, S., Berger, I., Riether, N., Wrede, S. and Wrede, B. (2012), Effects of Different Robot Interaction Strategies During Cognitive Tasks, *in* 'Social Robotics', Vol. 7621 of *Lecture Notes in Computer Science*, Springer, pp. 496–505. doi: 10.1007/978-3-642-34103-8_50

Self, J. (1990), 'Bypassing the intractable problem of student modelling', *Intelligent Tutoring Systems: At the crossroads of artificial intelligence and education* (41), 107–123.

Shah, A., Herbert, R., Lewis, S., Mahendran, R., Platt, J. and Bhattacharyya, B. (1997), 'Screening for depression among acutely ill geriatric inpatients with a short geriatric depression scale', *Age and Ageing* **26**, 217–221. doi: 10.1093/ageing/26.3.217

Shah, J., Wiken, J., Williams, B. and Breazeal, C. (2011), Improved human-robot team performance using chaski, a human-inspired plan execution system, *in* 'Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction', pp. 29–36. doi: 10.1145/1957656.1957668

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A. and Blake, A. (2011), Real-time human pose recognition in parts from single depth images, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', IEEE, pp. 1297–1304. doi: 10.1109/CVPR.2011.5995316

Simpson, R. C. (2005), 'Smart wheelchairs: A literature review', *The Journal of Rehabilitation Research and Development* **42**(4), 423–438. doi: 10.1682/JRRD.2004.08.0101

Simpson, R. C., LoPresti, E. F. and Cooper, R. A. (2008), 'How many people would benefit from a smart wheelchair?', *The Journal of Rehabilitation Research and Development* **45**(1), 53–72. doi: 10.1682/JRRD.2007.01.0015

Soh, H. and Demiris, Y. (2012), Towards Early Mobility Independence : An Intelligent Paediatric Wheelchair with Case Studies, *in* 'Proceedings of the IROS 2012 Workshop on Progress, Challenges and Future Perspectives in Navigation and Manipulation Assistance for Robotic Wheelchairs'.

Soh, H. and Demiris, Y. (2014), 'Spatio-Temporal Learning With the Online Finite and Infinite Echo-State Gaussian Processes', *IEEE Transactions on Neural Networks and Learning Systems* **26**(3), 522–536. doi: 10.1109/TNNLS.2014.2316291

Starner, T. and Pentland, A. (1995), Real-time American Sign Language recognition from video using hidden Markov models, *in* 'Proceedings of the International Symposium on Computer Vision', IEEE, pp. 265–270. doi: 10.1109/ISCV.1995.477012

Stolcke, A. (1994), Bayesian Learning of Probabilistic Language Models, PhD thesis, University of California at Berkeley.

Stolcke, A. (1995), 'An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities', *Computational Linguistics* **21**(2), 165–201.

Storfjell, J. L., Omoike, O. and Ohlson, S. (2008), 'The balancing act: patient care time versus cost', *The Journal of Nursing Administration* **38**(5), 244–249. doi: 10.1097/01.NNA.0000312771.96610.df

Sung, J.-Y., Christensen, H. I. and Grinter, R. E. (2009), Robots in the Wild: Understanding Long-term Use, *in* 'Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction', p. 45. doi: 10.1145/1514095.1514106

Sutton, R. S. and Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, MIT Press.

Taberner, P. V. (1980), 'Sex differences in the effects of low doses of ethanol on human reaction time', *Psychopharmacology* **70**(3), 283–286. doi: 10.1007/BF00427886

Takayanagi, K., Kirita, T. and Shibata, T. (2014), 'Comparison of verbal and emotional responses of elderly people with mild/moderate dementia and those with severe dementia in responses to seal robot, PARO', *Frontiers in Aging Neuroscience* **6**. doi: 10.3389/fnagi.2014.00257

Tapus, A., Tapus, C. and Mataric, M. J. (2009), The use of socially assistive robots in the design of intelligent cognitive therapies for people with dementia, *in* 'Proceedings of the IEEE International Conference on Rehabilitation Robotics', pp. 23–26. doi: 10.1109/ICORR.2009.5209501

Tidemann, A. and Demiris, Y. (2008), Groovy Neural Networks, *in* 'Proceedings of the European Conference on Artificial Intelligence', Vol. 178, IOS Press, pp. 271–275. doi: 10.3233/978-1-58603-891-5-271

Trumbly, J. E., Arnett, K. P. and Martin, M. P. (1993), 'Performance effect of matching computer interface characteristics and user skill level'. doi: 10.1006/imms.1993.1033

van der Vaart, E. and Hemelrijk, C. K. (2014), ''Theory of mind' in animals: ways to make progress', *Synthese* **191**(3), 335–354. doi: 10.1007/s11229-012-0170-3

VanLehn, K. (2006), 'The Behavior of Tutoring Systems', *International Journal of Artificial Intelligence in Education* **16**(3), 227–265.

VanLehn, K. (2011), 'The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems', *Educational Psychologist* **46**(4), 197–221. doi: 10.1080/00461520.2011.611369

Vygotsky, L. S. (1978), *Mind in society: The development of higher psychological processes*, Harvard University Press.

Wall, M. and Duffy, A. (2010), 'The effects of music therapy for older people with dementia', *The British Journal of Nursing* **19**(2), 108–113. doi: 10.12968/bjon.2010.19.2.46295

Weir, M., Aggarwal, S., de Medeiros, B. and Glodek, B. (2009), Password Cracking Using Probabilistic Context-Free Grammars, *in* 'Proceedings of the IEEE Symposium on Security and Privacy', pp. 391–405. doi: 10.1109/SP.2009.8

Wolpert, D. M., Miall, R. C. and Kawato, M. (1998), 'Internal models in the cerebellum', *Trends in cognitive sciences* **2**(9), 338–47. doi: 10.1016/S1364-6613(98)01221-2

Xia, L., Chen, C.-C. and Aggarwal, J. K. (2012), View invariant human action recognition using histograms of 3D joints, *in* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops', IEEE, pp. 20–27.

Yamato, J., Ohya, J. and Ishii, K. (1992), Recognizing human action in time-sequential images using hidden Markov model, *in* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition', IEEE, pp. 379–385. doi: 10.1109/CVPR.1992.223161

Younger, D. H. (1967), 'Recognition and parsing of context-free languages in time n3', *Information and Control* **10**(2), 189–208. doi: 10.1016/S0019-9958(67)80007-X

Zhang, Z., Tan, T. and Huang, K. (2011), 'An extended grammar system for learning and recognizing complex visual events', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**(2), 240–255. doi: 10.1109/TPAMI.2010.60

AUTHOR'S PUBLICATIONS

This appendix is a compilation of all our peer-reviewed publications. For each article, we provide the full reference, a short summary as well as its contribution to the thesis.

Sarabia, M., Young, N., Canavan, K., Edginton, T., Demiris, Y. and Vizcaychipi, M.P., Assistive Robotic Technology to Combat Social Isolation in Acute Hospital Settings, in 'Journal of Social Robotics', Springer, [submitted].

- Presents a study with Wizard of Oz (WoZ) NAO companion and 49 patients in a hospital in London.
- Section 4.2 is based on this article.

Sarabia, M., Lee, K. and Demiris, Y. (2015), Towards a Synchronised Grammars Framework for Adaptive Musical Human-Robot Collaboration, in 'Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication', pp. 715–-721. doi: 10.1109/ROMAN.2015.7333649.

- Describes the synchronised grammars framework and a user study in human-robot musical collaboration with Baxter.
- Sections 3.3 and 5.2 are based on this article.

Sarabia, M. and Demiris, Y. (2013), A Humanoid Robot Companion for Wheelchair Users in 'Social Robotics', Vol. 8239 of Lecture Notes in Computer Science, Springer, pp. 432–441. doi: 10.1007/978-3-319-02675-6_43.

- Introduces an implementation of a humanoid companion for wheelchairs users and two user studies with children and adults.
- Section 4.1 is based on this article.

Sarabia, M., Le Mau, T., Soh, H., Naruse, S., Poon, C., Liao, Z., Tan, K. C., Lai, Z. J. and Demiris, Y. (2013), iCharibot: Design and Field Trials of a Fundraising Robot, in 'Social Robotics', Vol. 8239 of Lecture Notes in Computer Science, Springer, pp. 412–421. doi: 10.1007/978-3-319-02675-6_41.

- Reports on the implementation of iCharibot, a charity robot that raised funds for charity in a busy street in London interacting with 386 people.

Ognibene, D., Chinellato, E., *Sarabia, M.* and Demiris, Y. (2013), Contextual action recognition and target localization with active allocation of attention, in 'Bioinspiration & Biomimetics', Vol. 8, IOP Publishing, 035002.
doi: 10.1088/1748-3182/8/3/035002.

- Presents a visual attention control mechanism based on HAMMER with the inverse models implemented as Kalman filters. Reports on experiments against other attention control mechanisms.

- Section 3.2.1 describes, but is not based on, this article.

Ognibene, D., Chinellato, E., *Sarabia, M.* and Demiris, Y. (2012), Towards contextual action recognition and target localization with active allocation of attention, in 'Biomimetic and Biohybrid Systems', Vol. 7375 of Lecture Notes in Computer Science, Springer, pp. 193–203.
doi: 10.1007/978-3-642-31525-1_17.

- Shorter version of the journal article above, describes the same visual attention mechanism but lacks the evaluation of the mechanism.

Sarabia, M., Ros, R. and Demiris, Y. (2011), Towards an open-source social middleware for humanoid robots, in 'Proceedings of the IEEE/RAS International Conference on Humanoid Robots', pp. 670–675.
doi: 10.1109/Humanoids.2011.6100883.

- Introduces the HAMMER middleware and an example of how to use it to recognise dance steps.

- Sections 3.1 and 3.5 are based on this article.

This thesis was typeset with XeLaTeX using a style based on classicthesis by André Miede with our own modifications. The text's main font is Monotype's Perpetua. For titles and figures we use Monotype's Gill Sans, whereas computers listings are displayed with Adobe's `SourceCode Pro`. Finally, mathematical expressions were formatted with AMS's `Euler` font.

Figures were created with matplotlib, inkscape and gimp.