HAROLD SOON HONG SOH

ONLINE SPATIO-TEMPORAL LEARNING AND PREDICTION FOR ADAPTIVE ROBOTIC SYSTEMS

Applications in Tactile Classification and Learning Assistance by Demonstration

Thesis submitted for the degree of Doctor of Philosophy

Department of Electrical and Electronic Engineering
Imperial College London

Online Spatio-Temporal Learning and Prediction for Adaptive Robotic Systems, Applications in Tactile Classification and Learning Assistance by Demonstration.

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution-Non Commercial-No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or distribution, researchers must make clear to others the licence terms of this work.

For my family.

ABSTRACT

Successful biological systems adapt to change. Humans, for example, are capable of continual self-improvement and gain new skills with experience. Similar online learning characteristics would enable robotic systems to autonomously improve their capabilities over time. In this thesis, we focus on the problem of iteratively learning from multivariate time-series; the "raw material" that we use to make inferences about the future.

We adopt a combined approach: gaining inspiration from biological systems, in particular recurrent neural networks, and merging these ideas with recent advances in statistical machine learning. The resulting algorithm — the online echo-state Gaussian process (OESGP) — learns in an online manner, produces predictive distributions and attains state-of-the-art results on a variety of benchmark problems. We further extend this method to networks of "infinite size" through a recursive kernel with automatic relevance determination. This allows for online optimisation of the hyper-parameters through stochastic natural gradient descent, which improves adaptability and alleviates the problem of reservoir parameter specification.

Using this online infinite ESGP (OIESGP) as a building block, we address two challenging problems in robotics: online tactile learning using the iCub humanoid platform and smart mobility assistance on the ARTY smart wheelchair. For the former, we develop online generative and discriminative classifiers that learn new objects "on-the-fly" and refine older models with new sensory input. For the latter, we adopt a novel approach by applying imitation learning to derive assistive policies. We present an OIESGP-based probabilistic mixture model for learning when and how to appropriately assist, and demonstrate its effectiveness in simulation and real-world experiments with human subjects.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Yiannis Demiris. A supervisor makes or breaks the PhD experience and I'm very glad to have had Dr. Demiris's advice, encouragement and support[1] throughout the past four years.

Thank you to all the members of the BioART/Personal Robotics lab (alumni included) for all the great memories in-and-out of the lab: Simon Butler, Sotirios Chatzis, Eris Chinellato, Jonathan Claassens, Murilo Fernandes Martins, Yixing Gao, Theodosis Georgio, Dimitris Korkinof, Kyuhwa Lee, Dimitri Ognibene, Arturo Ribes, Raquel Ros Espinoza, Balint Takacs, and Paschalis Veskos. I would like to specially thank Yan Wu for introducing me to the lab all those years ago, Tom Carlson for his advice in the initial stages of my research and Miguel Sarabia for his assistance in upgrading the wheelchair. Among the visitors to the lab, I am grateful to Yanyu Su for his contribution in the tactile-learning experiments with the iCub and Alberto Jardón Huete for the hours we spent building electronic components together.

Thank you to the staff at Tadworth Children's Trust, particularly Carolyn Dunford and Michelle Holmes, for sharing their immense knowledge — without these remarkable individuals, the case-studies with special-needs children would not have been possible.

To my friends in ISN and Imperial, I'm especially grateful for all the times we've shared; from the thought-provoking discussions to the silly antics, I would have enjoyed my PhD far less without all of you being a part of it. I would also like to thank Dr. Sunny Bains for the fascinating lunch time discussions and for giving me a rewarding job as a Graduate Teaching Assistant.

1 To the point of volunteering his 6 year-old son as a preliminary experimental subject on a yet untested prototype smart wheelchair.

Thank you to Yayasan Khazanah for their financial support, which allowed me to pursue my dream of building intelligent robots. In addition, I would like to acknowledge Gary Hellings (HMS Mobility) and Andrew McLaren (Otto Bock, UK) for their help with the Skippi, and Jonathan Miller (Novint Technologies) for donating two Falcon controllers for the LAD experiments.

Those who know me would quickly attest to my dislike of administrative paperwork and I thank our group administrators, Shahareen Hilmy, Faye Gosling and Patrick Kelly, for their assistance in navigating the red tape.

My PhD would not have been possible without support and love from my parents, Chai Hock and Bee Lee, my brother Howard and my sister Joy.

Finally, to Hazel, thank you for your love, patience and for putting up with my "insane"[2] ideas.

2 Some might say "awesome".

DECLARATION

I hereby declare that I composed this thesis entirely by myself and that it describes my own research.

London, July 2013

Harold Soh

CONTENTS

1 INTRODUCTION 18

1.1 The Challenge of Online Learning for Robotic Systems 19

1.2 Thesis Contributions 19

1.3 Roadmap: Overview and Organisation 21

1.4 Mathematical Notation 24

I PRELIMINARIES 25

2 BACKGROUND: ONLINE MACHINE-LEARNING AND ECHO-STATE NETWORKS 26

2.1 Online Machine Learning for Robotics 27

2.1.1 Local Model Learning 28

2.1.2 Basis Functions for Projections to Feature Spaces 31

2.1.3 A Bayesian Treatment 32

2.1.4 Gaussian Processes 34

2.1.5 Online Learning with GPs 37

2.2 Echo State Networks 39

2.2.1 Online Training for ESNs 42

2.3 Summary, Connections and Moving Ahead 44

II ONLINE SPATIO-TEMPORAL MACHINE-LEARNING METHODS 46

3 ONLINE ECHO-STATE GAUSSIAN PROCESS 47

3.1 The Echo State Gaussian Process 48

3.2 The Online Echo State Gaussian Process 49

3.2.1 Bayesian Online Learning and the Projected Process Approximation 50

3.2.2 Maintaining Sparsity 52

3.2.3 Training Summary and Making Predictions 53

3.3 Empirical Results . . . 55
3.3.1 Implementation and Setup . . . 55
3.3.2 One-Step Prediction on Benchmark Problems . . . 56
3.4 Discussion . . . 60
3.4.1 Comparison to the Full ESGP . . . 61
3.4.2 Trading-off Computational Cost and Accuracy . . . 62
3.5 Activity Recognition with Generative Classification . . . 63
3.6 Conclusion . . . 65
4 ONLINE INFINITE ECHO-STATE GAUSSIAN PROCESS 68
4.1 The Infinite Echo State Gaussian Process . . . 70
4.1.1 Recursive Kernels . . . 71
4.1.2 Recursive Kernel with Input Relevance . . . 72
4.2 Online Hyperparameter Adaptation . . . 75
4.2.1 Stochastic Natural Gradient Descent . . . 76
4.2.2 An Illustrative Example . . . 80
4.3 Empirical Results on Benchmark Problems . . . 83
4.3.1 Performance Measurement . . . 84
4.3.2 Accuracy Results . . . 84
4.3.3 Computational Costs . . . 89
4.4 Discussion: A Dynamical Systems Perspective . . . 90
4.4.1 The Problem of False Neighbours . . . 92
4.5 Summary and Conclusion . . . 93

III APPLICATIONS IN ROBOTICS 94
5 ITERATIVELY LEARNING OBJECTS BY TOUCH 95
5.1 Related Work on Tactile-based Classification . . . 97
5.1.1 Discriminative Tactile Classifiers . . . 97
5.1.2 Generative Tactile Classifiers . . . 98
5.2 OIESGP Generative and Discriminative Tactile Classifiers . . . 99
5.2.1 Discriminative Tactile Classifier . . . 99
5.2.2 Generative Tactile Classifier . . . 101
5.3 Experimental Setup . . . 102

5.3.1 Test Objects . . . 102
5.3.2 Grasping with the iCub Hand . . . 103
5.3.3 Classifier Set-up . . . 106
5.4 Empirical Results . . . 107
5.4.1 Performance Comparison to Offline Classifiers . . . 109
5.4.2 Early Classification . . . 110
5.4.3 Computational Costs . . . 111
5.4.4 Performance Comparison to Human Subjects . . . 112
5.5 Summary and Conclusions . . . 114
6 LEARNING ASSISTANCE BY DEMONSTRATION 116
6.1 Learning Assistance by Demonstration: Problem Statement . . . 118
6.2 An Online Probabilistic Model for LAD . . . 119
6.2.1 Mixtures of OIESGP Experts . . . 120
6.2.2 Prediction and Control . . . 124
6.3 Simulation Experiment: Assisted Mobility . . . 124
6.3.1 Experimental Setup . . . 125
6.3.2 Model Setup and Parameters . . . 127
6.3.3 Simulation Results . . . 129
6.4 Real-World Experiment with ARTY and Haptic Controllers . . . 132
6.4.1 Robot Platform: ARTY Smart Wheelchair . . . 132
6.4.2 Experimental Setup . . . 137
6.4.3 Results: Lap Performance and Survey Results . . . 138
6.5 Conclusions . . . 143

IV CLOSING 145

7 SUMMARY, CONCLUSIONS AND FUTURE WORK 146
7.1 Summary of Contributions . . . 146
7.2 Towards the Future . . . 148
7.2.1 Changing Distributions and Non-stationarity . . . 148
7.2.2 Unconstrained Basis Vectors . . . 149
7.2.3 Noisy Inputs, Iterated Predictions and Heteroscedasticity . . . 150
7.2.4 Complex Hierarchical Architectures . . . 151

7.2.5 Large-scale Learning of Objects via Touch and Other Modalities . . . 151
7.2.6 Learning Complex Assistive Policies . . . 152

V APPENDIX 154

A MATHEMATICAL FORMULAE 155
A.1 Matrix Inversion Lemma . . . 155

B RECURSIVE ARD KERNEL DERIVATION 156

C PARAMETERS AND GRID-SEARCH BOUNDS FOR THE EXPERIMENTAL STUDIES 158
C.1 Delay-Line and Reservoir Memory Structures . . . 158
C.2 Algorithm Parameters . . . 159

D SUPPLEMENTARY BENCHMARK RESULTS 160

E SUPPLEMENTARY TACTILE EXPERIMENT RESULTS 166

F PRELIMINARY LAD EXPERIMENT (JOYSTICK CONTROL) 168
F.1 Empirical Results: Lap Performance . . . 168

G SUPPLEMENTARY LAD EXPERIMENT RESULTS 171

H CASE STUDIES WITH THE ARTY SMART WHEELCHAIR 173
H.1 Background . . . 174
H.1.1 Powered Wheelchairs for Young Children . . . 174
H.1.2 Smart Paediatric Wheelchairs . . . 175
H.2 Shared-Control on the ARTY Platform . . . 177
H.2.1 Obstacle Map . . . 178
H.3 Case Study with Able-bodied Participants . . . 178
H.3.1 Experimental Setup . . . 180
H.3.2 Results and Discussion . . . 181
H.4 Case Study: C, a 5-year old boy with special needs . . . 182
H.4.1 Results and Discussion . . . 183
H.5 Conclusions and Future Work . . . 186

BIBLIOGRAPHY 187

LIST OF FIGURES

Figure 1.1 Online Learning in Robotics 18
Figure 1.2 Thesis Roadmap 22
Figure 2.1 Local Model Learning 29
Figure 2.2 Squared-exponential Kernel 35
Figure 2.3 The Echo-State Network. 40
Figure 3.1 The Online Echo State Gaussian Process 47
Figure 3.2 NAE Convergence Profile 56
Figure 3.3 Benchmark Dynamical Systems 57
Figure 3.4 Error and Iteration Time Comparing OESGP to Competing Methods on Benchmark Problems 59
Figure 3.5 Comparison between OESGP and ESGP 61
Figure 3.6 OESGP's Computational Cost with Varying Basis Vector Set Sizes. 62
Figure 3.7 OESGP-based Generative Action Classifer 64
Figure 3.8 MSRAction3D Skeletal Dataset 66
Figure 4.1 Recursive Automatic Relevance Determination Kernel and Stochastic Natural Gradient Descent 68
Figure 4.2 SNGD Performance Sensitivity to Initial Hyperparameters 78
Figure 4.3 Online Hyperparameter Adaptation on the NARMA-10 System 80
Figure 4.4 Computational Cost of Online Hyperparameter Adaptation 82
Figure 4.5 RMSE Scores for the One-step Prediction and System Identification Benchmarks 85
Figure 4.6 RMSE Scores for the One-step Prediction and System Identification Benchmarks with Irrelevant Dimension 87
Figure 4.7 Coordinate Delay Embedding 91

Figure 4.8 Mutual Information of State-delay, Observation-delay and Reservoir Memory Structures . . . 92
Figure 5.1 OIESGP-based Tactile Classifier . . . 95
Figure 5.2 iCub with Test Objects . . . 103
Figure 5.3 Sample Tactile Sensor Spatio-temporal Blocks . . . 104
Figure 5.4 The iCub Hand . . . 105
Figure 5.5 iCub Hand and Grasping Motion . . . 105
Figure 5.6 OIESGP Online Classification Accuracy . . . 107
Figure 5.7 Post-Learning Characteristic Lengthscales for OIESGP Discriminative Tactile Classifier . . . 108
Figure 5.8 Early-Classification Accuracy . . . 111
Figure 5.9 Human Online Tactile Classification Experiment and Results . . . 112
Figure 6.1 Learning-Assistance-by-Demonstration (LAD) System. . . . 116
Figure 6.2 Kernel Clustering using Euclidean and Kernel Centers . . . 121
Figure 6.3 Driving Course for Simulation LAD trials . . . 126
Figure 6.4 LAD Model Accuracy Performance Measures . . . 128
Figure 6.5 Lap times in Seconds by Simulated Driver Categorised by Mode . . . 129
Figure 6.6 Lengthscales for the LAD Simulation Experiment . . . 130
Figure 6.7 The Assistive Robotic Transport for Youngsters (ARTY) Smart Wheelchair . . . 131
Figure 6.8 High-level schematic of ARTY's Software Components . . . 133
Figure 6.9 Novint Falcon Haptic Controller on the ARTY wheelchair 135
Figure 6.10 Driving Course used in the Haptic-Control LAD Experiment . . . 137
Figure 6.11 LAD Experiment Survey Questions . . . 138
Figure 6.12 Lap times in Seconds for Participants Categorised by Mode 139
Figure 6.13 Driven Paths and Normalised Smoothed Density Plots . . . 139
Figure 6.14 LAD Experiment Survey Responses (Q1 to Q4) . . . 141
Figure 6.15 Survey Responses for the LAD Experiment (Q1 to Q11) . . . 142

Figure E.1 Confusion Matrices for the Online Tactile Classification Experiment (All Features) . . . 166
Figure E.2 Confusion Matrices for the Online Tactile Classification Experiment (Tactile Features) . . . 167
Figure E.3 Confusion Matrices for the Online Tactile Classification Experiment (Encoder Features) . . . 167
Figure F.1 Lap times for LAD Experiment with Joystick Control . . . 169
Figure F.2 Driven Paths and Normalised Smoothed Density Plots (Preliminary LAD Experiment) . . . 170
Figure H.1 Obstacle Course for Forwards and Backwards Driving Task 179
Figure H.2 Task-completion Times for the Forwards and Backwards Driving Task . . . 179
Figure H.3 Post-trial Questionnaire Results . . . 180
Figure H.4 Experimental Area and C's Driving Route. . . . 182
Figure H.5 Data Analysis for C's Driving Session . . . 183
Figure H.6 Predicted Crash Positions . . . 184
Figure H.7 Joystick Profiles for C and the Expert Driver . . . 184

LIST OF TABLES

Table 3.1 ESN and Algorithm Experimental Parameters . . . 54
Table 3.2 Benchmark Dynamical Systems . . . 55
Table 3.3 Error Scores for the One-Step Prediction Task on Benchmark and Noisy Problems . . . 59
Table 3.4 NMSE Scores for the One-Step Prediction Task on Noisy Problems . . . 59
Table 3.5 Accuracy scores for the MSRAction3D dataset. . . . 65
Table 4.1 Error Scores for Benchmarks . . . 86

Table 4.2 Error Scores for Benchmarks with Irrelevant Dimension. . 88
Table 5.1 Object Classification Scores for Online and Offline Methods 109
Table 5.2 Features Computed for Offline Methods 109
Table 5.3 Confusion Matrix for Full, Half-Full and Empty Plastic Bottles (Human Study) 113
Table 6.1 Assistive Model Parameters 127
Table 6.2 PD Parameters for the Falcon Haptic Devices 136
Table C.1 Algorithm Parameters for Benchmark Experiments 159
Table D.1 Wilcoxon Rank Sum Test p-values for the RMSE Distributions on the One-step Prediction Tasks 161
Table D.2 Wilcoxon Rank Sum Test p-values for the RMSE Distributions on the One-Step Prediction Tasks with Irrelevant Dimension 162
Table D.3 Wilcoxon Rank Sum Test p-values for the RMSE Distributions on the System Identification Tasks 163
Table D.4 Wilcoxon Rank Sum Test p-values for the RMSE Distributions on the System Identification Tasks with Irrelevant Dimensions 163
Table D.5 Training and Prediction CPU Time for Benchmarks 164
Table D.6 Training and Prediction CPU Time for Benchmarks with Irrelevant Dimension 165
Table G.1 Survey Responses from 15 participants for Baseline and Limited Control Laps 171
Table G.2 Survey Responses from 15 participants for Human and Model Assistance Laps 172
Table G.3 Survey χ^2 p-Values 172

ACRONYMS

ARD	Automatic Relevance Determination
ARTY	Assistive Robotic Transport for Youngsters
AUC	Area-under-curve score
BV	Basis Vector
CAN	Controller-area network
CTRNN	Continuous time RNNs
DWA	Dynamic Window Approach
ESGP	Echo-state Gaussian Process
ESN	Echo-state Network
GMM	Gaussian Mixture Model
GP	Gaussian Process
HAMMER	Hierarchical, Attentive, Multiple Models for Execution and Recognition
HM-OIESGP	Hierarchical Mixture of Online Infinite Echo-State Gaussian Processes
HSC	Hybrid Shared Control
KL	Kullback-Leibler (divergence)
KLMS	Kernel Least Mean Squares
KNLMS	Kernel Normalised Least Mean Squares
KRLS	Kernel Recursive Least Squares
LAD	Learning Assistance by Demonstration
LbD	Learning by Demonstration

LGP	Local Gaussian Process
LHS	Left hand side
LSTM	Long-short-term-memory
LWPR	Locally-Weighted Process Regression
ME	Mixture of Experts
ML	Maximum likelihood
MNAE	Mean normalised average error
NARMA	Nonlinear Auto-Regressive Moving Average
NLPD	Negative log predictive density
NORMA	Naïve Online R_{reg} Minimisation Algorithm
OESGP	Online Echo-state Gaussian Process
OIESGP	Online Infinite Echo-state Gaussian Process
OT	Occupational Therapist
PCA	Principal Component Analysis
PLS	Partial Least Squares
RD	Robot driver
RHS	Right hand side
RLS	Recursive Least Squares
RMSE	Root-mean squared error
RNN	Recurrent Neural Network
ROS	Robot Operating System
SGD	Stochastic Gradient Descent
SOGP	Sparse Online Gaussian Process
SOM	Self-Organising Map
SSP	Sparse sequential projection algorithm
SVM	Support Vector Machine
VFH	Vector Field Histogram

1

INTRODUCTION

One attribute of successful biological systems is the ability to adapt to changing physical and environmental conditions. This capacity can be exhibited by a short life-cycle with rapid mutations, or in the case of the human species, the continuous development of skills and knowledge throughout life. To be successful, robotic systems acting in similarly unconstrained environments will require such *online*, *lifelong-learning* characteristics.

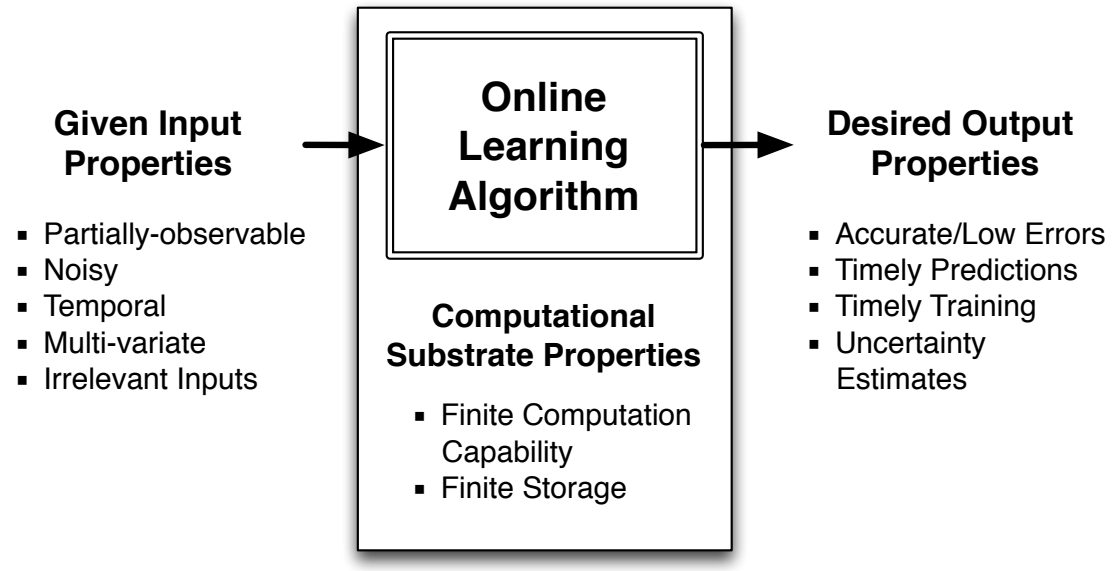


Figure 1.1: Robotics problems typically consist of noisy multi-variate data streams from which we have to extract accurate predictions in a timely (often real-time) manner. Moreover, we are restricted by limited computation and storage resources, particularly on mobile robots. In this thesis, we aim to address these issues through a novel combination of echo-state networks and Bayesian online learning.

1.1 THE CHALLENGE OF ONLINE LEARNING FOR ROBOTIC SYSTEMS

In the robotics domain, this ability to learn online — improve behaviour and extend capabilities with experience — is prized but at the same time, challenging to actualise. As illustrated in Fig 1.1, robotic systems have to process inputs with are not only noisy partial observations of the state, but also temporal and multivariate with irrelevant dimensions. From this "raw material", our learners are expected to produce predictions that are accurate and timely, ideally with uncertainty estimates to mitigate high-loss scenarios. All this has to be performed on a computational substrate with finite processing and storage resources.

Our primary task in this thesis is to address these issues, which we undertake via a combination of online statistical machine-learning and biologically-inspired recurrent neural networks. As a result, we derive two novel supervised-learning methods: the Online Finite and Infinite Echo-State Gaussian Processes. Both methods operate on noisy multi-variate time-series data on a fixed maximum computational and storage budget to produce high accuracies relative to state-of-the-art methods. Using our developed algorithm as a "building-block", we engage in two modern robotics applications: learning objects by touch and learning assistance by demonstration.

1.2 THESIS CONTRIBUTIONS

The work in this thesis offers the following contributions:

- A combination of a recently-proposed class of RNNs, the echo-state network, with Bayesian online learning. Our approach bridges a statistical method developed by the machine-learning community and the concept of reservoirs developed within the computational intelligence community. This fusion yields the Online Echo-State Gaussian Process (OESGP) [Soh and Demiris, 2012a], an accurate fixed-budget probabilistic algorithm that is capable of handling noisy temporal sequences and producing predictive uncertainties.

- A recursive kernel with automatic relevance determination (ARD), in effect, extending the OESGP to reservoirs of *infinite* size. This obviates the need to create and maintain an explicit reservoir, resulting in a simpler and computationally more efficient method that is capable of feature selection: the Online Infinite Echo State Gaussian Process (OIESGP) [Soh et al., 2012].

- Efficient, online hyperparameter optimisation (typically neglected in existing online learning methods) through stochastic natural gradient descent (SNGD). We demonstrate that SNGD makes the OIESGP more robust against mis-specification, yielding higher accuracies. When combined with the aforementioned recursive ARD kernel, it permits a better understanding of the learned system via inspection of the resulting hyperparameters.

- An investigation of generative and discriminative tactile classification using OIESGP as a base model [Soh et al., 2012]. In contrast to existing work, our online classifiers are capable of refining existing models of known objects and creating new models "on-the-fly". Experiments using tactile sensors on the iCub anthropomorphic hand demonstrates that our methods attain accuracies comparable to highly-optimised offline classifiers.

- A novel approach to the development of *assistive* policies using the learning-by-demonstration framework. This work was borne out of our interactions with children with special-needs and their occupational therapists at a local rehabilitation centre. We develop an OIESGP-based mixture model that addresses the difficult problems of "when" and "how" to assist by *observing an assistant*. This approach — Learning Assistance by Demonstration (LAD) [Soh and Demiris, 2013] — is applied to the problem of helping wheelchair users navigate their environment and we present experimental results with human subjects using the ARTY smart wheelchair and paired haptic controllers.

Along with the scholarly aspects of this work, we have developed:

- The Online Temporal Learning (OTL) C++ and Matlab Library [Soh, 2012b]; an open-source implementation of OESGP, OIESGP and competing echo-state methods for the online learning of spatio-temporal data.
- The iCub Grasp Dataset [Soh, 2012a] comprising twenty tactile samples for nine different objects (and one baseline "none" class with no object) grasped using the iCub hand: a total of 200 samples. In an effort to encourage the development of tactile learning methods, this dataset has been made freely available on the web.
- The Assistive Robotic Transport for Youngsters (ARTY) [Soh and Demiris, 2012b]; a paediatric robotic wheelchair designed to help young children safely navigate their environment. ARTY has been successfully tested with two young children with special needs at the Tadworth Children's Trust Rehabilitation Center. ARTY was a national finalist for the 2012 James Dyson Award.

1.3 ROADMAP: OVERVIEW AND ORGANISATION

This thesis is organised into four major parts comprised of seven chapters and a final fifth part with supplementary details as appendices (Fig. 1.2):

- **PART I: Preliminaries**
  - **Chapter 2** gives background material related to online learning, focussing on problems in robotics. We review popular online learning methods (such as Locally Weighted Process Regression and online Gaussian Processes) and discuss the recent advances made in this area, as well as current limitations. We then review echo-state networks, a recently proposed form of reservoir computing, for learning temporal signals. These two pillars form the foundation for our methods, presented in the subsequent chapters.

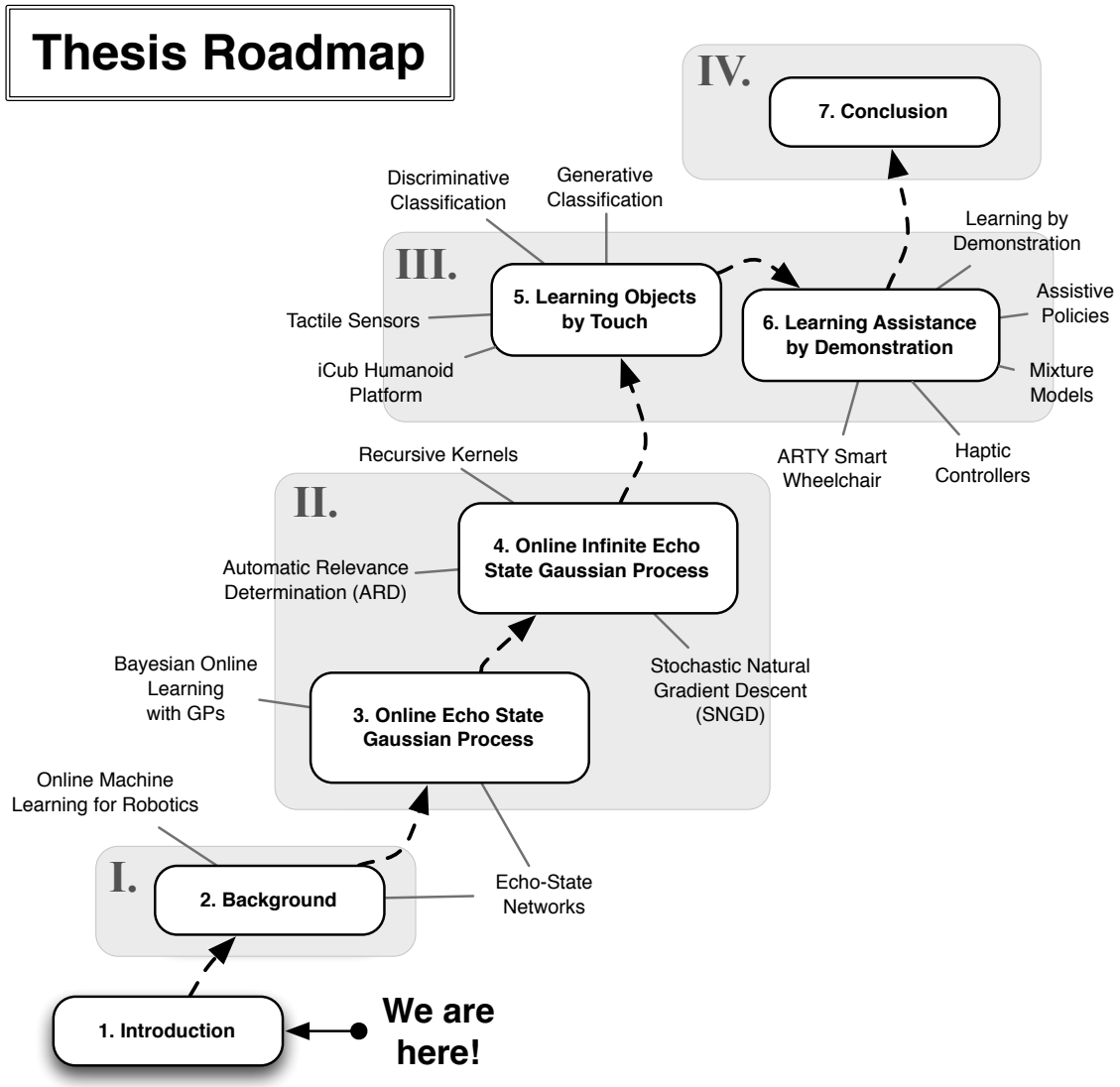


Figure 1.2: This roadmap illustrates the overall organisation of this thesis. The boxes indicate the main chapters, linked with the concepts introduced and discussed.

- **PART II: Online Spatio-Temporal Machine-Learning Methods**
  - **Chapter 3** presents our first algorithmic contribution: the Online Echo-State Gaussian Process (OESGP), a non-parametric Bayesian method for iteratively learning spatio-temporal data. Experiments on a variety of dynamical system benchmarks demonstrate that OESGP achieves low errors relative to current online learning methods. We also describe how the OESGP can be used for real-time recognition of full-body actions via a generative classifier.

  - **Chapter 4** discusses two major extensions to the OESGP. First, we present the concept of recursive kernels and propose a new recursive kernel with automatic relevance determination (ARD). Second, we describe how the hyperparameters of this kernel can be optimised through SNGD. Extensive experiments on one-step prediction, system identification and irrelevant dimension prediction tasks are presented, showing the OIESGP attains accuracies better than state-of-the-art methods, particularly on problems with irrelevant dimension.

- **PART III: Applications in Robotics**
  - **Chapter 5** presents our first robotics application: learning to classify objects by touch on the iCub humanoid platform. Compared to existing methods, our online generative and discriminative classifiers work directly on the tactile sensor data and do not require the generation of an extensive feature vector. Experiments reveal our OIESGP classifiers produce excellent results (achieving perfect accuracy on the final 20% of our dataset).
  - **Chapter 6** describes Learning Assistance by Demonstration; a fresh approach to the complex problem of developing assistive controllers. We formally define the task of learning an assistive policy by observing a demonstrator and formulate an online mixture of experts (ME) model as a solution. Experiments using both a simulated robot and the ARTY smart wheelchair show that the proposed model learns quickly to provide contextual assistance, improving driver performance by 191% after only a *single* demonstration.

- **PART IV: Closing**
  - **Chapter 7** concludes this thesis by summarising our main contributions and findings, followed by a discussion of outstanding issues.

- **PART V: Appendix**
  - **Appendix A** presents mathematical formulae used in our derivations.

- **Appendix B** presents the derivation of the recursive ARD kernel.
- **Appendix C** specifies parameters used in our benchmark experiments.
- **Appendix D** contains supplementary empirical results for benchmark problems (p-value tables and computational times).
- **Appendix E** collates confusion tables for the tactile classification.
- **Appendix F** presents a preliminary LAD experiment with the ARTY wheelchair and joystick control.
- **Appendix G** contains supplementary results (raw survey tables) for the LAD experiments with haptic control.
- **Appendix H** discusses smart mobility for young children and case-studies using ARTY with eight able-bodied children and one child with special-needs.

1.4 MATHEMATICAL NOTATION

We have attempted to use standard notation throughout the thesis. Unless explicitly stated, lowercase letters (e.g., x, y, z) represent scalars or random variables. Vectors are represented by bold letters, e.g., $\mathbf{x}$ and matrices by bolded uppercase letters, $\mathbf{X}$. Vectors are assumed to be column vectors and a superscript $\top$ represents a transposed vector or matrix. As such, $\mathbf{x}^\top$ represents a row vector. Gradients of a function f with respect to a variable x are denoted by $\nabla_x f$. Angle brackets with a single argument are averages, e.g., $\langle f(x) \rangle_q$ represents an average taken with respect to the distribution q. Angle brackets with two arguments $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}}$ represents the inner product between $\mathbf{x}$ and $\mathbf{y}$ in some space $\mathcal{H}$. The notation $\|\mathbf{x}\|$ denotes the norm of $\mathbf{x}$.

Part I

PRELIMINARIES

2

BACKGROUND: ONLINE MACHINE-LEARNING AND ECHO-STATE NETWORKS

Learning — the extraction of patterns from data — plays an important role in robotics; it allows robots to develop capabilities and skills from observed input. For example, learning is employed to develop inverse kinematic controllers when pure theoretical constructs fail to capture all necessary complexities (e.g., in [Vijayakumar et al., 2005]). In robot programming by demonstration, learning algorithms are used to extract task policies from teacher demonstrations [Argall et al., 2009]. The field of machine learning has grown substantially over the years and we refer readers to [Bishop, 2006] and [Murphy, 2012] for comprehensive modern treatments.

This chapter presents an introduction to machine learning for robotics-related problems, with an emphasis on *online* methods that adapt models iteratively. As a guide, we begin with the standard linear model and extend it to non-linear regression, covering popular algorithms such as Locally Weighted Process Regression (LWPR) [Vijayakumar et al., 2005] and approximated Gaussian process models [Csató and Opper, 2002; Nguyen-Tuong et al., 2009]. Following this, we give an overview of echo-state networks [Jaeger, 2001], a class of recurrent neural networks, for learning temporal or time-series data. Finally, we explore connections between the presented ideas, laying the groundwork for our contributions presented in the subsequent chapters.

2.1 ONLINE MACHINE LEARNING FOR ROBOTICS

Machine Learning involves the automation of data analysis to detect patterns and create models, which can be then be used to make predictions and/or decisions [Murphy, 2012]. In this thesis, we will be principally concerned with *supervised learning* tasks where the objective is to learn a mapping from given inputs $\mathbf{x}$ to outputs y (also called "response variables" or "targets") given training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. Depending on the nature of y, we can further categorise our task; if it is real-valued, we are performing *regression* and if it is categorical, *classification*. For example, learning the forward kinematics of a robot arm is a regression problem — we extract a mapping from joint parameters to the cartesian coordinates of the end-effector. In contrast, deciding if an object is a box or a cup is a classification problem.

For now, let us focus on regression. A straightforward approach — called the Standard Linear Model — is to assume that the outputs are generated via a linear combination of the inputs with some additive noise:

$$y = \mathbf{w}^\top \mathbf{x} + \epsilon. \tag{2.1}$$

where the noise is typically assumed Gaussian, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Here, the objective is to learn the weights $\mathbf{w}$ that represent the input-output relationship of the data. This can be achieved by minimising a loss function $\mathcal{L}$, which captures the "cost" associated with a wrong answer. For example, a commonly used loss function is the squared-error loss:

$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2 \tag{2.2}$$

where $\hat{y}$ is our predicted output. The ordinary least squares (OLS) solution to this standard linear model has an analytical solution:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{2.3}$$

where $\mathbf{X}$ and $\mathbf{y}$ are matrices of our inputs and outputs respectively. One drawback associated with this solution is the implicit assumption that the dataset $\mathcal{D}$ is fully present to be worked on. In many robotics tasks, $\mathcal{D}$ may only be partially accessible or may be too large for our computational or storage resources. In addition, $\mathcal{D}$ may only consist of *approximated* data obtained via simulations, which may be poor representatives of real samples [Hagras et al., 2001].

To update models iteratively as more information becomes available, we use *online* or *incremental* methods. Online machine learning methods process data incrementally, i.e., one sample at a time or in mini-batches. After processing, the sample can be discarded, freeing up resources for other purposes. Using our forward kinematics example, online methods enable our robot to learn as it is actively exploring or "motor-babbling" in the space of joint coordinates. Using our loss function (2.2), we can learn the weights $\mathbf{w}$ using an iterative optimisation algorithm such as stochastic gradient descent (SGD), which "follows" the gradient of the cost function iteratively:

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \eta \nabla_{\mathbf{w}} \mathcal{L} \tag{2.4}$$

where η is the *learning rate* that controls the speed of convergence and $\nabla_{\mathbf{w}} \mathcal{L}$ is the gradient of the loss function with respect to $\mathbf{w}$.

Although the Standard Linear Model suffices for simple problems, the assumption of linearity imposed by (2.1) is too naïve for many robotics applications, resulting in poor model fits and high errors. We can consider two ways of extending the Standard Linear Model to the *non-linear* regime: local-model learning and projection via basis functions. These two approaches can be used complementarily.

2.1.1 *Local Model Learning*

The fundamental idea underlying local model learning is simple: although the function we are trying to model may be *globally* non-linear, *local* portions may be approximately linear. With this notion in mind, we can build a collection of

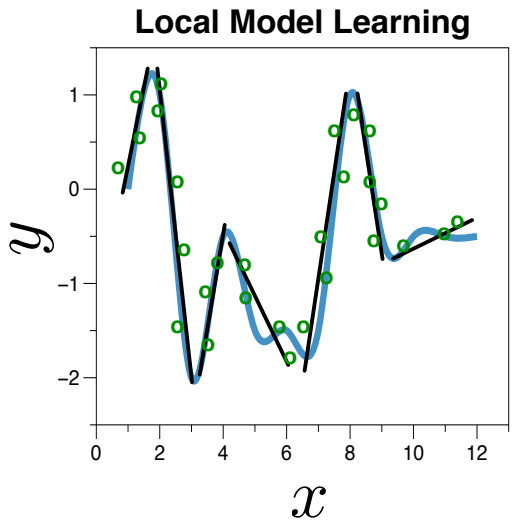


Figure 2.1: Local models are composed of a collection of piecewise local linear models (black lines) to approximate a globally non-linear function (blue line) from noisy observations (green circles).

piecewise linear models, each associated with a small subset of the space (Fig. 2.1).

Among the assortment of proposed local methods, one of the most widely used in robotics is locally-weighted process regression (LWPR) [Vijayakumar et al., 2005]. To make predictions at a test point $\mathbf{x}$, LWPR fuses predictions $\hat{\mathbf{f}} = [\hat{f}_k]_{k=1}^M$ from M locally linear models via a weighted combination:

$$\hat{y} = \mathbf{w}(\mathbf{x})^\top \hat{\mathbf{f}} \tag{2.5}$$

where $\mathbf{w}(\mathbf{x})$ is a vector of normalised weights representing how much each model contributes to the final prediction:

$$\mathbf{w}(\mathbf{x}) = \left[\frac{w_k(\mathbf{x})}{\sum_{i=1}^M w_i(\mathbf{x})} \right]_{k=1}^M \tag{2.6}$$

These weights play a significant role in LWPR and indicate the "region of validity" or "receptive field" of each model. Each weighting function is parameterised

by a distance metric $\mathbf{D}_k$ and the model center $\mathbf{c}_k$ using a Gaussian function (kernel):

$$w_k(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_k)^\top \mathbf{D}_k(\mathbf{x}-\mathbf{c}_k)\right) \tag{2.7}$$

The individual predictions $\hat{f}_k$ are computed using (locally-weighted) Partial least squares (PLS) regression models. In brief, PLS is a latent (hidden) variable regression method that performs regression along one dimensional orthogonal projections. In other words, it *projects* or *maps* the predicted and observed variables onto new "inputs" $\mathbf{z}$, which live in a lower-dimensional space. PLS generalises Principal Components Analysis (PCA), which includes only information from the inputs $\mathbf{x}$, to include the relevancy of y. Each prediction is then assumed to be a linear combination of these inputs:

$$\hat{f}_k = \boldsymbol{\beta}_k^\top \mathbf{z} \tag{2.8}$$

Learning with LWPR involves iterative estimation of the number of local models M, the distance metrics $\mathbf{D}_k$ and the local regression parameters $\boldsymbol{\beta}_k$. During training, updates to each local model is weighted by w_k. If all the generated weights are low, $w_k < w_{gen}$ for $k = 1, \ldots, M$, a new model is created with the input as its center. Learning the distance metric is performed via stochastic gradient descent on a penalised leave-one-out cross-validation loss function:

$$\mathcal{L} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} \frac{w_i (y_i - \hat{y}_i)^2}{(1 - w_i \mathbf{z}_i^\top \mathbf{P}_z \mathbf{z}_i)^2} + \frac{\gamma}{d} \sum_{i,j=1}^{d} D_{i,j}^2 \tag{2.9}$$

where N is the number of training points, d is the dimension on the input and $\mathbf{P}_z$ is the (diagonal) inverse covariance matrix of the projected inputs. In addition, projection variables related to the PLS regressors are locally updated at each step (see [Vijayakumar et al., 2005] for details).

In practice, LWPR's appeal lies in its fast training and prediction times, allowing for real-time inference in high-dimensional spaces, and it has been applied to a variety of robotics problems, e.g., modelling the inverse dynamics of a 30

degree-of-freedom (DoF) SARCOS humanoid robot [Vijayakumar et al., 2005] and for teaching a Sony Aibo robot dog elements of robot soccer [Grollman and Jenkins, 2008].

2.1.2 *Basis Functions for Projections to Feature Spaces*

The second method of extending (2.1) is through the use of basis functions $\phi(\mathbf{x})$ to project the inputs into a (higher-dimensional) feature space where linear regression can be performed:

$$y = \mathbf{w}^\top \phi(\mathbf{x}) + \epsilon. \tag{2.10}$$

Depending on the application at hand, we are free to choose from a variety of basis functions. For example, the widely-used polynomial regression is simply a projection of a scalar x onto the space of powers $\phi(x) = [1, x, x^2, x^3, \dots]$ [Rasmussen and Williams, 2006]. Another popular choice is the radial basis function:

$$\phi_c(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{m}_c||^2}{\sigma_c^2}\right) \tag{2.11}$$

where $\mathbf{m}_c$ and σ_c are pre-set parameters for each basis function c. Though not described as such, we can see that LWPR uses (adaptive) basis functions as PLS projects its inputs onto a lower-dimensional space of latent variables.

The attractiveness of using basis functions in this manner is that it allows us to search for latent function $f(\mathbf{x})$ in a larger space of functions, yet the mechanics of linear regression remain intact and we can follow the same techniques given earlier in this section. The main difficulty lies in choosing appropriate basis functions; a problem that is elegantly solved by the Bayesian approach and Gaussian Processes.

2.1.3 *A Bayesian Treatment*

In Bayesian statistics, we assign *prior distributions* (what we initially believe) to each parameter of the model. Using an appropriate *likelihood function*, we update these priors via Bayes rule to yield a *posterior distribution*. In short, the likelihood function conveys the information provided by the observed sample and the posterior distribution finally summarises everything we know about the parameters after evaluating the evidence.

Within the context of linear regression, let us place a Gaussian prior over the weights $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$; a zero mean normal with positive-definite covariance matrix Σ_p. From the assumptions encoded in (2.10) and the noise model $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, the likelihood is given by:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \mathbf{w}^\top \phi(\mathbf{x}_i))^2}{2\sigma_n^2}\right). \tag{2.12}$$

Applying Bayes Theorem,

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})}{p(\mathcal{D})} \tag{2.13}$$

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}, \tag{2.14}$$

we arrive at a posterior distribution:

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \sim \mathcal{N}\left(\sigma_n^{-2}\mathbf{A}^{-1}\Phi\mathbf{y}, \mathbf{A}^{-1}\right) \tag{2.15}$$

where Φ is matrix of the basis functions evaluated at our N training inputs, and $\mathbf{A} = \sigma_n^{-2}\Phi^\top\Phi + \Sigma_p^{-1}$, i.e., a Gaussian distribution with mean $\sigma_n^{-2}\mathbf{A}^{-1}\Phi\mathbf{y}$ and covariance $\mathbf{A}^{-1}$ [Rasmussen and Williams, 2006; Murphy, 2012]. Using this posterior, we can make predictions at an unknown test point $\mathbf{x}_*$:

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) \sim \mathcal{N}(\mu_*, \sigma_*^2) \tag{2.16}$$

where

$$\mu_* = \sigma_n^{-2}\phi(\mathbf{x}_*)^\top \mathbf{A}^{-1}\Phi\mathbf{y} \tag{2.17}$$

$$\sigma_*^2 = \phi(\mathbf{x}_*)^\top \mathbf{A}^{-1}\phi(\mathbf{x}_*) \tag{2.18}$$

As we can see from (2.16), the posterior predictive distribution is also Gaussian with mean and variance given by (2.17) and (2.18) respectively. We can recognise that (2.17) is a generalisation of the OLS solution (2.3) and we also obtain the uncertainty or "error bars" around the predicted output. This form, however, is rather unwieldy since it involves the inversion of a potentially large matrix $\mathbf{A}$ which is of dimension $D_\phi \times D_\phi$ where D_ϕ is the dimension of the feature space. To help us simplify, let us define the *kernel* function:

$$k(\mathbf{x}, \mathbf{x}') = (\Sigma_p^{\frac{1}{2}}\phi(\mathbf{x}))^\top(\Sigma_p^{\frac{1}{2}}\phi(\mathbf{x}')) \tag{2.19}$$

$$= \psi(\mathbf{x}) \cdot \psi(\mathbf{x}') \tag{2.20}$$

and after some algebraic manipulation (See Chapter 2 of [Rasmussen and Williams, 2006]), we arrive at the alternative but equivalent equations:

$$\mu_* = \mathbf{k}_*^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{2.21}$$

$$\sigma_*^2 = k(\mathbf{x}, \mathbf{x}_*) - \mathbf{k}_*^\top(K + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_* \tag{2.22}$$

where $\mathbf{k}_* = [k(\mathbf{x}, \mathbf{x}_*)]_{i=1}^N$, a vector of kernel evaluations at our input and each of the N training points, and $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^N$ is the Gram matrix of kernel evaluations between each pair of training inputs.

This form has several features, the most notable being that the basis functions $\phi(\mathbf{X})$ have disappeared and been replaced by the kernel function $k(\mathbf{x}, \mathbf{x}')$; this is informally known as the "kernel trick" [Aizerman et al., 1964; Boser et al., 1992]. Since the basis functions are never explicitly computed, we are free to use basis functions that project inputs into very large, even infinite, feature spaces, provided a corresponding kernel function exists. The choice of the kernel implicitly defines the basis function. But how do we choose a proper kernel? Through the

Gaussian Process framework, the kernel function has a specific interpretation that helps guide our choice.

2.1.4 *Gaussian Processes*

The approach by which we arrived at the formulations above is known as the "weight-space view". An alternative approach — the "function-space view" — is to directly place a prior over the space of functions (instead of the weights). In particular, we use a Gaussian Process (GP) prior: a set of random variables where *any* finite subset is jointly Gaussian. A GP is fully specified by its mean function,

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.23}$$

and its kernel or *covariance* function,

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))].$$

It is often assumed that $m(\mathbf{x}) = 0$ and we write the GP as:

$$f(\mathbf{x}) \sim \mathcal{N}(0, k(\mathbf{x}, \mathbf{x}')). \tag{2.24}$$

Here, we see that the kernel specifies the covariance between random variables indexed by the inputs, i.e., how much two random variables should change together. In other words, it computes the "similarity" between inputs. For example, the popular squared exponential (SE) kernel — also called the Radial Basis Function (RBF) or Exponentiated Quadratic kernel — has the form:

$$k^{SE}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right) \tag{2.25}$$

where l is the characteristic length scale (a *hyperparameter* of the model). As can be seen from Fig 2.2, k^{SE} is symmetric and smoothly decaying. It is also *isotropic*, i.e., invariant to rigid motions (translations and rotations of the entire input

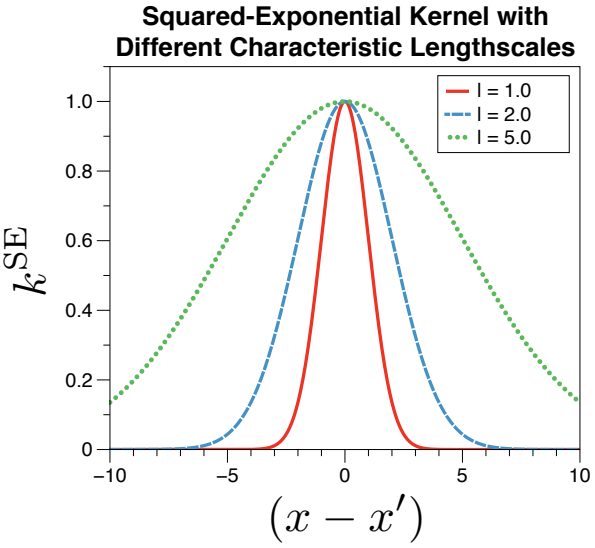


Figure 2.2: The squared-exponential kernel with three different characteristic lengthscales, $l = 1, 2$ and 5. As the figure shows, k^{SE} is symmetric and smoothly decaying. The rate of decay becomes more gradual with increasing l.

space). If we expect our unknown function to have these properties, then k^{SE} makes for a good choice for our prior covariance function.

To perform inference with GPs, we use the introduced Bayesian machinery: first, we place a joint GP prior over our latent function values and a single test point, $\mathbf{x}_*$:

$$\mathbf{f}, f_* \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{k}_* \\ \mathbf{k}_*^\top & k(\mathbf{x}, \mathbf{x}_*) \end{bmatrix}\right) \tag{2.26}$$

where $\mathbf{f} = [f(x_i)]_{i=1}^N$ is a vector of latent function values at the training inputs and f_* is the function value at the test output. Then, we specify our likelihood function:

$$p(\mathbf{y}|\mathbf{f}, \mathbf{X}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathbf{I}) \tag{2.27}$$

which follows from our assumption of Gaussian noise and captures how likely the function values are given the observed outputs $\mathbf{y}$. Finally, computing the posterior and marginalising (integrating) out the latent function values $\mathbf{f}$ yields:

$$p(f_*|\mathbf{y}, \mathbf{X}, \mathbf{x}_*) = \mathcal{N}(\mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \ , \ k(\mathbf{x}, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*). \tag{2.28}$$

which is exactly the same as what we derived before, i.e., Equations (2.16), (2.21) and (2.22). When GP priors are used in this fashion, we are said to be performing Gaussian Process Regression (GPR). Note that changing the likelihood allows us to address other learning problems such as classification, without changing the overall scheme.

It is worth noting that GPs are closely related to another popular kernel learning method: the Support Vector Machine (SVM) [Cortes and Vapnik, 1995; Vapnik, 1995]. Indeed, both are special cases of spline smoothing models [Seeger, 1999], but were developed under different philosophies; the GP has a long history in probability theory and the SVM was formed under the framework of statistical learning theory. In brief, the SVM combines the aforementioned kernel trick and a loss function to promote sparsity. This results in a solution based on a subset of the training data, called the *support vectors*. Although the SVM was initially developed for classification, it has been extended to the regression case by using the epsilon insensitive loss function [Vapnik et al., 1997]. A review of the relationship between GPs and SVMs (and other machine learning methods) can be found in Chapter 6 of [Rasmussen and Williams, 2006] and Chapter 15 of [Murphy, 2012].

In contrast to SVMs, GPs provide predictive distributions instead of point predictions. This feature, along with a solid theoretical foundation and good empirical results, has spurred a great deal of interest in GPs over the past decade. In the robotics domain, GPs have been used in a variety of research work (e.g, for learning inverse dynamics [Chai et al., 2008] and learning-by-demonstration [Schneider and Ertel, 2010]).

2.1.5 *Online Learning with GPs*

In its original formulation, the GP is a poor choice for *online* learning. The main problem lies in its unfavourable computational and storage costs; the kernel matrix $\mathbf{K}$ grows quadratically with the number of training points and prediction requires the inversion of $\mathbf{K}$, which is $O(n^3)$. In short, GPs are expensive.

There are two main ways of adapting GPs for sequential learning. Although in a different context, the first approach is one we have already seen: local modelling. The second approach is through a combination of Bayesian online updates [Opper, 1998; Csató, 2002] and sparsification [Quiñonero Candela and Rasmussen, 2005]. Both techniques are approximations that make the GP more amenable to iterative updates and predictions.

2.1.5.1 *Local Gaussian Process Approximations*

Similar to LWPR, the key idea behind local GPs is to cluster the training samples into smaller partitions[1]. Indeed, the Local Gaussian Process (LGP) [Nguyen-Tuong et al., 2009] bears resemblance to LWPR; if the Gaussian kernel is used, then the same weighting function (2.7) is applied to combine predictions via (2.5). Similarly, new models are spawned whenever all weights fail to reach a predefined threshold w_{gen}.

As we might expect, the major difference is that GPs are used as the local models instead of PLS regression. Each local model maintains a model center $\mathbf{c}_k = N_k^{-1} \sum_{i=1}^{N_k} \mathbf{x}_i$, the average of the training inputs assigned to this model, the kernel matrix $\mathbf{K}_k = [k(\mathbf{x}_i, \mathbf{x}_j]_{i,j}^{N_k}$ and its inverse[2]. With each incorporated training point, $\mathbf{K}_k^{-1}$ can be updated iteratively using the Matrix Inversion Lemma (See Appendix A.1). This procedure is $O(N_k^2)$ per update.

The LGP was recently extended in [Schneider and Ertel, 2010] to incorporate ideas from the Bayesian Committee Machine (BCM) [Tresp, 2000] to provide

1 Note that the primary motivation here is to reduce computational/storage costs and there is no inherent assumption of local linearity.

2 Alternatively, the Cholesky decomposition of both $\mathbf{K}$ and $\mathbf{K}^{-1}$ can be used.

theoretically-sound predictive distributions. Given M local models, the BCM predictive distribution is:

$$\mathbb{E}[f_*] = \mathbb{V}[f_*] \sum_{k=1}^{M} \mathbb{V}_k[f_*]^{-1} \mathbb{E}_k[f_*] \tag{2.29}$$

$$\mathbb{V}[f_*] = \left[\frac{-(M-1)}{\Sigma_*^{-1}} + \sum_{k=1}^{M} \mathbb{V}_k[f_*]^{-1} \right]^{-1} \tag{2.30}$$

where $\mathbb{E}_k[f_*]$ and $\mathbb{V}_k[f_*]$ are the predictive means and variances of the individual models. In words, model predictions are weighted by their associated uncertainties.

In practice, the use of non-linear local models allows LGP to produce more accurate predictions than LWPR but at a higher computational cost (e.g., for real-time control of a Barrett WAM robot arm [Nguyen-Tuong et al., 2009]). The computational complexity of the LGP is dominated by the kernel matrix updates which are $O(N_k^2)$. To keep N_k from becoming too large, the oldest points or those contributing the least information can be dropped.

One core difficulty with LGP is related to the parameter w_{gen}, which controls partitioning of the space. Selecting a proper initial value for w_{gen} is crucial for the algorithm's performance. If w_{gen} is too small, too many small, potentially inaccurate models will be created. On the other hand, if w_{gen} is too large, only a small number of large models will be generated. Let us now consider the second approach for reducing GP complexity which does not suffer from this particular complication.

2.1.5.2 *Sparse Online Gaussian Process*

The Sparse Online Gaussian Process (SOGP) proposed by by Csató and Opper [Csató, 2002; Csató and Opper, 2002] is an application of Bayesian Online Learning to GPs. Recall that in section 2.1.4, we updated the GP prior (2.26) with the likelihood for a single test point (2.28) to yield a posterior. However, instead of marginalising out the latent function values (when making a prediction), we keep the (parameterised) posterior as the prior for the next successive update.

In the case of regression with Gaussian noise, the posterior is also a GP and the updates can be repeatedly applied. In the general case however, the posterior is not update-friendly and we have to resort to "projecting" the posterior onto the closest GP. Here, "closest" is measured via the Kullback-Leibler divergence, $\mathrm{KL}(\hat{p}_t \| q)$. In [Csató and Opper, 2002], it was shown that minimising the KL divergence is equivalent to matching the first two moments of the true posterior $\hat{p}$ and GP approximation q. The updates involve the posterior coefficients, a vector $\boldsymbol{\alpha}$ and a matrix $\mathbf{C}$, which grow with each processed datum.

To control the model's growth, the number of the data-points retained, termed the *basis vectors* (BVs), denoted $\mathbf{b}_i \in \mathcal{B}$ are limited. Each incoming point is scored for "novelty" using a scoring function $\gamma(\mathbf{x}_{t+1})$. If the score is below some threshold, ϵ_γ, then an *approximate* update is performed, which incorporates observations but does not increase the number of BVs. To limit the maximum size or *capacity* of the BV set, it may be necessary to delete a BV. Again, we score each $\mathbf{b}_i \in \mathcal{B}$ and remove the lowest scoring BV using a reduced update; this "downdate" also minimises the KL divergence between the GPs. The computational costs associated with these updates are on the order of $O(s_{\mathcal{B}}^2)$ where $s_{\mathcal{B}}$ is the maximum pre-defined size of $\mathcal{B}$. In effect, $s_{\mathcal{B}}$ determines the trade-off between accuracy and computational cost; larger basis vector sets allow the SOGP to approach the accuracy of the full GP with higher cost per iteration.

2.2 ECHO STATE NETWORKS

In this section, we consider the special case of regression on time-series and temporally correlated data, i.e., where the output depends not only on the current observed state of the system, but also past observations. Time-series regression is particularly relevant for robotics where sensory data is often temporal in nature (e.g., laser scans, sonar readings and tactile feedback). Furthermore, when the states are partially observable (e.g., in robot localisation or object tracking with occlusions), optimal predictions and decisions are achieved through consideration of a *history* of observations and actions.

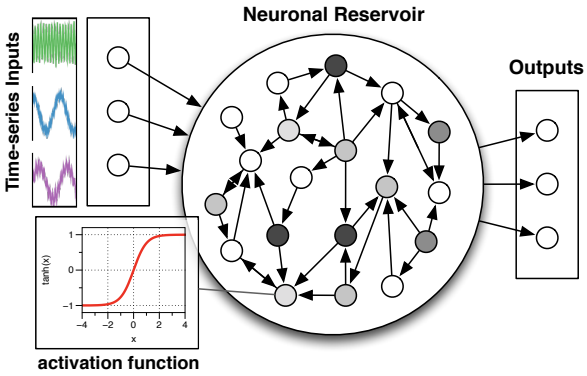


Figure 2.3: The Echo-State Network (ESN). In the typical setup, the inputs are fully connected to a randomly-generated neuronal reservoir (obeying the echo-state property) with a hyperbolic tangent activation function. The outputs are also fully connected to the reservoir with weights learned via linear regression.

Within the computational intelligence community, recurrent neural networks (RNNs) represent the standard approach for temporal learning[3]. Compared to their cousins, the multi-layer perceptions, RNNs contain feedback connections that allow them to represent temporal signals. RNNs are *universal approximators* and theoretically, can represent any open dynamical system [Schäfer and Zimmermann, 2006]. Multiple RNN variants have emerged over the years; from biologically-relevant continuous time RNNs (CTRNNs) [Funahashi and Nakamura, 1993] to long-short-term-memory (LSTM) networks [Hochreiter and Schmidhuber, 1997], which are well-suited for time-series with long time lags between events.

Unfortunately, early RNNs were plagued by training difficulties which presented scaling issues and limited their wide-spread use [Kianifard and Swallow, 1996] — RNNs were typically trained by adapting *all weights* through gradient descent methods such as backpropagation through time (BPTT) [Rumelhart et al., 1986; J. and Werbos, 1988] and real-time recurrent learning (RTRL) [Williams and Zipser, 1989]. These methods were slow to converge (due to vanishing gradients), difficult to implement and/or subject to bifurcations.

In recent years, advances in *reservoir computing* [Jaeger, 2001; Maass et al., 2002; Lukosevicius and Jaeger, 2009] has rekindled research into RNNs. One

3 Readers wanting a more comprehensive discussion of RNNs are referred to excellent reviews [Jaeger, 2005; Grossberg, 2013].

reservoir method, the Echo State Network (ESN) proposed by Jaeger [Jaeger, 2001] (Fig. 2.3), is a novel approach to RNN training and architecture that addresses all three aforementioned issues. The basic notion is to drive a randomly-generated fixed RNN (called the *reservoir*) using the input signal and then derive the output via some combination of the reservoir units (e.g., using standard linear regression). In other words, instead of adapting all network weights, *only the output weights are trained*. More precisely, the state of the reservoir is updated during training:

$$\mathbf{s}_{t+1} = (1-\gamma)h(\mathbf{W}\mathbf{s}_t + \mathbf{W}_i\mathbf{x}_{t+1} + \mathbf{W}_b\mathbf{y}_t) + \gamma\mathbf{s}_t \tag{2.31}$$

where $\mathbf{s}_t$ is the state of the reservoir units at time t, $\mathbf{x}_t$ is the input, $h(\cdot)$ is the activation function, $\mathbf{y}_t$ is the desired output, $\mathbf{W}$ is reservoir weight matrix, $\mathbf{W}_i$ is the input weight matrix, $\mathbf{W}_b$ is the output feedback weight matrix, and γ is the leak (or retainment) rate. After training, the update equation becomes:

$$\mathbf{s}_{t+1} = (1-\gamma)h(\mathbf{W}\mathbf{s}_t + \mathbf{W}_i\mathbf{x}_{t+1} + \mathbf{W}_b\hat{\mathbf{y}}_t) + \gamma\mathbf{s}_t \tag{2.32}$$

and the predicted outputs $\hat{\mathbf{y}}_{t+1}$ are obtained using:

$$\hat{\mathbf{y}}_{t+1} = \mathbf{W}_o\boldsymbol{\psi}_{t+1} \tag{2.33}$$

where $\mathbf{W}_o$ is the linear output weight matrix and

$$\boldsymbol{\psi}_{t+1} \triangleq [\mathbf{s}_{t+1}; \mathbf{x}_{t+1}] \tag{2.34}$$

is the augmented reservoir state and input vector[4]. Surprisingly, this simple procedure yields excellent results on a range of problems from chaotic time-series prediction to system identification [Jaeger, 2003].

As one might expect, the structure of reservoirs has a substantial impact on performance and this has resulted in an abundance of fresh research into opti-

4 This augmentation is not always necessary and it is sometimes sufficient to regress solely on the reservoir state.

mising reservoir topology (e.g, [Deng and Zhang, 2007; Rodan and Tino, 2011]). Arguably, the most important reservoir property is the *echo state property* [Jaeger, 2001], which asserts that initial conditions will be asymptotically "washed out". In other words, the impact of the current state and input will gradually diminish over time. If this property is not present, signals may be amplified, leading to chaotic behaviour. For hyperbolic tangent activation units, the echo-state property is empirically observed to hold if the *spectral radius* (the largest eigenvalue of $\mathbf{W}$) is less than one, $\rho < 1$. In practice, this value is manually tuned to match the given task and memory structure; the closer ρ is to 1, the longer temporal correlations between reservoir states and hence, the memory of the system.

2.2.1 *Online Training for ESNs*

In the case of offline training, the augmented reservoir states, $\boldsymbol{\psi}$, are gathered and regressed against the desired outputs (e.g., using standard linear regression introduced in Section 2.1). For online training, this regression can be performed as the reservoir evolves over time via stochastic gradient descent as demonstrated in [Kountouriotis et al., 2005]; the output weights are updated iteratively:

$$\mathbf{W}_{o,t+1} = \mathbf{W}_{o,t} + \eta(y_t - \hat{y}_t)\mathbf{x}_t \tag{2.35}$$

where η is the learning rate. The authors illustrated the SGD-ESN was sufficient to perform multi-step tracking of a three-dimensional Lorenz system. That said, convergence performance of the SGD approach is not only heavily dependent on η but is also negatively impacted by the eigenvalue spread of the reservoir cross-correlation matrix [Lukosevicius and Jaeger, 2009].

To enable better convergence, Jaeger proposed using the recursive least squares (RLS) algorithm [Jaeger, 2003]. Briefly, RLS is an adaptive filter which finds the

output weights that minimise the least squares error function. In its basic form, RLS (applied to the ESN) consists of the following iterative updates:

$$\begin{aligned}
\vartheta_{t+1} &= y_{t+1} - \boldsymbol{\psi}_{t+1}^{\top}\mathbf{w}_{o,t} \\
\mathbf{g}_{t+1} &= \mathbf{P}_t\boldsymbol{\psi}_{t+1}(\lambda + \boldsymbol{\psi}_{t+1}^{\top}\mathbf{P}_t\boldsymbol{\psi}_{t+1})^{-1} \\
\mathbf{P}_{t+1} &= \lambda^{-1}\mathbf{P}_t - \mathbf{g}_{t+1}\boldsymbol{\psi}_{t+1}^{\top}\lambda^{-1}\mathbf{P}_t \\
\mathbf{w}_{o,t+1} &= \mathbf{w}_{o,t} + \vartheta_t\mathbf{g}_{t+1}
\end{aligned} \tag{2.36}$$

where $\mathbf{w}_{o,t}$ is a row of the output weights matrix at time t and λ is the forgetting factor. At $t = 0$, $\mathbf{w}_{o,0} = 0$ and $\mathbf{P}_0 = \delta^{-1}\mathbf{I}$ where δ is user-defined. From the equations, readers may recognise RLS as a special-case of the popular Kalman filter. Using the RLS-ESN, Jaeger demonstrated online adaptation of the ESN for a system identification task — a tenth-order Non-linear auto-regressive moving average (NARMA) problem. In general, the RLS-ESN exhibits fast convergence but is subject to numerical instability[5] and is more computationally expensive compared to SGD; each RLS update is on the order of $O(N_{\psi}^2)$ where N_{ψ} is length of $\boldsymbol{\psi}$.

In this thesis, we expand upon this body of work and contribute a novel online training method for ESNs through Bayesian online learning, specifically the SOGP. In fact, the SOGP is intimately connected with the Kernel Recursive Least Squares (KRLS) and its more recent variants [Van Vaerenbergh et al., 2012; Engel et al., 2004]. KRLS is a kernelised RLS filter with a sparse dictionary (BV set) and is a popular method within the signal processing community. Indeed, both KRLS and SOGP give *identical* mean predictions if parameterised accordingly [Van Vaerenbergh et al., 2012]. However, the SOGP has an underlying probabilistic foundation and offers predictive distributions conveying uncertainty. Let us now close off this chapter by summarising the ideas introduced and examining connections that will pave the way towards our contributions.

5 As discussed in [Jaeger, 2003], numerical instability can be partially resolved using noise insertion.

2.3 SUMMARY, CONNECTIONS AND MOVING AHEAD

In this chapter, we presented a review of machine-learning methods for regression in online settings, focussing in on three state-of-the-art methods: LWPR, LGP and SOGP. We then gave an overview of echo-state networks, which have emerged as surprisingly accurate and computationally efficient models for temporal prediction tasks.

From our discussion above and taking up ideas in Section 2.1.2, the echo-state network can be viewed as a basis function that operates on time-series instead of single data points. From this perspective, we see that the ESN architecture can be combined with *any* of the aforementioned online regressors to yield a temporal learner that may be more accurate than RLS and more importantly, capable of producing confidence estimates. The question then is which method is likely to work well?

Comparisons between the LWPR, LGP and SOGP reveal that the GPs tend to attain better accuracy but are slower [Grollman and Jenkins, 2008; Nguyen-Tuong et al., 2009; Xiao and Eckert, 2013]. Of the three, SOGP is a *fixed-budget* algorithm: it is possible to limit and ascertain the maximum computation time and storage requirements through $s_{\mathcal{B}}$. In contrast, LWPR and LGP may grow models indefinitely, causing slowdowns. For example, in a learning by demonstration experiment with LWPR, "as more data was generated by the user to further teach the task, the increasing processing time per point became a problem" such that "interaction became impractical" [Grollman and Jenkins, 2008].

Empirically, it remains to be seen whether SOGP or LPR is the more accurate model. Although it was reported in [Nguyen-Tuong et al., 2009] that LGP outperformed the SOGP on three inverse dynamics tasks, a more recent study [Xiao and Eckert, 2013] comparing the two methods (alongside other online algorithms) showed that the SOGP obtained better scores across six large datasets including the SARCOS robot-arm dataset. As is usually the case, algorithm accuracy is likely task-dependent. On the one hand, LGP may better model local details that may not be well captured by a small BV set. On the other hand,

SOGP is a "global" model and is not susceptible to the problems associated with partitioning the space.

As we are firmly within the robotics domain, bounded computational and storage costs are a priority; trackers and controllers are often expected to operate at a specified frequency. Also, intuitively, SOGP is the more elemental model — one may naturally envisage a collection of local SOGP models, but the reverse is not evident[6]. These aspects, along with the positive accuracy scores, made it an appealing choice as a "base model" on which to build our contributions. In Chapter 3, we will undertake a closer examination of the SOGP as we combine it with the ESN to yield a novel online temporal learner.

6 Mixture models are further discussed in Chapter 6.

Part II

ONLINE SPATIO-TEMPORAL MACHINE-LEARNING METHODS

In the next two chapters, we present our online machine-learning methods for learning temporal multi-variate data streams. In Chapter 3, we apply Bayesian online-learning to the echo-state network and develop the Online Echo-State Gaussian Process (OESGP). In Chapter 4, we extend the reservoirs to infinite size and propose a novel recursive kernel with feature relevance determination. When combined with the online GP and stochastic natural gradient descent, this approach yields an adaptive method we term the Online Infinite Echo-State Gaussian Process (OIESGP). Both chapters present extensive numerical results on benchmarks evaluating the proposed algorithms against state-of-the-art methods.

3

ONLINE ECHO-STATE GAUSSIAN PROCESS

This chapter presents the *Online Echo-State Gaussian Process* (OESGP) [Soh and Demiris, 2012a], a non-parametric iterative temporal learner (illustrated in Fig. 3.1). Unlike current online ESNs trained using RLS [Jaeger, 2003] and SGD [Kountouriotis et al., 2005], our Bayesian-based formulation gives predictive distributions (confidence estimates) instead of point-predictions. Uncertainty estimates are relevant in real-time decision making to compute probable outcomes and also in active-learning [Kapoor et al., 2007] where data is scarce or expensive to obtain. Moreover, the use of kernels allows for non-linear mappings between reservoir states and desired outputs, permitting greater flexibility in modelling and adapting to dynamical systems.

One can view the OESGP as an extension of the recently-proposed echo state Gaussian process (ESGP) [Chatzis and Demiris, 2011] to online learning scenar-

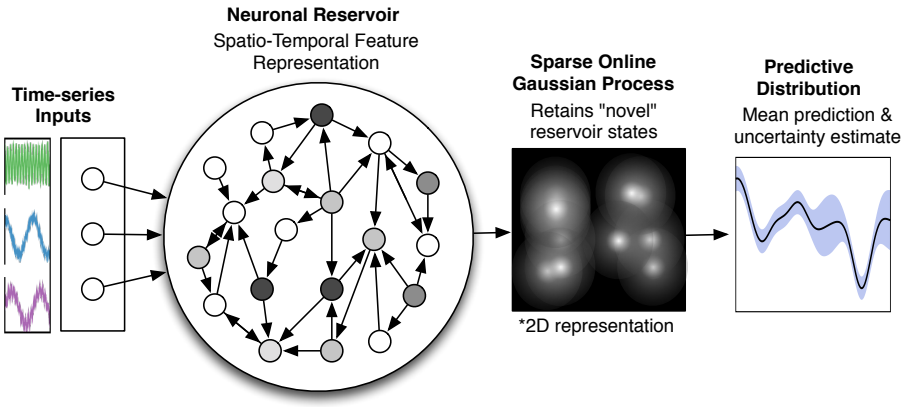


Figure 3.1: The Online Echo State Gaussian Process which learns online from temporal sequences and produces predictive distributions.

ios. The ESGP has high $O(n^3)$ computational complexity, making it too expensive for real-time robotics applications. To reduce computational and storage costs, OESGP stores only "informative" or "novel" neural states; non-novel states are absorbed but do not increase model size. By using online sparse approximations for GPs [Csató and Opper, 2002], we enable fast iterative learning and prediction given large amounts of sequential data. As we will show, OESGP's computational cost can be controlled by limiting the number of stored neural states. Comparing the OESGP to existing online regression algorithms (e.g., LWPR [Vijayakumar et al., 2005] and KRLS [Engel et al., 2004]), our ESN-based method takes into account temporal dependencies via the constructed reservoir.

The remainder of this chapter are organised as follows. The next section presents the ESGP as a starting point for our work. In Section 3.2, we describe the OESGP, in particular the sparsification techniques employed. Experimental results on benchmark datasets are given in Section 3.3. Section 3.4 discusses the trade-off between computational cost and accuracy. We then present in Section 3.5 an application of the OESGP towards online generative classification of full-body actions. Finally, we conclude this chapter in Section 3.6 with a summary and outstanding issues as a prelude to the next chapter on the online infinite echo state GP.

3.1 THE ECHO STATE GAUSSIAN PROCESS

The ESGP proposed by Chatzis and Demiris [Chatzis and Demiris, 2011] is a Bayesian formulation of the standard echo-state network. Recall from (2.33) that each echo state output is produced by multiplying a row from the readout matrix $\mathbf{W}_o$ and ψ (the augmented network-state and input vector). Using the Bayesian formalism discussed in Chapter 2 and placing a Gaussian prior over each row of weights, $\mathbf{w}_j \sim \mathcal{N}(0, \mathbf{I})$, we can derive that the distributions of the ESN outputs yield a GP for each output:

$$[f_{j,t}]_{t=t_1}^{t_T} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_r) \tag{3.1}$$

where $\mathbf{K}_r = [k_r(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)]_{i,j=t_1}^{t_T}$ and $k_r(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j)$ is the chosen *reservoir kernel function*. Based on the above formulation, we observe that the ESGP uses a covariance function which is a kernel function on the ESN state, thus capturing temporal relationships between sequential observations [Chatzis and Demiris, 2011].

Intuitively, one can consider the ESGP as an echo state network where training is performed with GPR instead of linear regression. In fact, the ESGP is a generalisation of ESNs trained via linear and ridge regression; the mean-prediction of the ESGP with a linear kernel is identical to the one produced by a ESN trained using ridge-regression. Under the Bayesian framework, the ESGP provides a predictive distribution instead of a point prediction. In [Chatzis and Demiris, 2011], the authors found the ESGP to be highly effective on a variety of benchmark and real-world tasks, achieving high accuracies with a reasonable increase in training cost.

Turning our attention to online learning, one can consider a direct application of the ESGP by "growing" the kernel matrix as new samples are processed. However, two major issues make this approach infeasible in practice. First, the ESGP incurs a high computational cost; recall that the matrix inversion needed in the predictions are $O(n^3)$ for a $n \times n$ kernel matrix. This is likely too slow for real-time applications. Second, $\mathbf{K}_r$, will grow quadratically and *unbounded*, thereby straining both storage and computational resources.

3.2 THE ONLINE ECHO STATE GAUSSIAN PROCESS

The online learning approach we use is based on the SOGP [Csató, 2002; Csató and Opper, 2002] and we closely follow their treatment. Although other sparse representations of GPs exist (see [Quiñonero Candela and Rasmussen, 2005] for a comprehensive review), they generally require that the entire dataset is available during training, making them inappropriate for online learning.

As a guide, we first discuss how to perform successive updates to the ESGP as new data points arrive. Then, we address the unbounded storage problem by keeping only "novel" reservoir states up to some maximum capacity. The latter

is based on minimising the Kullback-Leibler (KL) divergence between exact updates (the model grows) and sparse updates (the model size remains the same); in short, we minimise the "distance" between the full and approximated GPs.

3.2.1 *Bayesian Online Learning and the Projected Process Approximation*

Taking the first problem, let us denote $\mathbf{x}_{t+1}$ as the input into the reservoir, y_{t+1} as the observed output signal, $\mathbf{s}_{t+1}$ as the updated reservoir state and $\boldsymbol{\psi}_{t+1} \triangleq [\mathbf{s}_{t+1}; \mathbf{x}_{t+1}]$. Our goal is to incrementally update the ESGP given $(\mathbf{x}_{t+1}, y_{t+1})$. Applying Bayesian online learning for regular GPs [Opper, 1998; Csató and Opper, 2002] to our specific case, our approach consists of two basic steps:

1. **Update the ESN state** using (2.31) to derive the new composite state $\boldsymbol{\psi}_{t+1}$.
2. **Update the model posterior** given $(\boldsymbol{\psi}_{t+1}, y_{t+1})$ and **project the posterior** onto the closest GP.

While step 1 is easily achieved, step 2 deserves more discussion. Let us assume we have an ESGP at time t. We incorporate a new datapoint into the ESGP by performing a Bayesian update to yield a posterior:

$$\hat{p}(\mathbf{f}|y_{t+1}) = \frac{P(y_{t+1}|f(\boldsymbol{\psi}_{t+1}))p_t(\mathbf{f})}{\langle P(y_{t+1}|f(\boldsymbol{\psi}_{t+1}))p_t(\mathbf{f})\rangle_t}. \tag{3.2}$$

where we have neglected conditioning on the inputs for notational simplicity. In the case of GP regression, this update is *exact*. In the general case however, the update cannot be applied repeatedly since it yields a posterior process that is typically non-Gaussian with intractable integrals. One way to get around this is to employ an approximation and *project* the process onto the closest GP where "closest" is measured via the Kullback-Leibler divergence, $KL(\hat{p}_t||q)$, and q is our desired approximation. Minimising the KL divergence is equivalent to matching

the first two moments of $\hat{p}$ and q [Csató and Opper, 2002]. The update equations in their "natural parameterisation" forms are given by[1]:

$$m_t(\psi) = \boldsymbol{\alpha}_t^T \mathbf{k}_r(\psi) \tag{3.3}$$

$$k_t(\psi, \psi') = k_r(\psi, \psi') + \mathbf{k}_r(\psi)^T \mathbf{C}_t \mathbf{k}_r(\psi') \tag{3.4}$$

where the vector $\boldsymbol{\alpha}$ and matrix $\mathbf{C}$ are updated using:

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + b_1(\mathbf{C}_t \mathbf{k}_{r,t+1} + \mathbf{e}_{t+1}) \tag{3.5}$$

$$\mathbf{C}_{t+1} = \mathbf{C}_t + b_2(\mathbf{C}_t \mathbf{k}_{r,t+1} + \mathbf{e}_{t+1})(\mathbf{C}_t \mathbf{k}_{r,t+1} + \mathbf{e}_{t+1})^T \tag{3.6}$$

where $\mathbf{k}_{r,t+1} = [k_r(\psi_1, \psi_{t+1}), ..., k_r(\psi_t, \psi_{t+1})]$, $\mathbf{e}_{t+1}$ is the $t+1^{th}$ unit vector and the scalar coefficients b_1 and b_2 are given by:

$$b_1 = \partial_{f_{t+1}} \ln \langle P(y_{t+1}|f(\psi_{t+1}))\rangle_t \tag{3.7}$$

$$b_2 = \partial^2_{f_{t+1}} \ln \langle P(y_{t+1}|f(\psi_{t+1}))\rangle_t \tag{3.8}$$

In the case of regression with Gaussian noise, we have:

$$b_1 = (y_{t+1} - \mu_{t+1})/\sigma^2_{t+1} \tag{3.9}$$

$$b_2 = -1.0/\sigma^2_{t+1} \tag{3.10}$$

where μ_{t+1} and σ^2_{t+1} are the predicted mean and variance at time $t+1$. Note that b_2 does not depend on the outputs y_{t+1} so, only a single matrix $\mathbf{C}$ needs to be updated regardless of the output dimensionality (if the same basis vectors are used for each output). However, using the same $\mathbf{C}$ in a multi-dimensional setting has implications for BV deletion (discussed in the next section). Furthermore, the computational cost savings should be weighed against potential accuracy loss, particularly in situations where the input-output relationship for each dimension is better modelled using a different BV set (e.g., the outputs have different time-scales).

1 These follow naturally from the update equations given in [Csató and Opper, 2002].

3.2.2 *Maintaining Sparsity*

Although the "full update" equations (3.3)-(3.4) allow us to update the ESGP sequentially, $\boldsymbol{\alpha}$ and $\mathbf{C}$ increase with the number of processed samples. The core concept in constraining model growth is to limit the number of the reservoir states retained (called the *basis vectors* (BV), $\mathbf{b} \in \mathcal{B}$) using a scoring function that computes the "novelty" of the state $\boldsymbol{\psi}_{t+1}$. Two basic steps are involved in maintaining the sparsity:

1. **Compute the score** of $\boldsymbol{\psi}_{t+1}$ and if the score is higher than some threshold, perform an update using (3.3)-(3.4).
2. **Maintain the size of** $\mathcal{B}$ by removing the lowest scoring BV if $|\mathcal{B}|$ exceeds some predefined capacity.

The scoring function proposed in [Csató, 2002; Csató and Opper, 2002] is the reconstruction/projection error using the existing BV set:

$$\gamma(\boldsymbol{\psi}_{t+1}) = k_r(\boldsymbol{\psi}_{t+1}, \boldsymbol{\psi}_{t+1}) - \mathbf{k}_{\mathcal{B},t+1}^T \mathbf{K}_{\mathcal{B},t}^{-1} \mathbf{k}_{\mathcal{B},t+1} \tag{3.11}$$

where $\mathbf{k}_{\mathcal{B},t+1} = [k_r(\mathbf{b}_i, \boldsymbol{\psi}_{t+1})]_{\mathbf{b}_i \in \mathcal{B}}$ and $\mathbf{K}_{\mathcal{B},t}^{-1} = [k_r(\mathbf{b}_i, \mathbf{b}_j)]_{\mathbf{b}_i, \mathbf{b}_j \in \mathcal{B}}$. If $\gamma(\boldsymbol{\psi}_{t+1})$ is below some constant threshold, ϵ_γ (10^{-5} in our work), then we do *not* perform the full update. Instead, we perform an *approximate update* using (3.5) and (3.6) with the only change being that we use:

$$\hat{\mathbf{e}}_{t+1} = \mathbf{K}_{\mathcal{B},t}^{-1} \mathbf{k}_{r,t+1} \tag{3.12}$$

instead of the unit vector $\mathbf{e}_{t+1}$. This update does not increase the size $\mathcal{B}$ but does absorb states which are not included. This operation may appear expensive since it involves computing the inverse of $\mathbf{K}_{\mathcal{B}}$. However, this inversion can be performed iteratively using the Sherman-Morrison formula (Eq. A.4), i.e., $\mathbf{K}_{\mathcal{B},t+1}^{-1} = \mathbf{K}_{\mathcal{B},t}^{-1} + \gamma_{t+1}^{-1}(\hat{\mathbf{e}}_{t+1} - \mathbf{e}_{t+1})(\hat{\mathbf{e}}_{t+1} - \mathbf{e}_{t+1})^T$.

Since we also limit the maximum size (*capacity*) of $\mathcal{B}$, it may be necessary to delete a basis vector. Assume that we have just added a new BV. We then score each $\mathbf{b}_i \in \mathcal{B}$ using the scoring function:

$$\epsilon_i = \frac{|\boldsymbol{\alpha}_{t+1}(i)|}{\mathbf{K}^{-1}_{\mathcal{B},t+1}(i,i) + \mathbf{C}_{t+1}(i,i)} \tag{3.13}$$

and remove the lowest scoring BV using a reduced update of our model. Note that this score is truncated loss (measured in KL-distance) between the approximated and updated GPs. For multi-dimensional outputs, we may combine the scores by computing the Euclidean norm or the maximum. In this work, we have opted for the former and preliminary experiments did not reveal substantial differences between the two techniques. Suppose we wish to remove the j^{th} BV. Let us define $\boldsymbol{\alpha}'$ as the vector $\boldsymbol{\alpha}_{t+1}$ with the element $\alpha^* = \boldsymbol{\alpha}_{t+1}(j)$ removed. We define $\mathbf{C}'$ as the matrix $\mathbf{C}_{t+1}$ without the j^{th} row and column and $c^* = \mathbf{C}_{t+1}(j,j)$. The column vector $\mathbf{c}^*$ is the j^{th} row without c^*. Let $\mathbf{Q} = \mathbf{K}^{-1}_{\mathcal{B},t+1}$ and $\mathbf{Q}'$, $\mathbf{q}^*$, q^* be similarly defined as for $\mathbf{C}$. Then, our reduced update equations are given by:

$$\hat{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}' - \alpha^* \frac{\mathbf{q}^*}{q^*} \tag{3.14}$$

$$\hat{\mathbf{C}}_{t+1} = \mathbf{C}' + c^* \frac{\mathbf{q}^* \mathbf{q}^{*\top}}{q^{*2}} - \frac{1}{q^*}\left(\mathbf{q}^* \mathbf{c}^{*\top} + \mathbf{c}^* \mathbf{q}^{*\top}\right) \tag{3.15}$$

$$\hat{\mathbf{Q}}_{t+1} = \mathbf{Q}' - \frac{\mathbf{q}^* \mathbf{q}^{*\top}}{q^*} \tag{3.16}$$

This completes our discussion of the main aspects of the sparse approximation.

3.2.3 *Training Summary and Making Predictions*

To summarise, training the OESGP consists of four basic steps:

1. **Update the ESN state** using (2.31) to derive the new composite state $\boldsymbol{\psi}_{t+1}$.
2. **Compute the score** of $\boldsymbol{\psi}_{t+1}$ using (3.11).

Problem	Reservoir size	Spectral radius	Connectivity	SGD-ESN η	RLS-ESN (δ, λ)	OESGP (l^2, σ)
Mackey-Glass	100	0.99	0.1	0.05	$(10^{-8}, 0.99999)$	(20, 0.005)
Henon	100	0.90	0.1	0.01	$(10^{-9}, 0.999)$	(10, 0.1)
Laser	100	0.90	0.1	0.05	$(10^{-9}, 0.999999)$	(20, 0.002)
Ikeda	200	0.90	0.1	0.001	$(10^{-8}, 0.999999)$	(10, 0.001)
Lorenz	100	0.99	0.2	0.05	$(10^{-1}, 0.999999)$	(10, 0.05)
NP1	50	0.90	0.1	0.01	$(10^{-3}, 0.999)$	(4, 0.0001)
NP2	100	0.90	0.1	0.01	$(10^{-3}, 0.99999)$	(2, 0.1)

Table 3.1: ESN and algorithm parameters for the problems considered in this study.

3. **Perform a full update** using (3.3)-(3.4) if the score is higher than a predefined threshold ϵ_γ. Otherwise, **perform an approximate update** by substituting $\mathbf{e}_{t+1}$ with $\hat{\mathbf{e}}_{t+1}$ (3.12).

4. **Maintain the size of** $\mathcal{B}$ by removing the lowest scoring BV using (3.14)-(3.16) if $|\mathcal{B}|$ exceeds some predefined capacity.

Making predictions with the OESGP is straightforward with the mean of the predictive distribution given by:

$$\mu_* = \mathbf{k}_{\mathcal{B},t}(\boldsymbol{\psi}_*)^\top \boldsymbol{\alpha}_t \tag{3.17}$$

and variance:

$$\sigma_*^2 = k_r(\boldsymbol{\psi}_*, \boldsymbol{\psi}_*) + \mathbf{k}_{\mathcal{B},t}(\boldsymbol{\psi}_*)^\top \mathbf{C}_t \mathbf{k}_{\mathcal{B},t}(\boldsymbol{\psi}_*) \tag{3.18}$$

Conceptually, our method can be seen as an online, iterative version of the ESGP using the sparse approximations developed by Csató and Opper [Csató and Opper, 2002; Csató, 2002]. Compared to ESGP, our variant has a lower computational complexity of $O(s_\mathcal{B}^2 + N_\psi s_\mathcal{B} + N_\psi^2)$ time where $s_\mathcal{B}$ is the maximum BV set size, typically chosen based on available computational resources. The first two terms result from the GP update and the third term is due to the reservoir update.

Problem	Equations	Parameters
Mackey-Glass	$y(t+1) = y(t) + \delta\left(0.2\frac{y(t-\tau\delta)}{1+y(t-\tau)^{10}} - 0.1y(t)\right)$	$\delta = 0.1, \tau = 17$
Henon	$z(t+1) = 1 - az(t)^2 + z(t)$ $y(t+1) = bz(t)$	$a = 1.4, b = 0.3$ $z(0) = 0$
Ikeda	$z(t+1) = a + rz(t)\exp\left(i(\phi - \frac{p}{(1+\lvert z(t)\rvert)^2})\right)$ $y(t+1) = [\Re z(t), \Im z(t)]$	$a = 1, r = 0.9, \phi = 6$ $z(0) = 0$
Lorenz	$\frac{dx}{dt} = -\sigma(x-y), \frac{dy}{dt} = -\rho x - y - xz$ $\frac{dz}{dt} = -xy - \beta z$	$\sigma = 10, \rho = 28, \beta = 8/3$ $x_0 = 5, y_0 = 5,$ $z_0 = 20, dt = 0.1$
NP1	$d(t) = (0.8 - 0.5\exp(-d(t-1)^2)d(t-1) -$ $(0.3 + 0.9\exp(-d(t-1)^2)d(t-2) + 0.1\sin(d(t-1)\pi)$ $y(t) = d(t) + \epsilon_1$	$\epsilon_1 \sim \mathcal{N}(0, 0.1)$ $d(0) = d(1) = 0.1$
NP2	$v(t) = 1.1\exp(-\lvert v(t-1)\rvert) + u(t)$ $d(t) = v(t)^2, y(t) = d(t) + \epsilon_2$	$u(t) \sim \mathcal{N}(0, 0.25),$ $\epsilon_2 \sim \mathcal{N}(0, 1.0)$ $v(0) = 0.5$

Table 3.2: Governing equations and parameterisations for the benchmark problems used in our experiments.

3.3 EMPIRICAL RESULTS

In this section, we report on experiments designed to investigate the performance of OESGP, relative to the other online ESNs (SGD-ESN and RLS-ESN) and state-of-the-art online regression algorithms (e.g., LWPR).

3.3.1 *Implementation and Setup*

Our OESGP was implemented in MATLAB and in C++ using the Eigen library [Guennebaud et al., 2010] and is available online [Soh, 2012b]. The LWPR results were obtained using the lwpr library with MATLAB bindings [Vijayakumar et al., 2012]. Runs were conducted on a 2.8Ghz Intel Core i7 processor, with each run repeated 30 times. All online ESNs were initialised so that the same reservoir was used for a given sequence.

For RLS and SGD parameters, we used a grid search on a subset of the data (5000 samples) and selected the parameters with the lowest RMSE (on the last 20% of the sequence averaged over 10 independent runs). LWPR parameters were set to recommended defaults with adaptation enabled [Klanke and Vijayakumar, 2008]. For OESGP, we used the squared-exponential kernel and var-

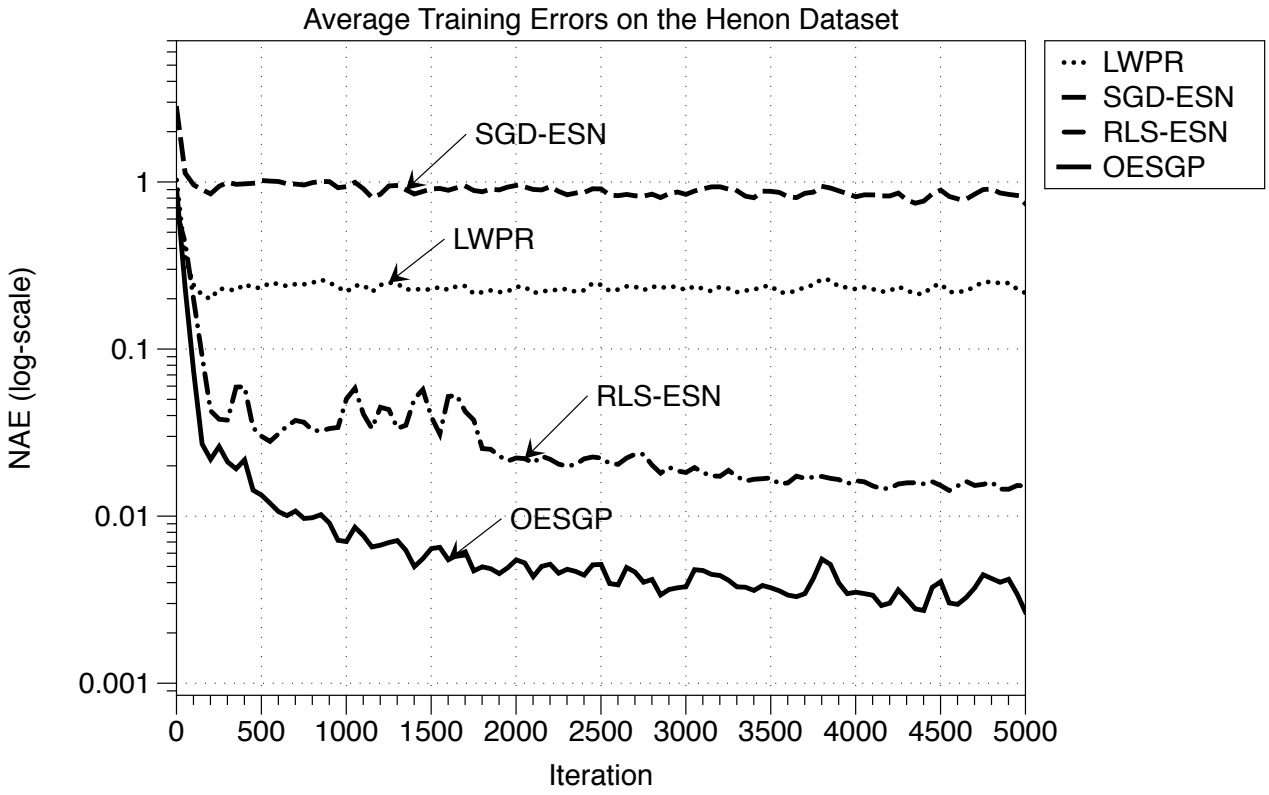


Figure 3.2: NAE convergence profiles during training on the Henon dataset (averaged across 30 runs and smoothed).

ied capacities, $s_{\mathcal{B}} = 50, 100$ and 500. Complete parameter listings for the underlying ESNs and algorithms for each problem are given in Table 3.1.

3.3.2 *One-Step Prediction on Benchmark Problems*

We begin by presenting empirical results for the classic one-step prediction task on well-known benchmark problems, i.e., the Mackey-Glass, Henon, Lorenz and Ikeda dynamical systems, along with the Sante-Fe Institute (SFI) laser competition dataset [Weigend, 1993]. Note that the Mackey-Glass, Ikeda, and Lorenz datasets were treated with the tangent hyperbolic transform, as in [Chatzis and Demiris, 2011]. In addition to these "clean" sequences, we include two systems with noisy observations, NP1 and NP2 [Richard et al., 2009]. For NP1, the inputs consisted of the pair (y_{t-1}, y_t) (consistent with [Richard et al., 2009] though not strictly necessary for the ESNs). We used standard equations for the dynamical

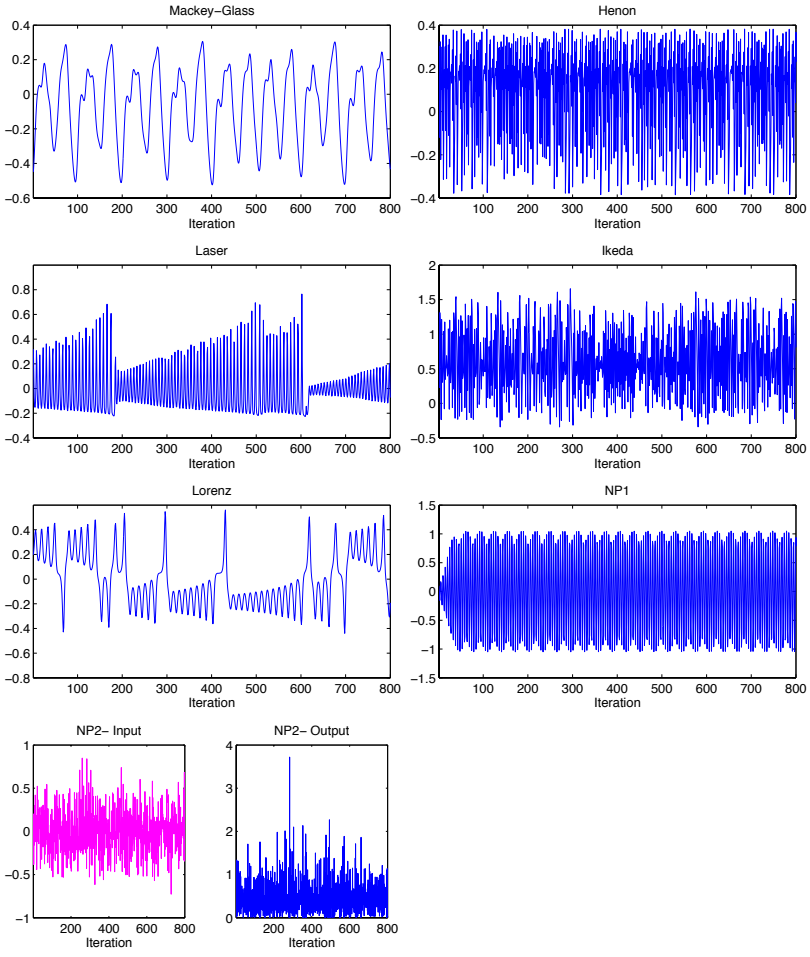


Figure 3.3: The first 800-steps (noise-free) for each benchmark dynamical system used in the experiments. The Mackey-Glass, Henon, Laser, Ikeda, Lorenz and NP1 are one-step prediction problems. NP2 is a system identification problem where the inputs and outputs are not from the same sequence.

systems (shown in Tbl. 3.2) and our scripts[2] for generating the sequences are available for download [Soh, 2012b]. Sample time-series (the first 800 steps) for each benchmark problem are shown in Figure 3.3.

Each problem consisted of sequences of 10^5 samples where training was performed for only half the sequence (predictions were still carried out during training), after which the algorithms were expected to produce predictions given inputs but without any future training. Given the true noise-free output, d_t,

2 Our scripts made use of code by Eric Wen [Wen, 2011].

methods were compared using two error functions computed on the final 50% of the sequence, i.e., the mean normalised absolute error,

$$\text{MNAE} = (sT_e)^{-1} \sum_{t=t_s}^{T_s} \sqrt{(d_t - \hat{y}_t)^2} \tag{3.19}$$

and the root-mean-square prediction error,

$$\text{RMSE} = T_e^{-1} \sum_{t=t_s}^{T_s} \sqrt{(d_t - \hat{y}_t)^2} \tag{3.20}$$

where $\hat{y}_t$ is the predicted output at time t, $t_s = 5001$, $T_s = 10000$, s^2 is the empirical variance of the desired target signal and T_e is the length of the testing sequence. Comparisons were made using the average scores. For multi-dimensional outputs, we averaged the scores over all outputs. Statistical significance between the error distributions generated by the online ESNs were tested using the Wilcoxon signed-rank test.

A summary of our results are shown in Table 3.3. In general, we observed that OESGP (with 100 or 500 BVs) performed remarkably well, achieving the lowest MNAE scores for *all* the benchmark problems (results statistically significant at $p < 0.0001$). Compared to RLS-ESN, OESGP_{500} attains MNAE and RMSE scores that are up to 90% lower. In addition, the training error profiles showed that the convergence rate for OESGP was fast relative to the other algorithms; for example, on the Henon problem (see Fig. 3.2), a log-log fit gave a steeper slope (rate) of -0.53 for OESGP and -0.39 for RLS-ESN. Among the online ESNs, SGD-ESN obtained the worst error scores, evidently hampered by its inability to converge to proper weights. LWPR did not fare much better for many of the problems since it did not take into account temporal relationships.

Figure 3.4 summarises the mean iteration times where we observed a consistent pattern where SGD-ESN was the fastest method followed by LWPR, then RLS-ESN and finally OESGP. This was not surprising because training and predictions (at the pre-set reservoir size of 100 for most problems) are more expensive for OESGP compared to $O(N_\psi^2)$ for RLS-ESN and $O(N_\psi)$ for SGD-ESN.

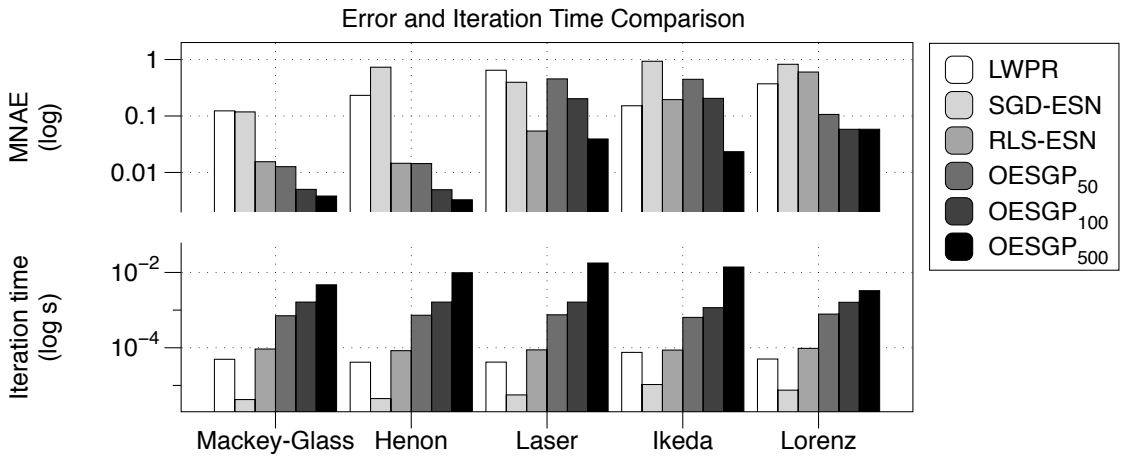


Figure 3.4: MNAE and iteration time on the one-step prediction task on the clean benchmark problems.

Problem	LWPR	SGD-ESN	RLS-ESN	OESGP$_{50}$	OESGP$_{100}$	OESGP$_{500}$
Mackey-	0.1232 (0.0008)	0.1181 (0.0718)	0.0155 (0.0471)	0.0127 (0.0021)	0.0050 (0.0004)	**0.0038** (0.0005)
Glass	0.0314 (0.0000)	0.0350 (0.0222)	0.0043 (0.0130)	0.0035 (0.0007)	0.0014 (0.0001)	**0.0011** (0.0001)
Henon	0.2317	0.7306 (0.0380)	0.0145 (0.0035)	0.0144 (0.0014)	0.0049 (0.0004)	**0.0033** (0.0001)
	0.2046	0.6064 (0.0306)	0.0135 (0.0032)	0.0143 (0.0013)	0.0055 (0.0003)	**0.0039** (0.0001)
Laser	0.6456	0.3953 (0.0307)	0.0541 (0.0020)	0.4538 (0.0761)	0.2024 (0.0455)	**0.0392** (0.0045)
	0.1458	0.1075 (0.0062)	**0.0198** (0.0011)	0.1208 (0.0254)	0.0569 (0.0117)	0.0210 (0.0017)
Ikeda	0.1517	0.9311 (0.0358)	0.1951 (0.0083)	0.4463 (0.0598)	0.2051 (0.0508)	**0.0233** (0.0040)
	0.0832	0.5039 (0.0218)	0.1177 (0.0048)	0.3049 (0.0631)	0.1418 (0.0501)	**0.0253** (0.0047)
Lorenz	0.3712	0.8230 (0.0301)	0.5997 (0.0820)	0.1066 (0.0414)	**0.0581** (0.0062)	0.0582 (0.0067)
	0.0494	0.1118 (0.0040)	0.0808 (0.0120)	0.0182 (0.0049)	**0.0133** (0.0012)	0.0133 (0.0013)
NP1	0.0859 (0.0016)	0.2888 (0.0658)	0.0580 (0.0023)	0.1308 (0.0189)	0.0596 (0.0018)	**0.0528** (0.0009)
	0.0910 (0.0013)	0.2502 (0.0497)	0.0572 (0.0019)	0.1451 (0.0264)	0.0589 (0.0016)	**0.0533** (0.0013)
NP2	0.4742 (0.0110)	0.9586 (0.3606)	0.8940 (0.4376)	0.1348 (0.0118)	**0.1325** (0.0139)	0.1354 (0.0117)
	0.2598 (0.0027)	0.4580 (0.1437)	0.3950 (0.1667)	0.0804 (0.0052)	**0.0797** (0.0056)	0.0807 (0.0053)

Table 3.3: MNAE (top) and RMSE (bottom) scores for the one-step prediction task on benchmark and noisy problems. Standard deviations are shown in brackets and lowest error scores are in **bold**.

	NORMA	KNLMS	SSP	KRLS	OESGP$_{50}$	OESGP$_{100}$	OESGP$_{500}$
NP1	0.1051	0.0197	0.0184	0.0173	0.0408 (0.0150)	0.0065 (0.0004)	**0.0053** (0.0003)
NP2	0.56	0.20	0.21	0.17	0.0233 (0.0112)	**0.0229** (0.0110)	0.0236 (0.0114)

Table 3.4: NMSE scores for the one-step prediction task on noisy problems NP1 and NP2. Lowest error scores are in **bold**.

LWPR, being a local method, also has favourable computational costs. That said, even when using only 50 BVs, OESGP achieves better scores compared to the other algorithms for the Mackey-Glass, Henon, Lorenz and NP2 problems. At this level of sparsity, each iteration remains reasonably fast at $\approx 10^{-3}$ s while achieving better accuracy and providing uncertainty values.

Focussing on the noisy problems NP1 and NP2, we observed that OESGP performs marginally better than RLS-ESN on the NP1 problem and significantly better than all other algorithms on the NP2 problem. To compare OESGP against recently published results demonstrating the performance of state-of-the-art online methods (KNLMS [Richard et al., 2009], KRLS [Engel et al., 2004], NORMA [Kivinen et al., 2004] and sparse sequential projection (SSP) [Kadirkamanathan and Niranjan, 1993]), we computed the normalised mean-square prediction error,

$$\text{NMSE} = \sum_{t=t_s}^{T_s} (d_t - \hat{y}_t)^2 / \sum_{t=t_s}^{T_s} d_t^2 \tag{3.21}$$

over the testing portion (from sample 5001 onwards). Our error scores along with results reproduced from [Richard et al., 2009] are shown in Table 3.4. We observed that OESGP's scores are an improvement of almost an order of magnitude over the published results.

3.4 DISCUSSION

Our experimental results demonstrate that the OESGP surpasses the other online methods in terms of lower error scores on the test problems, but this was generally achieved with a higher computational cost. This was not entirely surprising since OESGP produces variances which require additional computation time and is a global (sparse) method compared to LWPR, which uses smaller local models. In this section, we discuss how OESGP's computational cost can be controlled by limiting the maximum size of the BV set and how this affects the error through a comparison to the full ESGP and with large reservoirs.

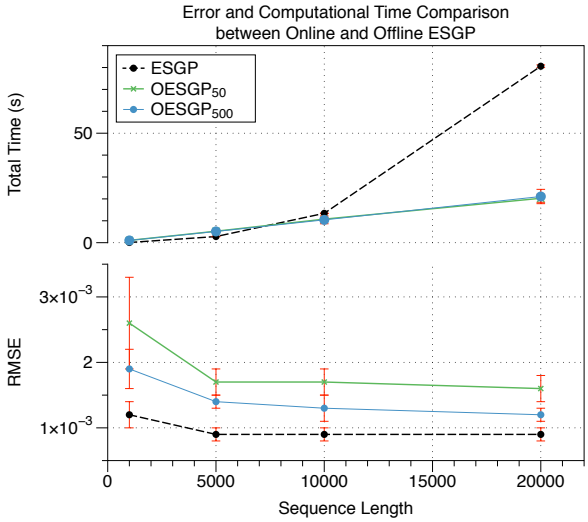


Figure 3.5: Compared to the full ESGP, our online method produces RMSE scores are higher, but at a significantly lower computational cost. This allows our method to be used in scenarios impossible for the full GP to address given current processing capabilities.

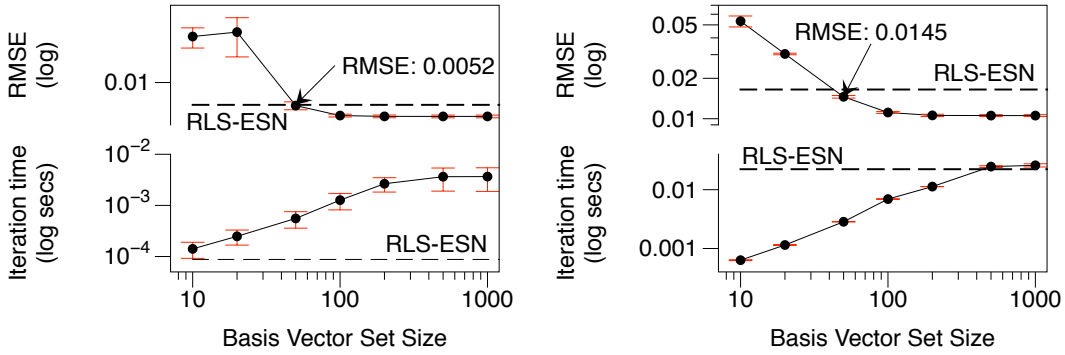
3.4.1 *Comparison to the Full ESGP*

For this comparison, we focus on the Mackey-Glass series with increasing sequence lengths $N = 1000, 5000, 10000, 20000$ time-steps. Both the offline and online algorithms were initialised with the same hyperparameters ($l = 20$, $\sigma_n = 0.005$) and used the same reservoir. We trained the methods with the first 50% of the sequence and tested on the remaining half. As quantitative metrics, we computed the RMSE scores (over the testing sequence) and the *total* computational time (training and testing)[3].

Results obtained over fifteen independent runs are shown in Fig. 3.5. For this dataset, the additional error incurred by the online approach is relatively small: a RMSE difference of $\approx 0.5 \times 10^{-3}$. The computational gains, on the other hand, are significant for the long sequences. At 20,000 time-steps, the OESGP with (a maximum of) 500 BVs took 25s; a quarter of the total time taken by the full ESGP[4]. Since the growth is *linear* in the total number of time-steps (constant per

3 To ensure that computational times were comparable, both algorithms we implemented in MATLAB. The full ESGP utilised the GPML toolbox [Rasmussen and Williams, 2006].

4 Even though the maximum BV set size was 500, the model only accumulated $\approx$ 120 BVs. As such, the difference in computational time between the 50 and 500 BV OESGP was small.



(a) Mackey-Glass (1D, 100 neurons) (b) Lorenz (3D, 1000 neurons)

Figure 3.6: Computational cost of OESGP with varying basis vector set sizes.

iteration given $s_{\mathcal{B}}$), this difference will only grow as more samples are processed. This matches up with our expectations; the ESGP has a computational time of $O(n^3)$ for training, $O(n)$ and $O(n^2)$ for predicting the mean and variance respectively. The online approach requires only $O(s_{\mathcal{B}}^2)$ *per iteration* where $s_{\mathcal{B}} \ll N$.

3.4.2 *Trading-off Computational Cost and Accuracy*

As seen in the experiments, OESGP's computational cost can be controlled by limiting the size of the basis vector set. Up to this point, our experiments used small reservoirs. Since the computational complexity for OESGP is $O(s_{\mathcal{B}}^2 + N_\psi s_{\mathcal{B}})$ per iteration, we see that for large reservoir sizes and small $s_{\mathcal{B}}$, OESGP may be cheaper than RLS-ESN, which takes $O(N_\psi^2)$. To illustrate this point, we trained OESGP using various maximum sizes for the BV set, $s_{\mathcal{B}} = 10, 20, 50, 100, 200, 500, 1000$ on the Mackey-Glass and Lorenz problems (training and testing conducted throughout). For comparison, we used reservoirs of 100 and 1000 neurons for the Mackey-Glass and Lorenz problems respectively.

The RMSEs and iteration times (averaged over 10 independent runs) are shown in Fig. 3.6. As expected, we observed a positive relationship between the $s_{\mathcal{B}}$ and the average iteration time and an inverse relationship with the error. For the Mackey-Glass problem, to achieve an error rate comparable to that of RLS-ESN, $s_{\mathcal{B}} = 50$ where the computation time is $\approx 10^{-3}$s per iteration com-

pared to $\approx 10^{-4}$ per iteration for RLS-ESN. On the Lorenz problem however, we find that at the same BV set size, $s_{\mathcal{B}} = 50$, OESGP (taking 0.0028s per iteration) not only obtained a lower mean error (0.0145) but was significantly faster than RLS-ESN (0.022s per iteration). Therefore, for larger reservoir sizes (e.g, the large hierarchical reservoirs of up to 20,000 neurons used in phoneme recognition [Triefenbach et al., 2010]), OESGP may prove to be more efficient and accurate than RLS-ESN, while simultaneously providing uncertainties.

3.5 ACTIVITY RECOGNITION WITH GENERATIVE CLASSIFICATION

In this section, we venture into action learning and recognition using the MSRAction3D Skeleton (MSRAS) dataset [Li et al., 2010]. The MSRAS contains 20 actions performed by 10 different subjects repeating each action two to three times, recorded using a Kinect-like device. For each action, the skeletal data consists of a sequence of 3-dimensional coordinates and a confidence value (indicating how well each point was tracked) for 20 joints (examples shown in Fig 3.8). During our preliminary examination, several recordings were found to be missing significant chunks of the action sequence, presumably when the tracked skeleton was lost. As such, we removed recordings that had an average confidence below 55%[5]; of the initial 567 sequences, 544 (96%) were retained after this process.

For this problem, we used a generative modelling approach whereby each action class $a_i \in \mathcal{A}$ was represented by a separate OESGP model, $m_i \in \mathcal{M}$ (Fig. 3.7). Each model was trained to predict the velocity of each joint coordinate (20×3 outputs) given the current velocities (20×3 inputs). Inference was then performed using a Bayes filter to iteratively update a probability distribution over the model classes:

$$p_t(a_i|o, \mathcal{M}) = \frac{p(o_t|m_i)p_{t-1}(a_i)}{\sum_j p(o_t|m_j)p_{t-1}(a_j)} \tag{3.22}$$

5 At 50%, only 490 sequences were left after the removal and at 60%, too many erroneous sequences remained.

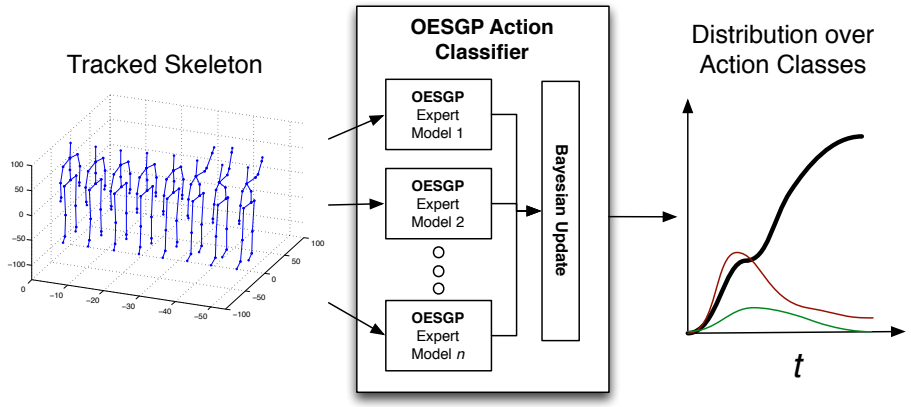


Figure 3.7: Schematic of our OESGP-based Generative Action Classifer. Each OESGP regressor models a separate action and predicts the next-step velocities of joint coordinates (of the tracked skeleton) given the current velocities. The probability distribution over the action classes was then updated using the actual observation and a Bayes filter.

where a_i is the action class i at time t and o is the observation (the velocities of the joints). We used a straightforward observational model where each coordinate was treated separately:

$$p(o_t|m_i) = \prod_{k}^{60} N(o_{k,t}|y_{i,k,t}, \sigma_{i,k,t}) \tag{3.23}$$

where $y_{i,k,t}$ and $\sigma_{i,k,t}$ are the k^{th} predicted mean and standard deviations using model m_i. The initial prior was assumed to be uniformly distributed and the most probable class at the end of the sequence was selected as the recognised action. In addition to being conceptually straightforward, this approach gives a probability distribution over the different actions during the entire course of the observed sequence. In our experiments, all OESGPs were initialised using the same parameters: $l = 2.0$, $\sigma_n = 0.05$ and capacity of 100 BVs with reservoirs of 100 neurons and 0.1 connectivity.

Following [Li et al., 2010], we used three subsets of eight actions (shown in Fig. 3.8e) and conducted two separate experiments. For the "same-subject" experiment, training was conducted for each participant on trial 1 and 2, and tested on final trial. For the "cross-subject" experiment, training was conducted on all trials for subjects one through five and tested on recordings from sub-

		Action Graph	OESGP
Same-Subject	AS1	0.934	0.950 (0.007)
	AS2	0.929	0.952 (0.012)
	AS3	0.963	0.965 (0.007)
	Overall	0.942	**0.955** (0.068)
Cross-Subject	AS1	0.729	0.806 (0.036)
	AS2	0.719	0.749 (0.064)
	AS3	0.792	0.871 (0.028)
	Overall	0.747	**0.809** (0.068)

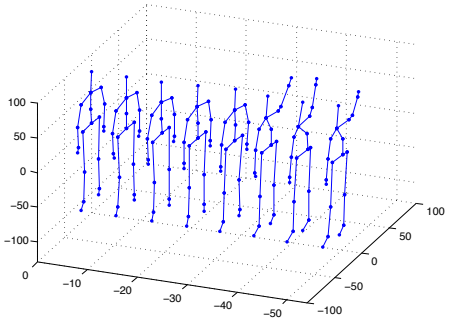
Table 3.5: Accuracy scores for the MSRAction3D dataset.

jects six through ten. Of the two, the cross-subject task is more challenging since different participants performed a given action in a dissimilar manner (Fig. 3.8).

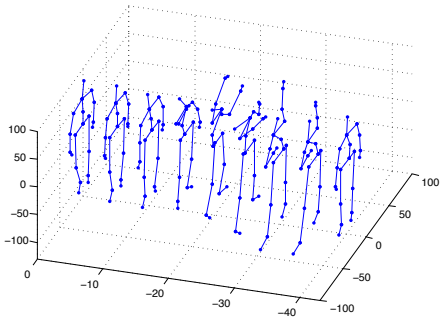
Classification accuracies averaged over 10 runs are shown in Table 3.5. For comparison, we have included the scores obtained using the action graph method [Li et al., 2010] (care should be taken when comparing these results since [Li et al., 2010] used the *full* depth map version of the dataset instead of the tracked skeletons). The OESGP-based classifier achieved comparably high scores (above 95%) for the same-subject test, even without fine-tuning of parameters for individual action models. Moreover, we obtained substantially higher accuracies for the cross-subject test across all three action subsets (overall score of 80.9%). Since only a *single pass* was made through each sequence, training was rapid (requiring an average of 0.0006s per iteration). Predictions were more costly at 0.0089s per iteration since updates had to be made across all action classes, but well above the 30Hz required for real-time use. Therefore, models can be trained and used in real-time (e.g., for interactive robot-learning-by-demonstration scenarios) given reasonable limits on the number of action classes.

3.6 CONCLUSION

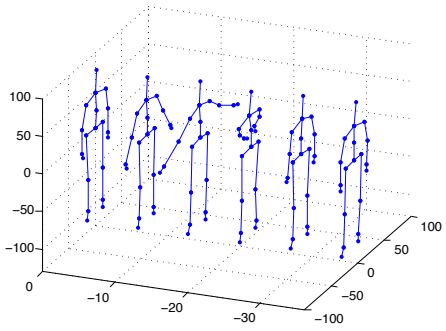
In this chapter, we presented the Online Echo-State Gaussian Process (OESGP), a sparse Bayesian formulation of the echo-state network. In contrast to existing methods, the OESGP has the capability to *learn online*, *provide uncertainty estimates* and *model temporal dynamics*. Experiments on a range of problems demon-



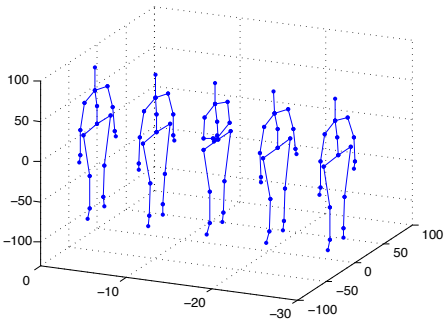
(a) S1: High Arm Wave



(b) S1: Tennis Serve



(c) S1: Hand Clap



(d) S5: Hand Clap

AS1	AS2	AS3
Horizontal arm wave	High arm wave	High throw
Hammer	Hand catch	Forward kick
Forward punch	Draw x	Side kick
High throw	Draw tick	Jogging
Hand clap	Draw circle	Tennis swing
Bend	Two hand wave	Tennis serve
Tennis serve	Forward kick	Golf swing
Pickup & throw	Side boxing	Pickup & throw

(e) Action Sets (AS) 1 to 3 [Li et al., 2010]

Figure 3.8: Four sequences from the MSRAction3D Skeletal dataset where examples 3.8a-3.8c were performed by subject one. In 3.8d, note how the same action (hand clap) was performed in a different manner (without outstretched arms) by subject 5. Table 3.8e shows the three action subsets used in the study.

strated OESGP's effectiveness; it obtained significantly better error scores compared to existing online ESNs trained using RLS and SGD as well as the popular LWPR algorithm. As a case study, we used OESGP in a Bayes generative classifier for online action recognition that achieves state-of-the-art performance.

One area that we have not addressed is that of hyper-parameter adaptation. As we might expect, the performance of RLS-ESN, SGD-ESN and OESGP are dependent on their parameters. For example, RLS-ESN relies on a proper choice of δ and λ. Under suboptimal settings, convergence could be slow or yield suboptimal weights. Unlike LWPR, parameter adaptation is not currently a feature of any of online ESN training algorithms.

In the case of OESGP, accuracy depends on the ESN parameters, the chosen kernel and noise hyperparameters, as well as the maximum capacity of BV set. As discussed in the previous section, the maximum capacity would be determined by the computational requirements of the application. In the next chapter, we first simplify our method by eliminating the explicit construction of a reservoir and present an "infinite" echo-state network with a novel recursive kernel. By doing so, the spectral radius becomes part of the kernel, which can then be optimised along with other kernel parameters. Moreover, by incorporating automatic relevance determination (ARD) in the recursion, we show how relevant features can be selected (which is difficult under the regular echo-state network paradigm).

4

ONLINE INFINITE ECHO-STATE GAUSSIAN PROCESS

In the previous chapter, we observed that the combination of the echo-state network and the online GP achieved excellent performance a variety of benchmark problems. But there's one caveat: the OESGP relies on a proper reservoir which, according to the typical ESN procedure, is randomly constructed. Although there has been great strides in understanding how the structure of reservoirs impacts performance [Jaeger, 2005; Lukoševičius, 2012; Lukoševičius and Jaeger, 2009], specification of the reservoir parameters remains somewhat of a "black art"; creating the right reservoir often requires not only skill but "numerous trials and even luck" [Rodan and Tino, 2011].

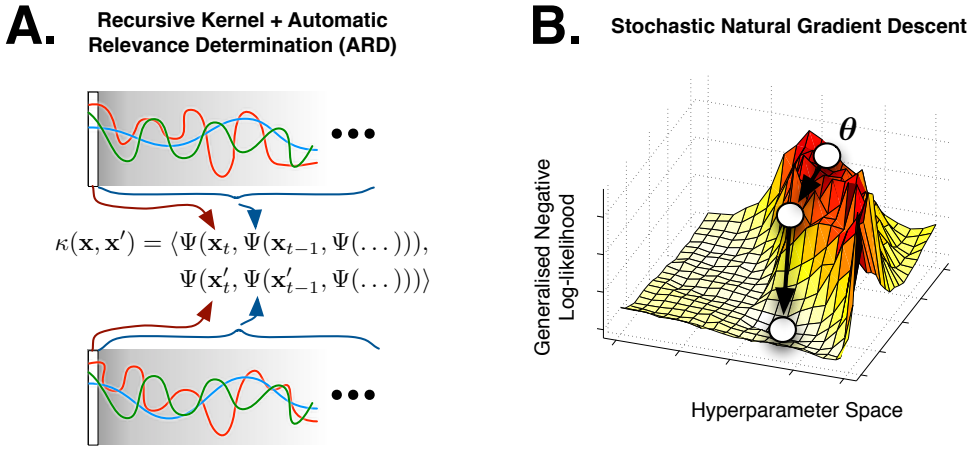


Figure 4.1: This chapter describes two main changes to OESGP. The first is the introduction of a recursive kernel with automatic relevance determination (ARD) for multivariate time-series where dimensions may have varying importance. We then show how the hyperparameters of this new kernel can be optimised in an online manner via stochastic natural gradient descent on the generalised negative likelihood.

In this chapter, we adopt a simplifying perspective: the reservoir is regarded as spatio-temporal kernel where the features are constructed explicitly within a finite neuronal substrate. Similar in spirit to how neural networks were extended to GPs, the construction of an explicit reservoir can be eliminated by considering reservoirs with an *infinite* number of neurons. This approach has resulted in an entire class of *recursive kernels* [Hermans and Schrauwen, 2011]. In this thesis, we derive and propose a novel recursive kernel with automatic relevance determination (ARD); the ability to specify and/or distinguish the most pertinent features. When combined with the online GP, this yields the *Online Infinite Echo State Gaussian Process* (OIESGP).

Training the OIESGP consists of two interrelated parts. With the finite OESGP, we were primarily concerned with the first part: updating the posterior mean and covariance, and maintaining a set of novel states or basis vectors. In this chapter, we address the second equally important portion: adapting the kernel hyperparameters. While some hyperparameters can be easy to ascertain, others such as the characteristic lengthscale may not be immediately obvious. We propose learning the hyperparameters in an online manner by *stochastic natural gradient descent* [Amari, 1998; Amari et al., 2000]. When used with the recursive ARD kernel, the OIESGP learns the impact of not only the spatial features, but also the relevancy of the past. This improves the method's adaptability and allows us to gain insights into the underlying system by inspecting the resulting hyperparameters.

The remainder of this chapter is organised as follows: in the next section, we review recursive kernels and present a new recursive kernel that can be used in both offline and online kernel-based machine learning methods. Section 4.2 describes the online hyperparameter optimisation method with an illustrative example. Numerical results on benchmarks are presented in Section 4.3, demonstrating OIESGP's effectiveness, particularly on problems with irrelevant dimensions. Section 4.4 presents an analysis of our experimental results using ideas from dynamical systems theory. Finally, 4.5 concludes this chapter with a summary.

4.1 THE INFINITE ECHO STATE GAUSSIAN PROCESS

Recall from Chapter 2 that the covariance function or *kernel* plays a significant role in GPs (and other kernel-based machine learning methods such as Support Vector Machines); intuitively, it defines the notion of closeness or *similarity* between the data points $\mathbf{x}$ [Rasmussen and Williams, 2006]. From a different perspective, the kernel defines the space of real functions (specifically, a Reproducing Kernel Hilbert space or RKHS) in which we expect our "true" function to live. The kernel projects our inputs to a higher dimensional space via an implicit map ϕ — *any* valid kernel is equivalent to a inner product between two mapping functions,

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}. \tag{4.1}$$

where $\mathcal{H}$ is our RKHS space. Since this inner product is effectively computed by the kernel function, it allows us to work in a higher (possibly infinite) dimensional space without having to pay a hefty computational price.

For time-series regression, if the input space is one-dimensional, one can consider a direct application using a sliding-window approach where we construct an "augmented" observational element $\hat{\mathbf{x}}_t = [x_t, x_{t-1}, \ldots, x_{t-\tau}]$ where x_t is the observation at time t. We can then use the aforementioned SE kernel (or any applicable standard kernel). Although this method can be effective, things become less straight-forward when each data point is multi-dimensional, i.e. $\hat{\mathbf{x}}_t = [\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-\tau}]$, which is typically the case when dealing with multiple sensors or actuators. It then becomes necessary to vectorise the matrix $\hat{\mathbf{x}}_t$ and important structural information can be lost in the process. Ideally, we would like the kernel to take into account the temporal nature of sequential observations.

Indeed, the ESN reservoir can be regarded as a spatio-temporal kernel which acts upon time-series or histories rather than individual data-points. Unlike the SE-kernel however, the projected features are *explicitly* computed and expressed as the reservoir state via Eq. (2.32). For typical ESNs, this iterated projection is computed by a *random matrix*. The use of a reservoir together with the SE-kernel

(as in the OESGP) can be seen as a two-step kernel whereby the time series is first randomly projected onto the neuronal state space (which is typically of *lower dimension* compared to the complete history), followed by a second implicit projection defined by k^{SE}. Then, a natural follow-up question is whether it would be possible to simplify this approach into a single "joint" kernel and remove the random element. As it turns out, the answer is in the affirmative and takes the form of recursive kernels, described in next section.

4.1.1 *Recursive Kernels*

Recursive kernels [Hermans and Schrauwen, 2011] are a class of kernels that share an intimate relationship with recurrent neural networks (RNNs) and were principally derived as a means to extend RNNs to infinite size. Consider a recurrent network with internal weights $\mathbf{W}$, input weights $\mathbf{V}$ and internal state $\mathbf{s}$. Upon encountering input $\mathbf{x}_t$ at time t, the RNN output is:

$$\hat{\mathbf{y}}_t = h(\mathbf{V}\mathbf{x}_t + \mathbf{W}\mathbf{s}_t) \tag{4.2}$$

where h is a combination of an activation function (such as the hyperbolic tangent) and a projection. The fundamental concept behind the recursive approach is that (4.2) can be written as:

$$h(\mathbf{W}\mathbf{s}_t + \mathbf{V}\mathbf{x}_t) = h\left([\mathbf{W}|\mathbf{V}]\begin{bmatrix}\mathbf{s}_t\\ \mathbf{x}_t\end{bmatrix}\right) \tag{4.3}$$

i.e., a function of the concatenation of the input with the previous internal state. Realising that the same reasoning could be applied to kernel functions, Hermans and Schrauwen [Hermans and Schrauwen, 2011] showed that recursive variants of kernels with the form $k(\mathbf{x}, \mathbf{x}') = f(\|\mathbf{x} - \mathbf{x}'\|^2)$ and $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x} \cdot \mathbf{x}')$ could be

derived in a similar manner. As an example, the recursive-SE kernel has the form:

$$\kappa_t^{\text{SE}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}_t - \mathbf{x}'_t\|^2}{2l^2}\right) \exp\left(\frac{\kappa_t^{\text{SE}}(\mathbf{x}, \mathbf{x}') - 1}{\sigma_\rho^2}\right) \tag{4.4}$$

Note that recursive kernels are denoted with the symbol κ to differentiate them from standard kernels. Although the recursive kernel can be theoretically applied to time-series of arbitrary length, in practical settings, it becomes necessary to limit the recursion depth (specified by a parameter τ).

4.1.2 *Recursive Kernel with Input Relevance*

In this work, we propose a recursive kernel based on automatic relevance detection (ARD) [Neal, 1996], which has the form:

$$\kappa_t^{\text{ARD}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}(\mathbf{x}_t - \mathbf{x}'_t)^{\mathrm{T}}\mathbf{M}(\mathbf{x}_t - \mathbf{x}'_t)\right) \exp\left(\frac{\kappa_{t-1}^{\text{ARD}}(\mathbf{x}, \mathbf{x}') - 1}{\sigma_\rho^2}\right) \tag{4.5}$$

where $\mathbf{M} = \text{diag}(\mathbf{l})^{-2}$, a diagonal $d \times d$ matrix where $\mathbf{l} = [l_i]_{i=1}^d$. To keep the flow of the main text, the derivation of this kernel has been deferred to Appendix B.

Unlike the standard squared exponential, this ARD kernel is *anisotropic*: varying the l_i's for different inputs allows us to control the impact that the inputs have on the predictions. From (4.5), we can see that the kernel function's responsiveness to input dimension k is inversely related to l_k. In general, $\mathbf{M}$ need not be diagonal; the parameterisation $\mathbf{M} = \Lambda\Lambda^\top + \text{diag}(\mathbf{l})^{-2}$ is called the factor analysis distance since Λ can be used to identify relevant directions among the observed inputs dimensions [Rasmussen and Williams, 2006].

As can be seen from the RHS of (4.5), the parameter σ_ρ weights the previous recursion kernel value. Intuitively, we can view it as a *temporal lengthscale* between the present inputs and the past; if σ_ρ is very large, the past recursive kernel value will cease to be relevant. Interestingly, this parameter is also related to the spectral radius ρ in ESNs and affects the stability of the kernel [Hermans

and Schrauwen, 2011]. If the inverse lengthscale $\sigma_\rho^{-1} < 1$, the kernel is stable and iteratively applications of the function will cause a convergence to a fixed point. The closer σ_ρ^{-1} is to 1, the slower this rate of decay will be. The spectral radius of the ESN performs the same role: the larger ρ is, the slower the network states will decay (the rate of "memory fade" is slower). Note however, if $\sigma_\rho^{-1} > 1$, this property no longer applies and instabilities may exist that cause orbits to diverge.

From one perspective, κ_t^{ARD} is a generalisation of both the SE and recursive-SE kernels; if all l_i's are equal, κ_t^{ARD} reduces to the regular SE recursive kernel. If recursion is not applied, it further reduces to the standard SE kernel. The principal advantage of this generalisation is that it allows for feature weighting/selection while maintaining the intuition that spatial-elements at a time t "belong together". Such input weighting can be difficult to achieve in regular reservoir approaches[1]. Moreover, the temporal lengthscale parameter introduces a single parameter to control the relevancy of the past; if we were to apply the regular ARD kernel on the time-series, there would be a lengthscale for each input at each time-step. Although this "full ARD" kernel can be used to identify relevant inputs at distinct time-steps, it requires the specification of $T_w \times d$ lengthscale parameters where T_w is the length of the sliding window.

4.1.2.1 *Recursive ARD Kernel Properties*

Before proceeding, we isolate important kernel properties. In particular, we show that the recursive ARD kernel is valid and anisotropic stationary (translation invariant). We also present (recursive) kernel derivatives, needed for gradient-based optimisation of its hyperparameters.

VALIDITY κ^{ARD} is a valid covariance function, i.e., it induces a covariance matrix $\mathbf{K}$ that is symmetric and positive semidefinite and obeys Mercer's Theorem such that $\kappa^{\text{ARD}}(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ where $\phi(\cdot)$ is a map from the input

1 Typically, the inputs weights are set to 1 are not adapted. Even changing the input weights may help only to a degree since the internal weights that propagate signals in the reservoir are not adapted.

space to the feature space $\mathcal{H}$. This property is a requirement for use in a GP or SVM.

Proof: The kernel is valid from the construction given in Appendix B. In addition, we offer an alternative induction proof that relies on the fact that sums, products and exponents of kernels yield valid kernels [Genton, 2002]; Consider the base case as the start of the time-series at $t = 1$ where κ_1^{ARD} is regular, valid, squared exponential ARD kernel. Our inductive hypothesis is that κ_k^{ARD} at some time $t = k$ is valid. The kernel $\kappa_{k+1}^{\text{ARD}}$ can be written as:

$$\begin{aligned}\kappa_{k+1}^{\text{ARD}}(\mathbf{x}, \mathbf{x}') = &\exp\left(-\frac{1}{2}(\mathbf{x}_{k+1} - \mathbf{x}'_{k+1})^{\mathrm{T}}\mathbf{M}(\mathbf{x}_{k+1} - \mathbf{x}'_{k+1})\right) \\ &\exp\left(\frac{\kappa_k^{\text{ARD}}(\mathbf{x}, \mathbf{x}')}{\sigma_\rho^2}\right)\exp\left(-\frac{1}{\sigma_\rho^2}\right) \end{aligned} \tag{4.6}$$

i.e., a multiplication of three exponentiated terms where the last term on the RHS is simply a constant. The first two terms are valid kernels; the first is an ARD kernel on the current input. The second term is a composition of a division by σ_ρ^2 and an exponentiation, which yields a valid kernel [Rasmussen and Williams, 2006; Genton, 2002]. Therefore, by induction, the recursive ARD kernel is valid up to any finite time-step T. □

ANISOTROPIC STATIONARITY A kernel is said to be stationary if it is translation invariant, i.e., only a function of $\mathbf{x} - \mathbf{x}'$. From (4.5), it is clear that this is the case for κ^{ARD}. The use of varying lengthscales for each dimension implies that κ^{ARD} is inherently anisotropic (directionally dependent). This is in contrast to the standard isotropic k^{SE} which is invariant to all rigid motions (note however that the recursive κ^{SE} is anisotropic with regard to the space of time-series since there are separate spatial and temporal lengthscales).

KERNEL DERIVATIVES Kernel gradients are necessary for optimising the hyperparameters (discussed in the next section). The derivatives for κ_t^{ARD} are recursive in nature:

$$\frac{\partial \kappa_t^{\text{ARD}}}{\partial l_i} = \kappa_t^{\text{ARD}} \left[\frac{1}{\sigma_\rho^2} \frac{\partial \kappa_{t-1}^{\text{ARD}}}{\partial l_i} + \frac{\beta_i}{l_i^3} \right] \tag{4.7}$$

$$\frac{\partial \kappa_t^{\text{ARD}}}{\partial \sigma_\rho} = \kappa_t^{\text{ARD}} \left[\frac{-2(\kappa_{t-1}^{\text{ARD}} - 1)}{\sigma_\rho^3} + \frac{1}{\sigma_\rho^2} \frac{\partial \kappa_{t-1}^{\text{ARD}}}{\partial \sigma_\rho} \right] \tag{4.8}$$

where $\beta_i = (x_{t,i} - x'_{t,i})^2$ with base cases:

$$\frac{\partial \kappa_1^{\text{ARD}}}{\partial l_i} = \frac{\beta_i}{l_i^3} \kappa_1^{\text{ARD}} \quad \text{and} \quad \frac{\partial \kappa_1^{\text{ARD}}}{\partial \sigma_\rho} = 0$$

4.2 ONLINE HYPERPARAMETER ADAPTATION

Applying the recursive ARD kernel requires us to specify its hyperparameters θ. The full Bayesian solution would be to place priors over the hyperparameters and compute posterior probability distributions given the data. However, this approach typically requires the evaluation of intractable integrals.

A common approximation is to maximise the marginal likelihood $p(\mathbf{y}|X, \theta)$ over the training set (referred to as type-II maximum likelihood estimation or ML-II for short). The downside of ML-II is the risk of over-fitting the training data. In our online GP, this problem is exacerbated since we have in storage only the basis vectors, which make up a small representative sample.

What we are principally interested in is minimising the error over *unseen* test samples, i.e., the *generalisation error*. In the online setting, we are continuously being provided with samples that we have yet to incorporate into our training set. As such, we use the alternative approach of optimising the hyperparameters with regard to the likelihood of leave-one-out sample. In this section, we give details on how we can perform this optimisation by following the natural gradient of a cost function.

4.2.1 *Stochastic Natural Gradient Descent*

Let us first define the cost function we want to minimise:

$$\mathcal{L}(\theta) = -\int \log p(y_t|\mathbf{x}_t, \theta) p(\mathbf{x}_t, y_t) d\mathbf{x}_t dy_t \tag{4.9}$$

where, for notational convenience, we have dropped the dependence on the training data seen thus far. In words, we want the negative log likelihood of the over the samples generated by the underlying dynamical system to be as small as possible. In our case,

$$\log p(y_t|\mathbf{x}_t, \theta) = \log \mathcal{N}(y_t - \mu_t, \sigma_t^2) \tag{4.10}$$

$$= -\frac{1}{2}\log \sigma_t^2 - \frac{(y_t - \mu_t)^2}{2\sigma_t^2} - \frac{1}{2}\log 2\pi \tag{4.11}$$

where μ_t and σ_t^2 are given by (3.17) and (3.18) respectively. If we had the entire training data available[2], θ could be optimised by following the gradient on the training set:

$$\nabla_\theta \tilde{\mathcal{L}} = \frac{\partial \tilde{\mathcal{L}}(\theta)}{\partial \theta} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial \log p(y_t|\mathbf{x}_t, \theta)}{\partial \theta} \tag{4.12}$$

where $\tilde{\mathcal{L}}$ is an approximation to the true cost. Note that naïve optimisation via the training gradient will lead to over-fitting the training set. In the online case, we approximate the gradient from observed samples:

$$\nabla_\theta \tilde{\mathcal{L}} = \frac{\partial \tilde{\mathcal{L}}(\theta)}{\partial \theta} = \frac{1}{s_g}\sum_{t=k}^{k+s_g} \frac{\partial \log p(y_t|\mathbf{x}_t, \theta)}{\partial \theta} \tag{4.13}$$

where s_g is the sampling interval. With standard stochastic gradient descent, we would update our current estimate θ_j using the following update

$$\theta_{j+1} = \theta_j + \eta \nabla_\theta \tilde{\mathcal{L}} \tag{4.14}$$

2 Let us assume the training data consists of a very long sequence to keep things concise. If it was several sample time-series, we would simply average over all the samples and time-steps.

where η is the step size or learning rate.

In Euclidean space, the gradient points in the direction of steepest descent (since we are minimising cost). However, when the parameter space is a non-Euclidean manifold, the conventional gradient may mislead causing slow convergence to the minimum [Amari, 1998]. Instead, the direction of steepest descent is given by the *natural gradient* [Amari, 1998]:

$$G_\theta^{-1} \nabla_\theta \mathcal{L} \tag{4.15}$$

where G_θ^-1 is the inverse Riemannian metric tensor (the covariance of the gradients). For the space of probability distributions represented by the hyperparameters, G_θ is the Fisher Information matrix:

$$\mathcal{I}_\theta = \left[\mathbb{E} \left[\frac{\partial \log p(y_t|\mathbf{x}_t, \theta)}{\partial \theta_i} \frac{\partial \log p(y_t|\mathbf{x}_t, \theta)}{\partial \theta_j} \right] \right]_{i,j=1}^{|\theta|} \tag{4.16}$$

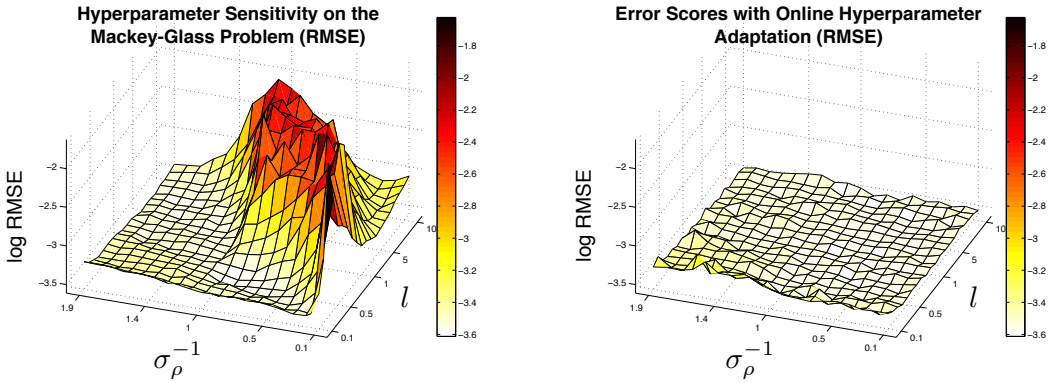
In this work, we approximate the natural gradient over s_g iterations using an averaging approach with the Sherman-Morrison formula (A.4):

$$\tilde{\mathcal{I}}_\theta^{-1} = \frac{\epsilon_t}{1-\epsilon_t} \tilde{\mathcal{I}}_\theta^{-1} - \frac{\epsilon_t}{1-\epsilon_t} \frac{(\tilde{\mathcal{I}}_\theta^{-1} \nabla_\theta \tilde{\mathcal{L}})^\top (\nabla_\theta \tilde{\mathcal{L}}^\top \tilde{\mathcal{I}}_\theta^{-1})}{(1-\epsilon_t) + \epsilon_t \nabla_\theta \tilde{\mathcal{L}}^\top (\tilde{\mathcal{I}}_\theta^{-1} \nabla_\theta \tilde{\mathcal{L}})} \tag{4.17}$$

where $\epsilon_t = 1/s_g$. A similar approach was used in [Amari et al., 2000] for multilayer perceptrons but here, we average over the sampling interval and update the hyperparameters progressively:

$$\theta_{j+1} = \theta_j + \eta \tilde{\mathcal{I}}_\theta^{-1} \nabla_\theta \tilde{\mathcal{L}} \tag{4.18}$$

Once the hyperparameters are updated, the GP posterior is out-of-date, forcing a re-computation of $\mathbf{K}$, $\mathbf{C}$ and $\mathbf{Q}$ which we perform directly using the basis vector (BV) set $\mathcal{B}$. This causes a loss of information since observations that were absorbed into $\mathbf{C}$ are not represented by $\mathcal{B}$. In other words, the optimisation causes the method to "forget" some of the training samples previously seen. Depending on the application, this is a mixed blessing; the information loss typically leads



(a) Hyperparameter Sensitivity (RMSE)

(b) Hyperparameter Adaptation (RMSE)

Figure 4.2: Adaptation and performance sensitivity to recursive Gaussian kernel prior hyperparameters. Fig. 4.2a The optimal hyperparameters (in terms of RMSE) are in a valley with $l \approx 0.2$ and $\sigma_\rho^{-1} \approx 0.5$. In 4.2b, we observe that the online gradient descent successfully minimised the error scores for the region where $\sigma_\rho^{-1} < 1$, flattening the error hill.

to higher errors in the short-run, but forgetting may improve the adaptability of the algorithm to non-stationary distributions. It may be possible to project the older OIESGP onto one with new hyperparameters (e.g., by minimising the KL divergence), while taking into the account the possibility of non-stationarity through appropriate weighting, but this remains future work.

As an example, Figure 4.2 illustrates the results of a test where we varied the characteristic and temporal lengthscales for the one-dimensional 10^5-step Mackey-Glass sequence and computed the RMSE scores over the final 2000 observations. From the error landscapes, we observe the optimal hyperparameter set sits in a valley with $l \approx 0.2$ and $\sigma_\rho^{-1} \approx 0.5$ and the RMSE increases with the respective lengthscales. In particular, we note a high error ridge as the spatial lengthscale is increased to 10, while the RMSE score was more robust against changes to the temporal lengthscale. Fig.4.2b shows the results after applying stochastic natural gradient descent starting from varying *initial* hyperparameters: the SNGD-enabled OIESGP effectively flattens the error ridge across the starting positions.

4.2.1.1 *Practical Issues*

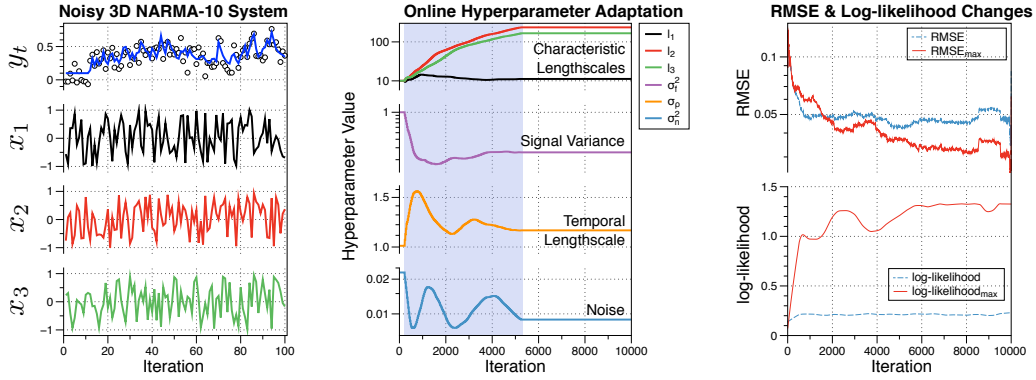
A reasonable starting position aids convergence to a good set of hyperparameters. In our experience, a simple trial of spatial lengthscales in the set $\{1, 10, 100\}$ is sufficient to find a good initial location. As a rule of thumb, spatial lengthscales should be smaller for dimensions deemed more important and setting the temporal lengthscale to 1.01 or 1.5 was adequate for our trials.

In our experiments, we found the optimisation process to be sensitive to the gradient sampling intervals. In particular, too short sampling intervals (e.g., $s_g = 1$) led to poor results because the approximated gradient was not representative of the true underlying gradient. On the other hand, too long a sampling interval led to slow convergence. We found setting the sampling interval in the range $s_g \in [10, 50]$ sufficed for our tests. In addition, the BV set $\mathcal{B}$ was fixed during the gradient sampling (similar to [Seeger et al., 2003; Csató, 2002]) to prevent fluctuations in the cost function.

The learning rate η also plays an important role; too high a learning rate causes a high variance in the hyperparameter estimates. But too low a learning rate causes slow conference to the optimum. Here, we have opted for the simple (and surprisingly effective [Bottou, 2012]) setting of $\eta = 1/j$ where j is the current sampling interval (as in Eq. 4.18).

Another important parameter to tune is the convergence criteria, which determines when hyperparameter optimisation should be stopped. In this work, we have used the squared norm of difference between updated and current hyperparameters, $||\triangle\theta_j||^2$ where $\triangle\theta_j = \theta_j - \theta_{j-1}$. In practice, we found stopping after $||\triangle\theta||^2 < c_g = 10^{-4}$ consistently over 50 iterations worked well.

Care should be taken to ensure that when the inverse matrices ($\mathbf{C}$ and $\mathbf{Q}$) are recomputed, numerical problems are avoided either by salting the matrix diagonal with small amounts of noise $\approx 10^{-6}$ or by using the Cholesky decomposition. In this work, we have opted for the former because the latter proved to be more computationally expensive in Matlab. Also, to prevent steps that are too large, we restricted the maximum coordinate change to be 0.05.



(a) Noisy NARMA-10 (b) Hyperparameter Evolution (c) Performance Changes

Figure 4.3: Online hyperparameter adaptation while learning to predict the NARMA-10 system with two irrelevant input dimensions. (4.3a) The first 100 time-steps of the three inputs and output. The true output (solid blue line) is hidden and only the noise corrupted outputs (circles) are observed. Only x_1 is relevant for generating y; the other two inputs are irrelevant. (4.3b) The blue shaded region indicates when hyper-parameter optimisation was performed; it automatically stopped after the convergence criteria was met. The hyper-parameters change to reflect the underlying characteristics of the generating system. (4.3c) The online adaptation led to a 29% decrease in prediction error and a 5-fold increase in the likelihood compared to the OIESGP with fixed hyperparameters.

Finally, to simplify hyper-parameter optimisation, we unified our parameters into a single function and modified (4.5) above to include the noise and signal variance:

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \kappa_t^{\text{ARD}}(\mathbf{x}, \mathbf{x}') + \sigma_n^2 \delta_{\mathbf{x},\mathbf{x}'} \tag{4.19}$$

where σ_f^2 is the signal variance, σ_n^2 is the noise variance, and $\delta_{\mathbf{x},\mathbf{x}'}$ is the Kronecker delta which is one iff $\mathbf{x} = \mathbf{x}'$ and zero otherwise. With this in mind, let us move on to an illustrative example of hyperparameter adaption.

4.2.2 *An Illustrative Example*

Consider a sample problem where the task is to provide an online model that learns from a three-dimensional data stream to predict a one-dimensional output. In fact, the unknown underlying generative process is the input-driven 10th order non-linear auto regressive moving average (NARMA-10) system [Atiya

and Parlos, 2000] *with two irrelevant input dimensions* and observations corrupted by noise $\epsilon \sim \mathcal{N}(0, 0.01)$. The NARMA-10 system consists of the following update equation:

$$d_{t+1} = 0.3y_{t-1} + 0.05y_{t-1}\sum_{i=0}^{9} d_{t-i} + 1.5u_{t-1}u_{t-9} + 0.1 \tag{4.20}$$

$$y_{t+1} = d_{t+1} + \epsilon \tag{4.21}$$

where the input signal $u_t \sim \mathcal{U}(0, 0.5)$. The first 100 steps of this system are shown in Fig. 4.3a; the top row illustrates the true and observed outputs, represented by the solid blue line and circles respectively. The true output in this problem is hidden. The next three rows are the inputs. The only relevant input is $x_1 = u_t$ (rescaled to [-1,1] for visualisation purposes) and both x_2 and x_3 are the two randomly generated irrelevant inputs with samples drawn from a uniform distribution $\mathcal{U}(-1, 1)$.

Our OIESGP's *initial* hyperparameters were mis-specified with equal length-scales $l_1 = l_2 = l_3 = 10$ and prior noise $\sigma_n^2 = \mathbb{V}[\mathbf{y}] = 0.0228$. The signal variance and temporal lengthscale were set at $\sigma_f^2 = 1.0$ and $\sigma_\rho = 1.01$ respectively. For this example, the gradient sampling interval was set at 25 and the convergence criteria was set at $\|\triangle\theta\|^2 < 10^{-4}$.

Fig. 4.3b summarises the results after 10^5 time-steps. Since we were optimising the hyperparameters in an online manner, we observe variations in the log-likelihood with an overall increasing trend. After meeting the convergence criteria, the characteristic length-scales for x_2 and x_3 had increased to > 150 (reducing their impact) and the lengthscale for the correct input converged to 11.2. These results strongly (and correctly) suggest that x_1 is relevant while x_2 and x_3 can be discarded. The temporal length-scale σ_ρ was approximately right; after minor initial fluctuation, it settled on a value of 1.15, indicating the past was relevant for making proper predictions. The noise variance parameter decreased from the wrong value of 0.0228 until 0.009, very close to the true value of 0.01.

These changes not only gave us insight into the underlying system, but also led to significantly lower prediction errors and higher likelihood scores com-

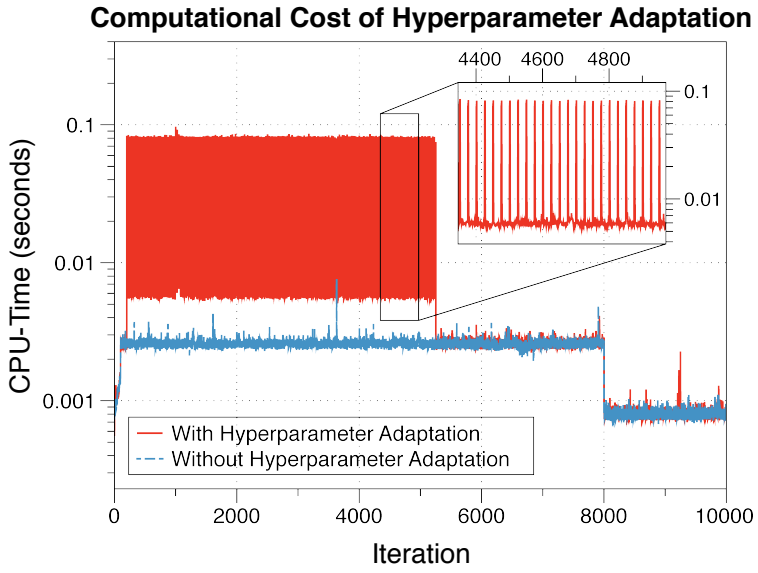


Figure 4.4: Optimisation of the hyperparameters requires gradient computations which results in an increased CPU load. Periodic kernel matrix inversions are also needed leading to the observed spikes after each sampling interval of 25. After convergence (at $t \approx 5000$), the required computational time decreased to match the OIESGP with fixed hyperparameters.

pared to the OIESGP with fixed hyperparameters (Fig. 4.3c); the testing average RMSE and log-likelihood scores over the last 20% of the sequence was 0.045 and 1.305 respectively compared to 0.064 and 0.213 when hyperparameters were fixed.

4.2.2.1 *Computational Cost*

Unfortunately, free lunches are hard to come by and Figure 4.4 shows that these benefits come with additional computational cost. In particular, optimisation of the hyperparameters requires gradient computations on the order of $O(|\theta|s_B^2)$ and periodic kernel matrix inversions of order $O(s_B^3)$, resulting in a pronounced spike after each sampling interval.

This inversion can be avoided by having the OIESGP start "fresh", but leads to slower adaptation and higher errors during the initial learning period. Alternatively, the inversion can be performed on a separate thread on multi-core systems and the out-dated OIESGP replaced thereafter. The kernel matrix derivative computations cannot be avoided but the gradient sampling can be amortised over a set of iterations. For example, we can compute the likelihood gra-

dient for a single parameter per iteration. In this case, we found it necessary to randomise the order of the hyperparameter gradients to prevent unintended correlations[3].

Ultimately, the choice of implementation depends on the application at hand; in situations where fast responses are required but slow adaptation is permissible (e.g., we already have a good sense of the hyperparameters or the underlying system changes slowly), we can sample over longer intervals.

4.3 EMPIRICAL RESULTS ON BENCHMARK PROBLEMS

In this section, we present empirical results comparing OIESGP to the OESGP and other state-of-the-art methods on three classes of benchmarks: one-step prediction, system identification and prediction with irrelevant inputs. We re-use the dynamical systems introduced in the previous chapter but here, we made the benchmarks more difficult by corrupting the observations with additive zero mean observation noise with sd. 0.05. Along with the NP2 system identification problem, we also considered the NARMA-10 benchmark.

For comparison, we included in this study the SOGP, LWPR, Kernel Least Mean Squares (KLMS) [Liu et al., 2008] and Naive Online Regularized Risk Minimization Algorithm (NORMA) [Kivinen et al., 2004]. In brief, both KLMS and NORMA are kernel-based algorithms that employ stochastic gradient descent (otherwise known as the least mean squares algorithm) to minimise an empirical risk cost function (NORMA minimises the *regularised risk*). These methods were trained online and use sliding-windows as "memory". For these experiments, we used the lwpr library [Vijayakumar et al., 2012] and the KAFBox MATLAB toolkit [Van Vaerenbergh, 2012].

For each of the above methods, we performed a grid-search over their respective parameters (given in Appendix C) to minimise the RMSE over the first 5000 time-steps (80% training, 20% testing). The kernel based methods used the standard SE kernel. For the SOGP, we also included the full ARD kernel (labelled

3 The Matlab OTL software toolbox produced with this thesis includes this functionality.

as SOGP_{ARD}); this differs from the recursive kernel used by OIESGP since there lengthscale parameter for each element in the sliding window. In addition, the GP-based methods (OESGP, OIESGP and SOGP) used online hyper-parameter optimisation as described in the previous section.

4.3.1 *Performance Measurement*

For each of the experiments conducted, we conducted 50 independent runs. Each time-series consisted of 10^5 time-steps where 80% was used for training and the remaining for testing. To compare the methods, we used three commonly employed performance metrics. The first two are the mean normalised absolute error (MNAE), and the root-mean-square prediction error (RMSE), introduced in Chapter 3. For the GPs and LWPR that were capable of producing predictive variances, we also computed the negative log predictive density:

$$\text{NLPD} = T_e^{-1} \sum_{t=t_s}^{T_s} \log p(d_t|\hat{y}_t, \hat{\sigma}_t^2). \tag{4.22}$$

Since we assume Gaussian noise:

$$\log p(d_t|\hat{y}_t, \hat{\sigma}_t^2) = \frac{1}{2}\log \hat{\sigma}_t^2 + \frac{(d_t - \hat{y}_t)^2}{2\hat{\sigma}_t^2} + \frac{1}{2}\log 2\pi. \tag{4.23}$$

where $t_s = 5001$, $T_s = 10000$ and T_e is the length of the testing sequence. Unlike the two aforementioned error scores, the NLPD also includes the predictive variance and penalises methods which are overconfident.

4.3.2 *Accuracy Results*

Let us first consider the noisy prediction problems without any irrelevant inputs. Table 4.1 summarises the three error scores obtains by each of the algorithms and Figure 4.5 provides a visual illustration of the RMSE distributions. Overall, the GP methods outperformed the other algorithms on the benchmarks except

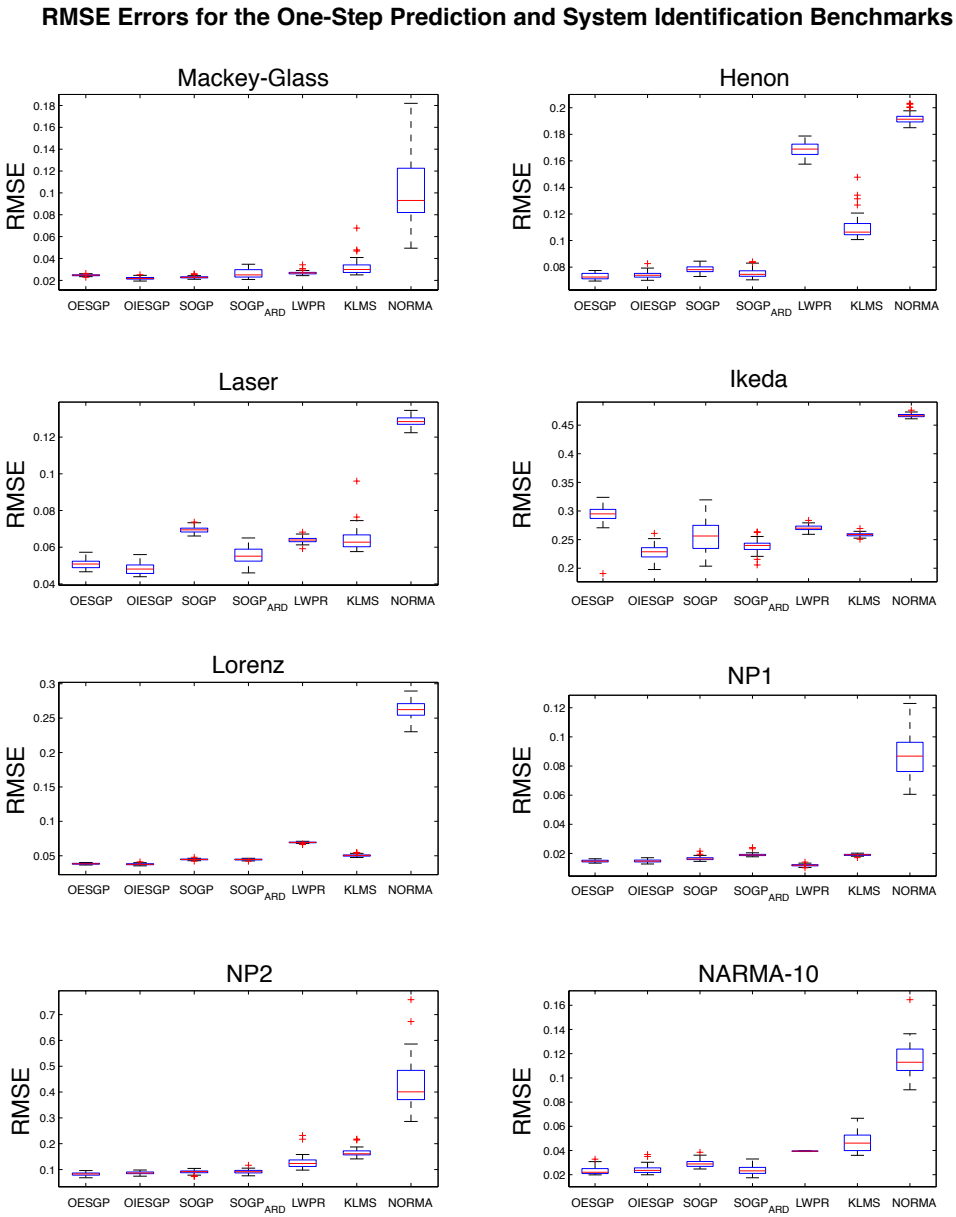


Figure 4.5: RMSE scores for the one-step prediction and system identification benchmarks. As can be seen from the box-plots, the online GPs, particularly the online finite and infinite echo-state GPs, obtain the best scores among the algorithms considered across the problems considered (except the NP1 problem where LWPR obtained the best scores).

Problem	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
Mackey-Glass	0.0904	**0.0802**	0.0839	0.0925	0.0980	0.1109	0.3518
	(0.0027)	(0.0052)	(0.0035)	(0.0248)	(0.0038)	(0.0293)	(0.1593)
	0.0247	**0.0218**	0.0228	0.0250	0.0268	0.0299	0.0931
	(0.0008)	(0.0014)	(0.0010)	(0.0068)	(0.0013)	(0.0069)	(0.0405)
	-1.9255	**-1.9718**	-1.9490	-1.9285	-0.4120	–	–
	(0.0307)	(0.0573)	(0.0231)	(0.1201)	(0.0193)	–	–
Henon	**0.2513**	0.2541	0.2715	0.2577	0.6372	0.3901	0.7147
	(0.0138)	(0.0092)	(0.0149)	(0.0140)	(0.0276)	(0.0276)	(0.0151)
	0.0725	0.0739	0.0781	0.0745	0.1689	0.1063	0.1913
	(0.0041)	(0.0027)	(0.0036)	(0.0041)	(0.0077)	(0.0084)	(0.0042)
	-1.1619	-1.1538	-1.0592	-1.1051	-0.0328	–	–
	(0.0928)	(0.0539)	(0.0660)	(0.0752)	(0.0322)	–	–
Laser	0.1962	**0.1913**	0.2407	0.2173	0.2140	0.2428	0.4820
	(0.0125)	(0.0233)	(0.0063)	(0.0316)	(0.0079)	(0.0542)	(0.0345)
	0.0508	**0.0481**	0.0695	0.0551	0.0639	0.0627	0.1284
	(0.0035)	(0.0046)	(0.0021)	(0.0065)	(0.0018)	(0.0065)	(0.0035)
	-1.5247	**-1.5515**	-1.1924	-1.4904	-0.3206	–	–
	(0.0657)	(0.0634)	(0.0774)	(0.0899)	(0.0148)	–	–
Ikeda	0.5075	**0.3825**	0.4079	0.4002	0.4675	0.4230	0.8099
	(0.0362)	(0.0308)	(0.0791)	(0.0330)	(0.0134)	(0.0081)	(0.0043)
	0.2950	**0.2288**	0.2563	0.2398	0.2701	0.2590	0.4664
	(0.0157)	(0.0162)	(0.0403)	(0.0110)	(0.0061)	(0.0036)	(0.0039)
	0.2183	**0.0642**	2.1283	0.0803	0.4160	–	–
	(0.0494)	(0.0695)	(2.7382)	(0.5110)	(0.0145)	–	–
Lorenz	0.1411	**0.1379**	0.1637	0.1635	0.2596	0.1757	1.0680
	(0.0044)	(0.0070)	(0.0036)	(0.0041)	(0.0055)	(0.0079)	(0.0699)
	0.0383	**0.0375**	0.0445	0.0445	0.0693	0.0505	0.2625
	(0.0012)	(0.0015)	(0.0012)	(0.0013)	(0.0009)	(0.0017)	(0.0168)
	-1.7517	**-1.7577**	-1.6525	-1.6471	-0.2792	–	–
	(0.0265)	(0.0250)	(0.0264)	(0.0333)	(0.0065)	–	–
NP1	0.0157	0.0159	0.0176	0.0198	**0.0123**	0.0202	0.1027
	(0.0011)	(0.0014)	(0.0015)	(0.0010)	(0.0007)	(0.0009)	(0.0224)
	0.0147	0.0147	0.0162	0.0189	**0.0119**	0.0191	0.0868
	(0.0011)	(0.0013)	(0.0013)	(0.0008)	(0.0008)	(0.0007)	(0.0201)
	-1.9897	-2.0139	**-2.0196**	-1.9306	-0.4538	–	–
	(0.0437)	(0.0323)	(0.0384)	(0.0447)	(0.0207)	–	–
NP2	**0.1292**	0.1454	0.1538	0.1525	0.2111	0.2760	0.7477
	(0.0132)	(0.0158)	(0.0142)	(0.0129)	(0.0379)	(0.0347)	(0.3435)
	0.0828	0.0866	0.0919	0.0915	0.1232	0.1610	0.4007
	(0.0083)	(0.0068)	(0.0068)	(0.0097)	(0.0255)	(0.0154)	(0.1131)
	0.9015	0.8984	0.8936	0.8714	**0.8344**	–	–
	(0.0229)	(0.0366)	(0.0235)	(0.0389)	(0.0363)	–	–
NARMA-10	**0.1687**	0.1792	0.2225	0.1784	0.3018	0.3716	0.9501
	(0.0327)	(0.0291)	(0.0304)	(0.0376)	(0.0011)	(0.1200)	(0.1660)
	0.0220	0.0235	0.0289	0.0232	0.0394	0.0461	0.1129
	(0.0041)	(0.0038)	(0.0040)	(0.0049)	(0.0002)	(0.0129)	(0.0177)
	-1.9167	**-1.9463**	-1.7946	-1.8476	-0.4208	–	–
	(0.0499)	(0.1107)	(0.0707)	(0.0747)	(0.0067)	–	–

Table 4.1: Median MNAE (top), RMSE (middle) and NLPD (bottom) scores for the next-step prediction and system identification benchmarks. Interquartile ranges are shown in brackets and lowest scores are in **bold**.

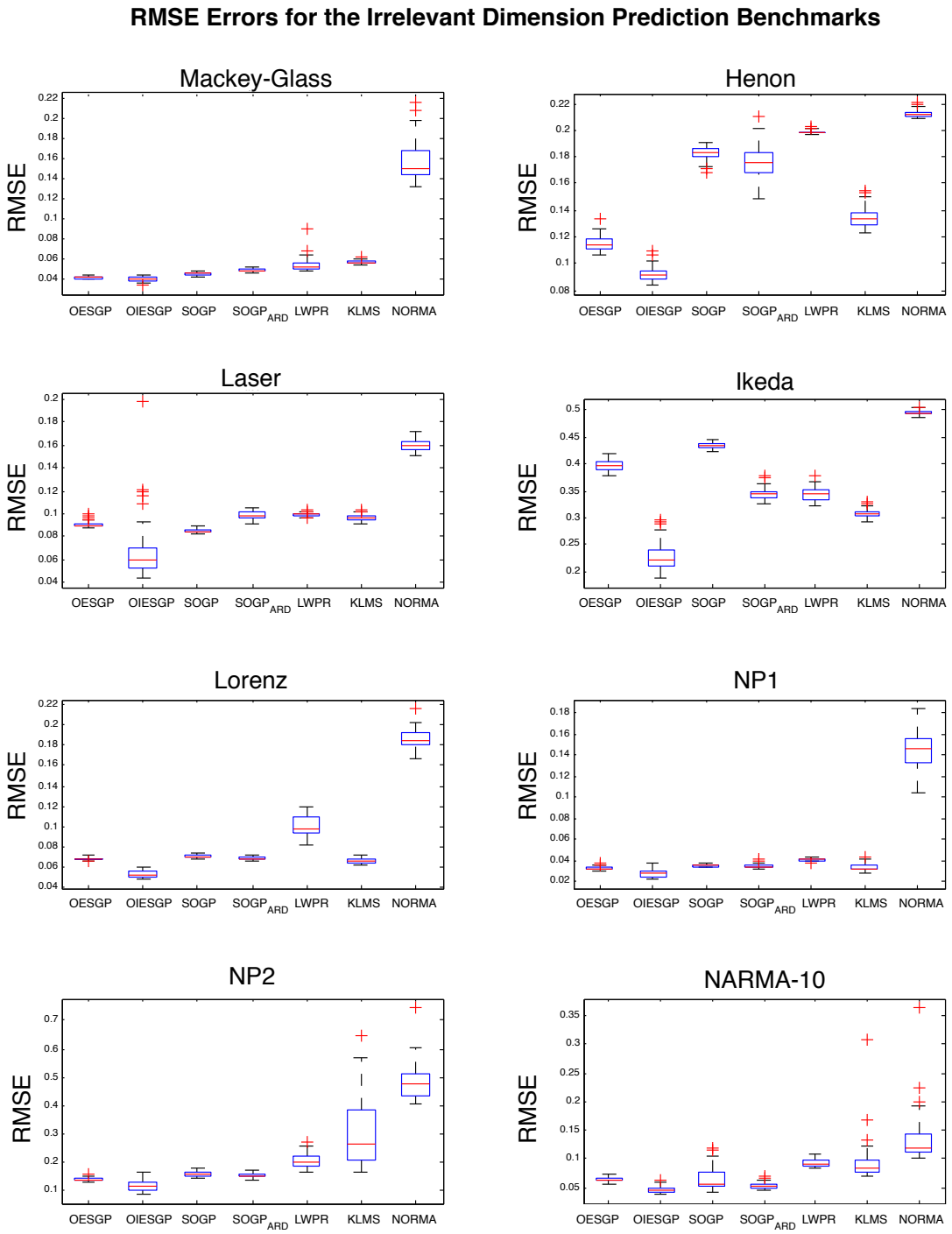


Figure 4.6: RMSE scores for the benchmarks with an irrelevant dimension included with the input. The OIESGP achieves the best scores among all the problems considered.

Problem	OESGP	OIESGP	SOGP	SOGP_{ARD}	LWPR	KLMS	NORMA
Mackey Glass	0.1544	**0.1502**	0.1692	0.1796	0.1994	0.2113	0.5799
	(0.0063)	(0.0080)	(0.0060)	(0.0079)	(0.0233)	(0.0078)	(0.0916)
	0.0416	**0.0404**	0.0456	0.0484	0.0530	0.0567	0.1507
	(0.0016)	(0.0023)	(0.0013)	(0.0020)	(0.0059)	(0.0021)	(0.0248)
	-1.7027	**-1.7200**	-1.6461	-1.5439	-0.1243	–	–
	(0.0275)	(0.0593)	(0.0240)	(0.0349)	(0.0429)	–	–
Henon	0.4105	**0.3354**	0.7009	0.6355	0.7563	0.4843	0.8262
	(0.0267)	(0.0243)	(0.0247)	(0.0630)	(0.0059)	(0.0294)	(0.0204)
	0.1135	**0.0921**	0.1836	0.1763	0.1983	0.1332	0.2125
	(0.0074)	(0.0062)	(0.0062)	(0.0155)	(0.0012)	(0.0086)	(0.0027)
	-0.7256	-0.6513	-0.2345	4.9408	0.3826	–	–
	(0.1185)	(0.1817)	(0.0340)	(1.8585)	(0.1984)	–	–
Laser	0.3375	**0.2170**	0.3189	0.3898	0.3714	0.3718	0.7061
	(0.0166)	(0.0570)	(0.0110)	(0.0273)	(0.0094)	(0.0178)	(0.0476)
	0.0899	**0.0603**	0.0851	0.0979	0.0994	0.0957	0.1592
	(0.0022)	(0.0171)	(0.0029)	(0.0052)	(0.0012)	(0.0027)	(0.0067)
	-0.9387	**-1.2749**	-0.9717	-0.8811	0.0707	–	–
	(0.1767)	(0.2470)	(0.1047)	(0.0482)	(0.0324)	–	–
Ikeda	0.7059	**0.3798**	0.7834	0.6019	0.6042	0.5092	0.8646
	(0.0262)	(0.0501)	(0.0156)	(0.0319)	(0.0318)	(0.0189)	(0.0066)
	0.3974	**0.2218**	0.4341	0.3434	0.3434	0.3066	0.4942
	(0.0136)	(0.0269)	(0.0089)	(0.0132)	(0.0163)	(0.0094)	(0.0053)
	0.5561	**0.1333**	0.7198	0.3600	0.5631	–	–
	(0.0700)	(0.3028)	(0.0444)	(0.0379)	(0.0282)	–	–
Lorenz	0.2558	**0.1955**	0.2637	0.2590	0.3717	0.2435	0.7040
	(0.0063)	(0.0184)	(0.0071)	(0.0059)	(0.0597)	(0.0134)	(0.0528)
	0.0684	**0.0527**	0.0704	0.0689	0.0974	0.0660	0.1849
	(0.0014)	(0.0044)	(0.0016)	(0.0016)	(0.0149)	(0.0031)	(0.0116)
	-1.2591	**-1.4993**	-1.2198	-1.2418	0.0027	–	–
	(0.0267)	(0.0800)	(0.0678)	(0.0238)	(0.0626)	–	–
NP1	0.0359	**0.0302**	0.0384	0.0377	0.0459	0.0355	0.1814
	(0.0019)	(0.0060)	(0.0016)	(0.0017)	(0.0019)	(0.0050)	(0.0269)
	0.0326	**0.0275**	0.0353	0.0346	0.0408	0.0327	0.1470
	(0.0018)	(0.0054)	(0.0014)	(0.0015)	(0.0014)	(0.0042)	(0.0236)
	-1.7028	**-1.7940**	-1.7728	-1.7704	-0.4290	–	–
	(0.1119)	(0.1287)	(0.0317)	(0.0423)	(0.0316)	–	–
NP2	0.2425	**0.1989**	0.2684	0.2619	0.3529	0.4932	0.8066
	(0.0140)	(0.0638)	(0.0097)	(0.0126)	(0.0553)	(0.4403)	(0.1222)
	0.1389	**0.1121**	0.1565	0.1543	0.2030	0.2638	0.4746
	(0.0075)	(0.0301)	(0.0109)	(0.0104)	(0.0290)	(0.1762)	(0.0808)
	0.9142	**0.8650**	0.9230	0.9001	0.9245	–	–
	(0.0143)	(0.0870)	(0.0171)	(0.0182)	(0.0331)	–	–
NARMA-10	0.4631	**0.3393**	0.4112	0.3768	0.6647	0.5917	0.8807
	(0.0272)	(0.0326)	(0.1633)	(0.0329)	(0.0236)	(0.1586)	(0.2052)
	0.0633	**0.0463**	0.0561	0.0521	0.0910	0.0843	0.1194
	(0.0051)	(0.0062)	(0.0252)	(0.0058)	(0.0097)	(0.0206)	(0.0325)
	-1.3372	**-1.6267**	-1.3080	-1.5275	-0.0381	–	–
	(0.0857)	(0.1175)	(0.7161)	(0.1322)	(0.0339)	–	–

Table 4.2: Median MNAE (top), RMSE (middle) and NLPD (bottom) scores for benchmarks with an irrelevant dimension. Interquartile ranges are shown in brackets and lowest scores are in **bold**.

for the NP1 problem where LWPR obtains the lowest MNAE and RMSE error scores.

Among the online GPs, OIESGP obtained the best scores on a majority of the benchmarks and never performed worse than the OESGP on all the error scores. Interestingly, the simple SOGP with the isotropic Gaussian kernel also performed well on the Mackey-Glass, Ikeda, NP2 and NARMA-10 benchmarks, echoing recent results [Rodan and Tino, 2011; Strauss et al., 2012] that complex reservoirs are not necessary for certain prediction tasks. We had expected the SOGP with the full ARD kernel to realise lower errors than the isotropic kernel because the additional lengthscale parameters would allow a better "fit" to the given problems. However, this only occured for the Henon, Laser and Ikeda benchmarks. This discrepancy may be due to the fact that the full ARD kernel required an optimisation over a relatively large parameter space and would settle into local minima (or not converge within the maximum number of iterations set).

The full benefit of the OIESGP with the recursive ARD kernel is evidenced by its performance on the problems with irrelevant inputs (Table 4.2 and Fig. 4.6); the OIESGP outperforms all the other methods, including the OESGP, in terms of the error scores on all the problems; results are statistically significant at the Bonferroni-corrected $\alpha = 0.05$ level (see Appendix D for p-value tables).

4.3.3 *Computational Costs*

Comparing computational costs, LWPR remains the overall fastest algorithm in terms of CPU time — average training and testing times are shown in Appendix D, Tables D.5 and D.6. Among the online GP methods, the isotropic Gaussian was the most rapid kernel, as expected. Contrasting OIESGP with OESGP, although training was more expensive on the benchmarks with irrelevant dimensions (due to the longer optimisation periods), the OIESGP produced predictions 20% faster on all the benchmarks except NP1 and NARMA-10 with irrelevant dimensions.

4.4 DISCUSSION: A DYNAMICAL SYSTEMS PERSPECTIVE

There are two notable findings obtained from our experiments. First, the isotropic Gaussian produces results comparable to more complex methods on the one-step prediction task with dynamical systems. Second, the OIESGP was decidedly more accurate than the other methods on the problems with irrelevant dimensions. In this section, we try to understand the reasons for these observations from the perspective of dynamical systems theory.

A hallmark result in the field of non-linear dynamical systems is Taken's Embedding Theorem [Takens, 1981], which states that the delay vector constructed from single observations is an embedding — a one-to-one mapping between manifolds — whenever the length of the embedding is larger than twice the intrinsic dimensionality of original space. Informally, a sufficiently long delay vector (sliding-window) reconstructs the state-space such that topological properties are preserved. Although Taken's original statement assumed infinite precision observations with no noise, this result has been extended to systems with observation and dynamical noise [Casdagli et al., 1991; Stark et al., 2003].

As a concrete example, let us consider a simple linear dynamical system consisting of a rotating two-dimensional state $\mathbf{s}_t$ by a fixed angle θ via iterative application of $\mathbf{T}$ to an initial state vector $\mathbf{s}_0$.

$$\mathbf{s}_{t+1} = \mathbf{T}\mathbf{s}_t \tag{4.24}$$

$$\mathbf{y}_t = \mathbf{O}\mathbf{s}_t + \epsilon_y \tag{4.25}$$

where

$$\mathbf{T} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad \mathbf{O} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

and $\theta = 0.1, \mathbf{s}_0 = [0, 1]^\top, \epsilon_y \sim \mathcal{N}(0, 0.1)$. In Figure 4.7b, we see that standard regression on the observations will fail to produce accurate predictions since the distribution is generally *bi-modal*; as a partial observation, y_t only contains

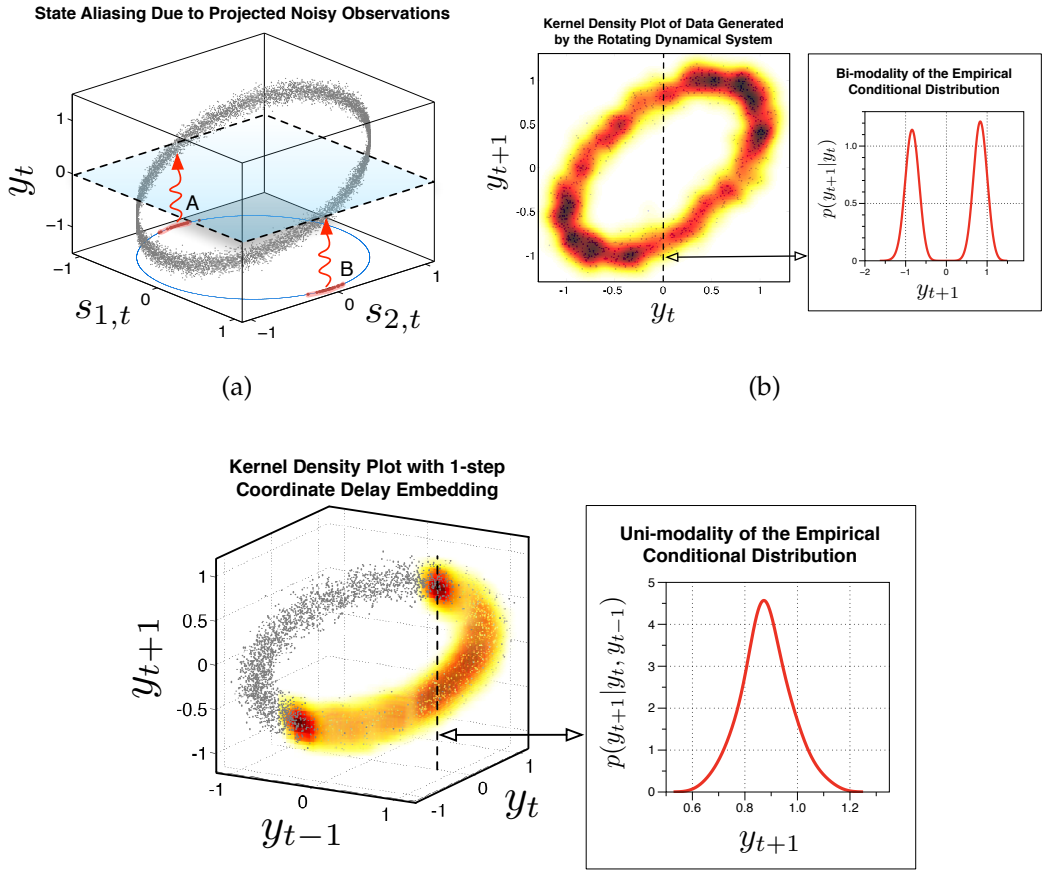


Figure 4.7: Fig. 4.7a shows that the true states live on a unit circle. However, the probability of the next observation conditioned upon the current one is bi-modal, violating assumptions made by typical regression methods (Fig. 4.7b). This bi-modality occurs because of state aliasing caused by partial observations. In 4.7c, a coordinate delay embedding reconstructs the state space sufficiently well that uni-modality is recovered and predictions can be performed accurately.

limited information about the underlying state. As Fig. 4.7a shows, contaminated states lead to *state aliasing*; two distinct state sets A and B can generate observations $y_t \approx 0$ (illustrated by the plane). This phenomena undermines predictability of the system.

However, a history of the observed time-series — the embedding — preserves important properties such as the system dynamics. In this particular example, a simple one-step delay embedding reconstructs the state space sufficiently well such that uni-modality of conditional probability is recovered (See Fig. 4.7c).

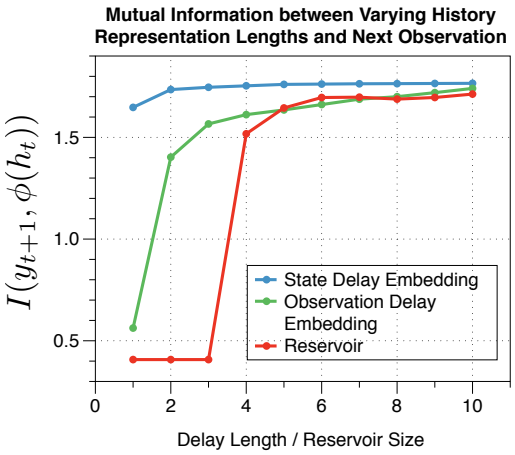


Figure 4.8: The mutual information between the next observation (to be predicted) and the delay-embedding grows with the delay length approaching that of the underlying state. As we would expect, embedding the underlying state does not increase the mutual information. The reservoir shows a different initial profile; a increase from 3 to 4 neurons results a sudden jump in informativeness.

Taken's Theorem explains why, for the basic prediction problems, the isotropic Gaussian with a sliding window works well. Using a sufficiently long window permits the GP to operate in a space topologically similar to the original state space. Indeed, a time-delay window is not the only reconstruction or representation we have considered; the reservoirs used in echo-state networks and the OESGP can be viewed as reconstructions. Computing the mutual information (MI) between the next observation and the delay-coordinate and reservoir embeddings (Fig. 4.8) allows us to quantitatively compare the informativeness of the different representations. For our sample problem, the delay-embedding and reservoirs have different MI profiles as the size of the representation is increased. While the increase is more gradual for the delay vector, the reservoir requires a minimum size — a "tipping point" — before sufficient information is available.

4.4.1 *The Problem of False Neighbours*

As to the second question of why the introduction of an irrelevant dimension hampered the algorithms except the OIESGP, we consider the issue of *false neigh-*

bours [Kantz and Schreiber, 2003], used in state-space reconstruction to find the optimal embedding dimension.

The idea itself is straightforward: as our simple rotating system demonstrates, partial observations with noise project states from their true high-dimensional spaces into neighbourhoods that they do not truly belong to. As such, these projected points have neighbours that may be from other regions of the space that are not truly similar. Irrelevant dimensions exacerbates the problem of false neighbours because the "extra" dimensions factor into the kernel and distort distances.

This has consequences for GPs (and other non-parametric methods) that rely on the notion of similarity between the inputs as represented by the kernel function. Since the computed covariances are contaminated by the irrelevant dimensions, prediction accuracies decrease. As such, we can interpret SNGD using the recursive ARD kernel as a "moulding" of the similarities (via the hyperparameters) using the predictability of the system to approximate the topological properties of the original space. Although the full ARD kernel with sufficiently long window offers this feature, the parameter space employed is far larger. In a practical sense, the OIESGP strikes a balance between having too many parameters (the full ARD and reservoir) and having too few (the isotropic Gaussian).

4.5 SUMMARY AND CONCLUSION

In this chapter, we investigated an extension of the OESGP in two important ways: first, we obviated the need to create and maintain a reservoir by using an "infinite" reservoir in the form of a recursive kernel with ARD. We then showed how the hyperparameters of this kernel can be learned using SNGD on the generalised error likelihood. Extensive experiments on benchmark problems demonstrated the superiority of the OIESGP over using finite reservoirs, the isotropic and full ARD kernels, as well as state-of-the-art methods. In the next part of this thesis, we will explore how the OIESGP can be used as a base model in two challenging learning problems in robotics.

Part III

APPLICATIONS IN ROBOTICS

In the next two chapters, we use the OIESGP as a "building block" in higher level structures applied to robotic learning problems. In Chapter 5, we develop classifiers for learning objects by touch on the iCub humanoid platform. Unlike existing methods, our classifiers learn online: they iteratively refine existing models and create new models when necessary. In Chapter 6, we propose a novel approach to building assistive controllers via learning by demonstration. Using the OIESGP, we propose a probabilistic model for Learning Assistance by Demonstration (LAD) and demonstrate its effectiveness with experiments on the ARTY robotic smart wheelchair using paired haptic controllers.

5

ITERATIVELY LEARNING OBJECTS BY TOUCH

Our sense of touch is considered our most pervasive sense and is exquisitely sensitive; a large number of nerve receptors distributed across our body allows us to finely examine our environment. Touch sensing allows us to detect, identify and manipulate objects not within our field-of-view; we can easily feel to locate and grasp a pen on a desk behind us while we are attending to another visual-based task (e.g., reviewing a thesis). In addition, objects that have similar appearance (e.g., ripe and unripe fruits) are more easily distinguished using tactile feedback.

The development of "artificial skin" — tactile sensors created out of flexible semiconducting materials — may provide future robots with similar (or height-

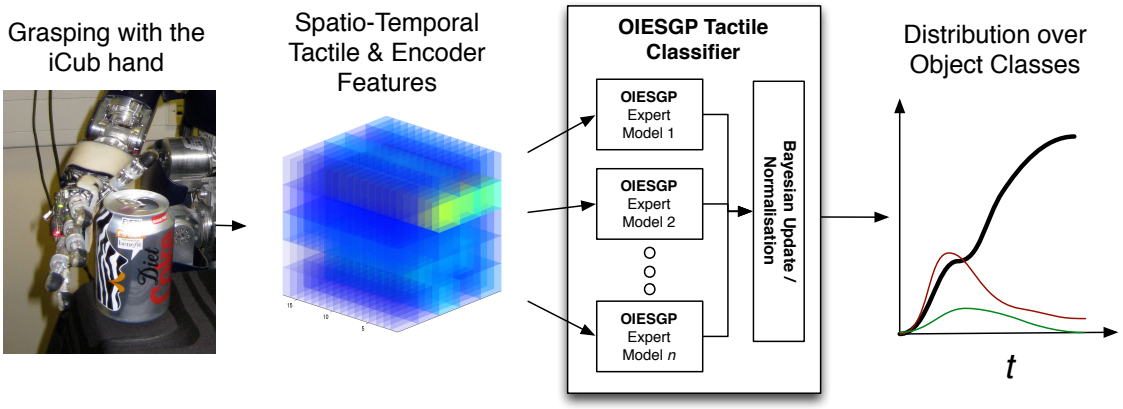


Figure 5.1: Online Tactile Classification using OIESGP Online Learning Experts. Our method works directly on temporal data (without the need for an extensive feature vector) and is capable of creating new spatio-temporal experts "on-the-fly" as new objects are encountered and refining models of familiar objects. A probability distribution over object classes is maintained and updated throughout the grasping action.

ened) touch-sensing capabilities. An interesting and challenging area of research is to learn to recognise objects through tactile feedback. For example, recent studies have explored tactile classification using the C4.5 decision tree [Chitta et al., 2010, 2011] and a bag-of-words model [Schneider et al., 2009]. Such classifications can inform subsequent manipulations, for example, moving the fingers to obtain a better grasp position.

In this chapter, we develop both generative and discriminative OIESGP-based tactile classifiers on the iCub humanoid platform [Soh et al., 2012]. Similar to recent work (e.g., [Schneider et al., 2009; Chitta et al., 2011]), we classified objects using tactile feedback during a grasping or palpation motion (without lifting the object). However, a primary difference is that our systems learn online, i.e., they continuously improve internal models of familiar things and create models for novel objects "on-the-fly"; this approach is more applicable to robots operating in the real-world where both novel and familiar items would be frequently encountered. Compared to existing tactile classifiers — such as C4.5 decision trees [Chitta et al., 2010, 2011] and neural networks [Schöpfer et al., 2007] — our models do not require the construction of an extensive feature vector (such as the bag-of-features approach used in [Schneider et al., 2009] and [Pezzementi et al., 2011]) and provide a probability distributions over object classes. Probability estimates allow systems to better decide when to halt the grasping/palpation motion (e.g., when a given threshold is breached) or to "pass" on uncertain classifications.

The remainder of this chapter is organised as follows: in Section 5.1, we first give an overview of related work. Section 5.2 presents our tactile classifiers; in particular, generative and discriminative classifiers using OIESGP sub-models. Section 5.3 describes our experimental setup, specifically, the tactile sensors and grasp controller used in our experiments and the test objects used. The performance of the classifiers are reported in Section 5.4 along with comparisons to offline classifiers and human subjects. Finally, Section 5.5 concludes this chapter with a summary and avenues for future research.

5.1 RELATED WORK ON TACTILE-BASED CLASSIFICATION

In this section, we give an overview of related work in object classification using data obtained from tactile sensors. The development of the tactile sensors themselves constitutes a major research field and we refer interested readers to recent reviews [Tegin and Wikander, 2005; Yousef et al., 2011]. Our focus here is on the machine learning (classification) techniques applied along with the features used to determine the identity of handled objects. To organise our discussion, we group the reviewed works into *discriminative* or *generative* methods.

5.1.1 *Discriminative Tactile Classifiers*

Given observation (or features) $\mathbf{o} \in \mathbb{R}^d$, we can directly model the conditional distribution $p(c_i|\mathbf{o})$ over object class $c_i \in \mathcal{C}$, which yields a *discriminative* classifier. For example, Schöpfer, Ritter and Heidemann [Schöpfer et al., 2007] used a local linear map (LLM) neural network trained on features (dimensionally-reduced using principal components analysis) obtained using a low-cost 16×16 tactile sensor array mounted on a Unimation 6-DOF PUMA 200 robot arm. More recent work by Schöpfer [Schöpfer et al., 2009] used a C4.5 decision tree trained on 51 tactile features (e.g., maximum taxel value), which achieved 92.5% accuracy score classifying 16 different objects.

The C4.5 tree was also used by Chitta *et al.* [Chitta et al., 2010, 2011] to estimate the internal state of bottles with varying fullness grasped using the PR2 gripper equipped with a capacitive sensor with 22 cells. Different from Schöpfer *et al.*, the classifier used the frequency response of the manipulated objects, achieving an accuracy score of 93.9% in recognising the different liquid containers.

Opting for a nearest neighbour approach, Drimus *et al.* [Drimus et al., 2011] performed classification of 10 objects (both rigid and deformable objects) using the k-Nearest Neighbour (k-NN) algorithm. Each object was represented by a variable length vector composed of the first two moments of each "tactile frame". Distances between the two vectors were computed using Dynamic Time

Warping (DTW) and the closest k neighbours voted on the object's class. The best accuracy obtained using this method on the 10 different objects (with 10 repeated grasps per object) was 92%.

5.1.2 *Generative Tactile Classifiers*

In contrast to the discriminative approach, the *generative* approach models the joint $p(\mathbf{o}, c_i) = p(\mathbf{o}|c_i)p(c_i)$ and through application of Bayes rule, obtains the class distribution $p(c_i|\mathbf{o}) \propto p(\mathbf{o}|c_i)p(c_i)$. In general, discriminative models tend to have lower (asymptotic) errors, whilst generative models may more quickly reach their lower bound [Ng and Jordan, 2001]. The primary task in generative modelling is in creating the distribution $p(\mathbf{x}|c_i)$ since $p(c_i)$ can usually be estimated easily from the obtained data or assumed uniformly distributed. Work by Meier *et al.* [Meier et al., 2011] demonstrated that compact 3-D representations of unknown objects could be obtained by tactile sensors. In particular, Kalman filters were used to build a probabilistic point cloud (stored as a k-dimensional-tree). This representation can then be used to classify objects via Bayes rule above.

Instead of operating directly on the raw tactile data, Schneider *et al.* [Schneider et al., 2009] extracted a bag-of-features from low-resolution tactile intensity images. A vocabulary of codewords was then obtained using unsupervised k-means to cluster the obtained images. In this work, distances were measured using a linear weighted combination of the pixel-by-pixel Euclidean distance (with vertical shift correction) and the width of the fingers. For each object, a histogram of the codewords was generated and classified using a Bayes classifier. This method achieved a recognition rate of 84.6% with 10 test grasps on each of the 16 objects tested.

Schneider's work was recently extended by Pezzementi *et al.* [Pezzementi et al., 2011] to incorporate descriptors from the image processing community and to use Gaussian Mixture Models (GMM), instead of k-means, for codebook creation. This improved method was capable of handling varying object poses

and achieved higher accuracies exceeding 95%. Another bag-of-words model was proposed by Gorges *et al.* [Gorges et al., 2010] which used an anthropomorphic robot hand. Classifications were performed using a combination of tactile and finger configurations (so called haptic key features), which were dimensionally reduced using PCA. Instead of using k-means or GMM, the codebook was obtained using the Self-Organising Map (SOM). The average accuracy of this approach was 75% (individual object accuracies varied greatly, ranging from 50% to 100%).

Whether discriminative or generative, we see that machine-learning classifiers generally achieved good performance on tactile feedback, reflecting the informativeness of the signals. However, one common attribute of the aforementioned methods is that they are inherently *offline*; the grasp or palpation has to be completed before a classification can be performed. In contrast, humans are typically able to determine the type or properties of handled objects *during* the manipulation process. In the next section, we describe our online classifiers that are similarly capable. Moreover, our models are iteratively trained — they can be incrementally updated with each observed sample.

5.2 OIESGP GENERATIVE AND DISCRIMINATIVE TACTILE CLASSIFIERS

In this study, we developed both generative and discriminative classifiers composed of OIESGP sub-models, $m_i \in \mathcal{M}$, representing each object class c_i. At each time-step t, the classifiers are provided with an observation $\mathbf{o}_t$ and are expected to provide a probability $p(c_i)$ for each class.

5.2.1 *Discriminative Tactile Classifier*

Recall that in the discriminative approach, we model the conditional distribution $p(c_i|\mathbf{o})$ directly. Here, we have used one-versus-all classifier scheme whereby one classifier is trained for each object.

Unlike regression, the target values $h_{i,t} \in \{-1, +1\}$ are positive for the correct class and negative otherwise, rendering the Gaussian likelihood inappropriate. As such, we applied the probit likelihood [Neal, 1997; Csató, 2002]. Each observation in a given sequence is associated with the appropriate target, which yields the following likelihood function:

$$q_t(c_i|\mathbf{o}_t) = \Phi\left(\frac{h_{i,t}\mu_{i,t}}{\sigma_{i,t}}\right) \tag{5.1}$$

where

$$\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(z|0,1)\,dz \tag{5.2}$$

is a cumulative density function of a standard normal distribution, $\mu_{i,t}$ and $\sigma_{i,t}$ are the predicted mean and standard deviations using model m_i. A normalised probability class distribution at a given time t can be obtained via:

$$p_t(c_i|\mathbf{o}_t) = \frac{q_t(c_i|\mathbf{o}_t)}{\sum_j^{|\mathcal{M}|} q_t(c_j|\mathbf{o}_t)}. \tag{5.3}$$

To train a given model m_i with a positive (correct) sequence, we perform updates to the GP as described in the previous section. However, negative samples present a problem when dealing with temporal sequences in a multi-class setting; the same observation may be generated from different classes and the number of classes imply a positive-negative imbalance. We experimented with four different training schemes for dealing with negative samples: providing negative samples to all models, none of the models (only positive training), and the closest (as measured by probability) or wrongly selected model. Positive training yielded the best scores and as such, we focus on the results obtained via this scheme for the remainder of this paper.

5.2.2 *Generative Tactile Classifier*

Recall that for generative classification, we model the joint probability distribution. Since our data is inherently temporal in nature, we model $p_t(\mathbf{o}_t|c_i)$ and $p_t(c_i)$ where the latter represents the current belief over the classes. Our approach is very similar to the generative classifier presented in Chapter 3. Inference is performed using a Bayes filter to update a probability distribution over the model classes as observations arrive:

$$p_t(c_i|\mathbf{o}, \mathcal{M}) = \frac{p_t(\mathbf{o}_t|m_i)p_{t-1}(c_i)}{\sum_j p(\mathbf{o}_t|m_j)p_{t-1}(c_j)} \tag{5.4}$$

where we use a standard (independent) observational model:

$$p(\mathbf{o}_t|m_i) = \prod_k N(o_{k,t}|\mu_{i,k,t}, \sigma_{i,k,t}). \tag{5.5}$$

Each OIESGP sub-model (for each feature) is the familiar GP regressor we have applied in the previous chapters and poses no difficulty; training the generative classifier simply corresponds to providing the correct model with example sequences. To reduce computational and storage costs, we use the same basis vector set for all sub-models for a given class. However, if the features are uncorrelated or have varying time-scales, using identical BV sets for each output may yield low accuracies. As such, our implicit assumption in this setup is that the observed tactile signals are correlated and have low *intrinsic* dimensionality.

When the grasping motion is initiated, the initial prior is set to be uniformly distributed across the object classes and the class distribution is continuously updated as sensory data is received. To compare accuracies with the other methods, the class with the highest probability (i.e., with the model that predicted the sensory input best over time) at the end of the grasping action is selected as the recognised object.

5.3 EXPERIMENTAL SETUP

The online-learning experiment was conducted in two phases: in the first phase, a sample from each object class was presented to the classifier once. Strictly speaking, this phase was not necessary since new models can be created as the trials progressed but this simplified comparison with offline methods and the human subjects who were informed of the objects were in-advance.

In the second phase, an object was chosen randomly (stratified sampling) and presented for classification. After making a prediction, the algorithm was trained with the actual class label; in other words, the algorithm iteratively learned from observed samples. Training and testing was conducted for the first 80% of the data, and only testing for the final 20%. In the following sub-sections, we give specifics on the test objects, grasping procedure and classifier initialisation parameters used in our experiment.

5.3.1 *Test Objects*

In this experiment, we used nine different everyday objects and one baseline where the grasp was performed with no object, totalling ten classes (shown in Fig. 5.2). These objects comprise plastic bottles with differing amounts of liquid, two soft toys, soda-cans (also of varying fullness), a bottle of lotion and a hardcover book. For each class, we created a dataset of twenty samples using our grasp controller and recorded data for the pressing portion of the grasp, yielding a total of 200 samples. As can be seen in Fig. 5.3, each grasped object generates a distinctive spatio-temporal "signature". Our classifiers are based on the notion that each of these signatures can be learned and represented by an "expert model" (Fig. 5.1). To encourage reproducibility and development of the field, we have made this dataset freely available online [Soh, 2012a].

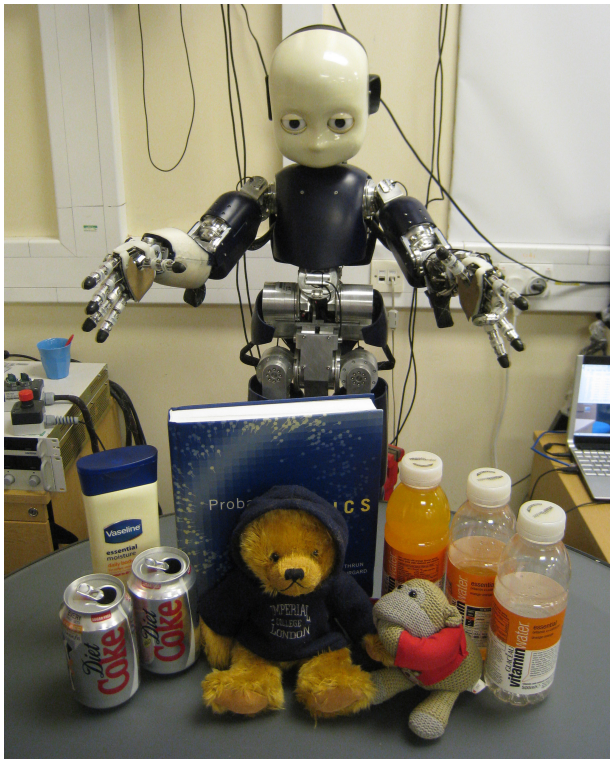


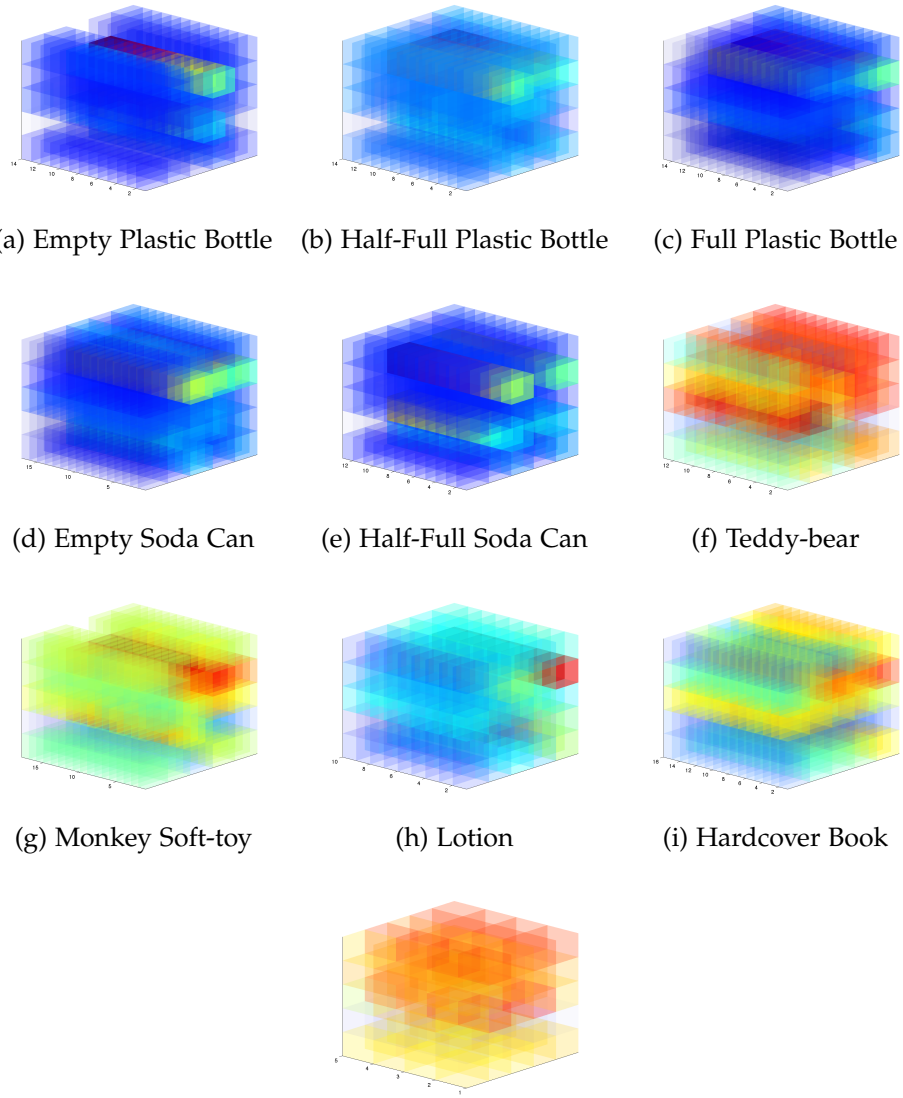
Figure 5.2: iCub with Objects: Plastic bottles (full, half-full, empty), Soda cans (empty, half-full), Teddy-bear, Monkey soft-toy, Lotion bottle and Book.

5.3.2 *Grasping with the iCub Hand*

Previous work on tactile classification has focussed mainly on using grippers (e.g., the two-fingered gripper on PR2 robot, [Chitta et al., 2010]) and industrial-style robotic arms [Schöpfer et al., 2007; Schneider et al., 2009]. For this study, we used the iCub humanoid robotic platform (Fig. 5.2) which has two anthropomorphic dexterous hands with 5 fingers (20 joints with 9 degrees of freedom as shown in Figure 5.4). Each fingertip is wrapped with 12 capacitive pressure sensors under a layer of soft silicone foam. When pressure is applied to the fingertips, the silicone foam is compressed, changing the capacitance. An embedded-board samples all the sensors sequentially, generating output values ranging from 0 to 255. In this work, the data capture rate was 10Hz.

We devised a grasp controller[1] that executed a three-step action: first, it fixed the hand position and orientation and then fully opened the hand digits, yielding the pre-shape (as shown in Fig. 5.5a). After we placed an object in a gras-

1 The grasp controller was implemented by Yanyu Su.



(a) Empty Plastic Bottle (b) Half-Full Plastic Bottle (c) Full Plastic Bottle

(d) Empty Soda Can (e) Half-Full Soda Can (f) Teddy-bear

(g) Monkey Soft-toy (h) Lotion (i) Hardcover Book

(j) None

Figure 5.3: Sample tactile sensor spatio-temporal blocks for the each of the 10 classes where each object generates an individual "signature". Each vertical 5x5 slice represents the 25 tactile features (mean, standard deviation, skewness, maximum and minimum for each finger). Note that these the features change across time (horizontally-stacked slices) as the fingers press on each object.

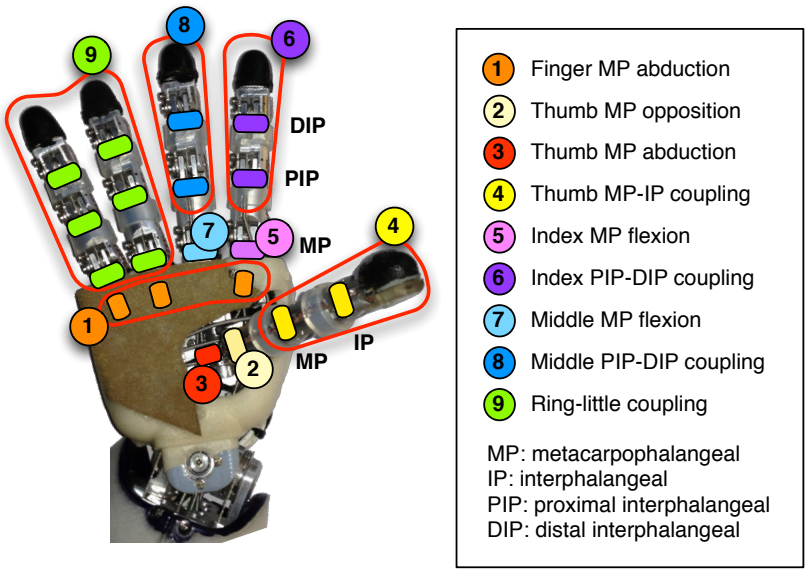
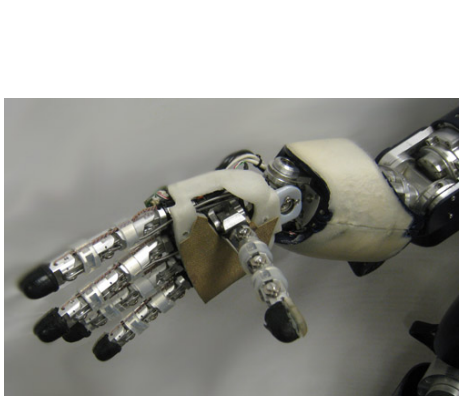
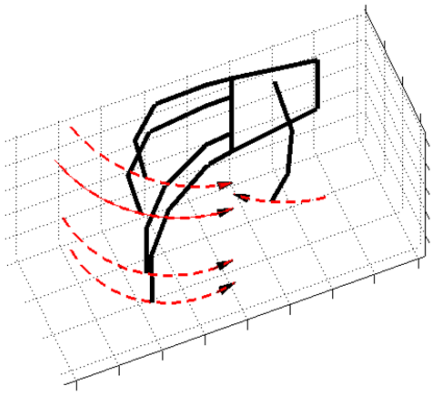


Figure 5.4: The iCub Hand is an anthropomorphic hand with 19 joints and 9 DoFs; this induces a coupling between certain joints as indicated by the colour codes and labelings. In our experiments, we used the tactile (capacitive) sensors on each fingertip. Each fingertip incorporates a flexible printed circuit board underneath a electrically conductive silicone and provides 12 pressure measurements.



(a) Grasping Pre-shape



(b) Grasping Motion

Figure 5.5: The Grasping controller used in this study would first initialise the iCub hand into a "pre-shape" as shown in Fig. 5.5a. It would then close the fingers onto the object via the motion depicted in Fig. 5.5b, pressing onto the object to obtain tactile sensory input.

pable position, the controller closed the digits with low velocity (20 deg/s for each degree of freedom) along a pre-defined trajectory (illustrated in Fig. 5.5b) until the tactile sensor readings breached a critical threshold (or the trajectory completed). In our trials, we found it necessary to vary this threshold from 15-20 to compensate for sensor drift. Finally, the controller would press on the object by moving each digit further along the same trajectory but with a higher velocity (40 deg/s). This continued until the motion was blocked or the trajectory motion was finished. Motion blocks were detected by checking that motor encoder values remained unchanged for 0.5 seconds.

5.3.3 *Classifier Set-up*

Our classifiers process the encoder (measuring joint angles for each of the 9 DOFs) and tactile sensor data from each finger. Instead of using all twelve sensors directly, the sensor data was reduced by computing the first three moments (mean, standard deviation and skewness) as well as the maximum and minimum reading for the twelve sensors on *each* finger; note that this data reduction is "spatial" and not temporal as we do not compute statistics across the time-steps. This resulted in an observation/feature vector $\mathbf{o}_t \in \mathbb{R}^{34}$ (25 tactile features and 9 encoder values) for each time step t. We also experimented with reduced data streams with only the tactile or encoder features.

Both methods were initialised with single model after the first object was encountered and a model class was created whenever a new object is taught to the classifier. In other words, the system "grows" new OIESGP models as needed. Each model was set to use 50 basis vectors with initial lengthscales $\mathbf{l} = [10.0]_{i=1}^{D}$ for the generative model and $\mathbf{l} = [1.0]_{i=1}^{D}$ for the discriminative model. The remaining hyper-parameters initialised with $\sigma_f = 1.0$, $\sigma_n = 0.1$, $\tau = 15$ and $\rho = 1.01$ for both classifiers. Hyper-parameter adaptation was enabled with the sampling interval s_g set at 30 and stopping criteria $\|\triangle\boldsymbol{\theta}\|^2 < c_g = 10^{-4}$ over 10 iterations.

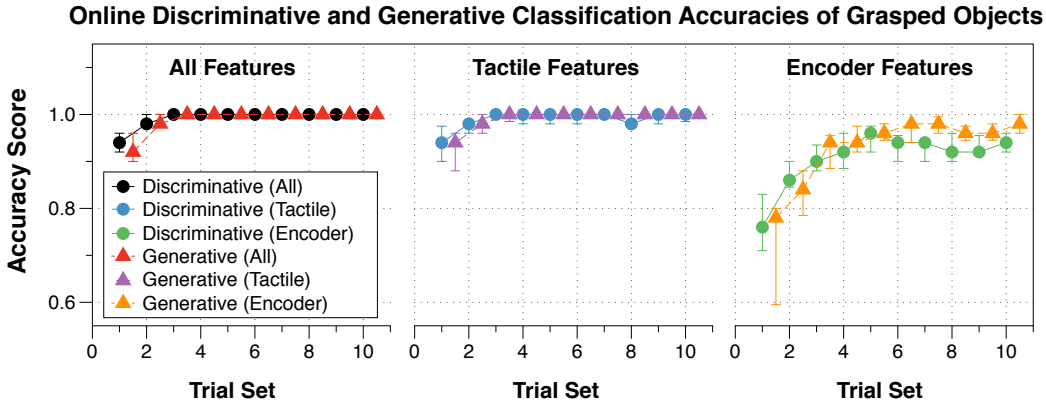


Figure 5.6: Classification Accuracy as Trials Progressed. Each trial set shown consists of 19 objects since we excluded Phase 1. Both the discriminative and generative models improved classification performance as the trials progressed. The best results were obtained when both tactile and encoder features were used; by the fourth trial set, both methods achieved perfect accuracy. However, using only the tactile features also resulted in strong performance, particularly with the generative classifier. The models trained only on the encoder features attained comparatively poor performance.

5.4 EMPIRICAL RESULTS

The results of our experiment (average score across 50 repeated tests with the order of presented samples shuffled) are summarised in Tbl. 5.1 and Fig. 5.6. As Fig. 5.6 shows, both online classifiers performed similarly, improving with more samples seen, as indicated by the higher accuracies and smaller standard deviations as learning progressed.

The best accuracies were obtained when both the tactile and joint angle features were used — both discriminative and generative classifiers attained perfect accuracy on the testing portion (final 20%) of the dataset. Of the two sets of attributes, a significantly higher score was achieved when using only the tactile rather than the encoder features, particularly for the discriminative model. This result suggests that the tactile attributes are more informative of object identity for this experiment. However, it should be noted that allowing the robot to perform grasps differently across the objects may yield better classifications using only hand-configuration features.

Comparing the progression of the online methods, we observed that the discriminative model achieved marginally better scores early on, but the generative

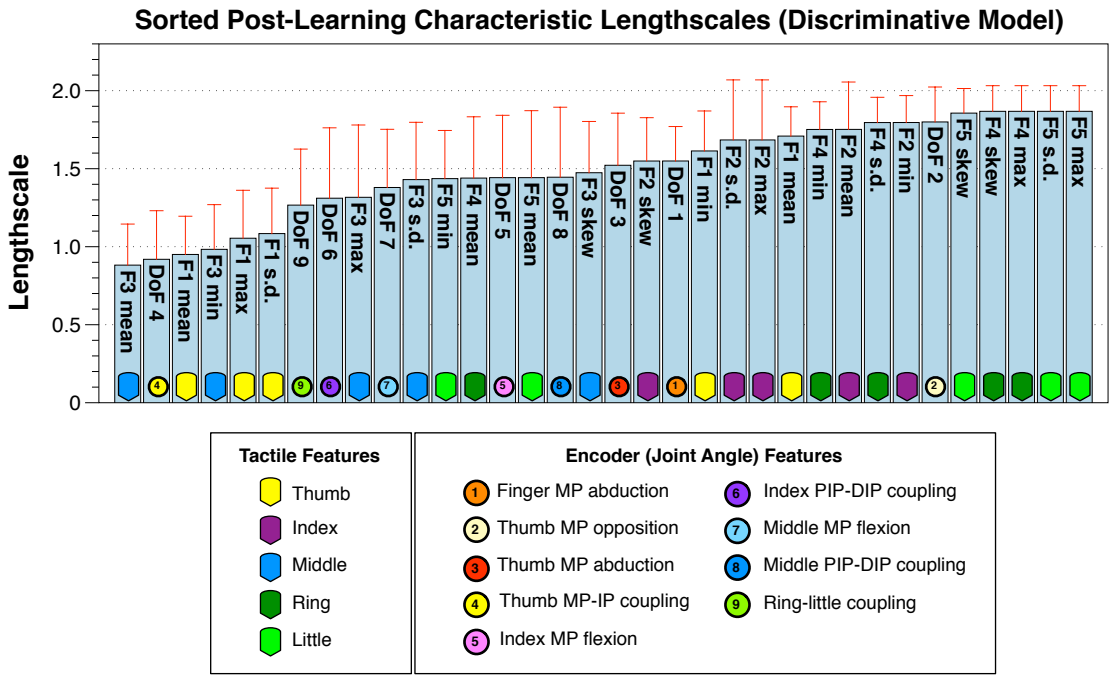


Figure 5.7: The characteristic lengthscales for the discriminative classifier post-learning. We observe that the lengthscales varied across the object classes. Overall, the most relevant features (as measured by average smallest lengthscales) were tactile features corresponding to the middle finger and thumb, as well as the thumb metacarpophalangeal-interphalangeal coupling.

model tended to achieve higher accuracies as more samples are provided. This phenomena is most clearly seen when using only the encoder values. That said, when using all features, both methods achieved perfect accuracy by the third trial set (after ≈ 60 object samples).

Looking closer at the classification errors, both methods made similar mistakes (complete confusion matrices are available as supplementary results in Appendix E). To summarise, the two most misclassified objects are the baseline class (no object) and the Half-full Soda Can. The baseline class was most frequently mistaken for the Monkey soft-toy and the Half-full Soda-can confused with the Half-Full Plastic Bottle. These mistakes are not surprising given the soft nature of the monkey toy (where the hand would sometimes close completely), and the size and compression similarities between the half-full containers. When using only the encoder values, the cans and plastic bottles were most frequently confused, as well as the lotion bottle and hardcover bottle, as these objects had similar widths and resulted in similar hand configurations.

Methodology	Algorithm	All Features	Tactile Features	Encoder Features
Online	Discriminative (All)	0.991 (0.026)	0.985 (0.025)	0.907 (0.066)
(OIESGP)	Discriminative (Final 20%)	1.000 (0.000)	0.992 (0.012)	0.936 (0.029)
	Generative (All)	0.991 (0.027)	0.989 (0.030)	0.921 (0.096)
	Generative (Final 20%)	1.000 (0.000)	1.000 (0.000)	0.970 (0.024)
Offline	C4.5	0.985 (0.012)	—	—
	SVM	0.835 (0.068)	—	—
	SVM (FS+Opt)	0.995 (0.001)	—	—

Table 5.1: Object classification scores show that our online methods perform well on the obtained dataset, achieving high accuracy scores relative to state-of-the-art offline methods. Accuracy scores for offline algorithms are the best 5-fold cross-validation accuracies. We also observed that the tactile information plays a crucial role in making a proper classification; using only the encoder features (joint angles) results in significantly lower accuracies.

Feature	Description
1-25	First three moments, max and min of the mean tactile data
26-50	First three moments, max and min of the sd. tactile data
51-75	First three moments, max and min of the skewness of the tactile data
76-165	Encoder readings during maximum and minimum tactile readings

Table 5.2: Features used for offline methods (computed for each finger separately).

Fig. 5.7 illustrates a sorted bar graph of the optimised lengthscales for the discriminative model. Although more data is needed for further optimisation, we observed the minimum lengthscales — indicative of importance — were different across the objects. By approximating feature relevance using the smallest average lengthscale, we ascertained the three most relevant features to be the mean tactile measurements of the middle finger and thumb, and the thumb metacarpophalangeal-interphalangeal encoder reading. Interestingly, the C4.5 decision tree method (described in the next section) selected a similar set with four features corresponding to the mean tactile reading for the thumb, middle and ring fingers and the standard deviation of the tactile reading of the thumb.

5.4.1 *Performance Comparison to Offline Classifiers*

We compared our results against the C4.5 decision tree (which was used with success in several research works on tactile classification [Chitta et al., 2010,

2011]) and SVM using the RapidMiner platform[2]. Since the offline methods do not directly work on temporal data, we created a feature vector similar to [Chitta et al., 2011] by computing statistics across the temporal dimension (shown in Tbl. 5.2). We optimised each algorithm using a grid-search to minimise the 5-fold cross-validation error. Specifically, we optimised the penalty term C and the RBF kernel width for the SVM. We optimised C4.5 parameters (i.e. leaf size, split criteria, minimal gain) in a similar manner.

The best 5-fold cross-validation accuracies of the two methods are shown in Tbl. 5.1. It is important to note that accuracy comparisons should be made *qualitatively* since the accuracy scores are computed differently from the online method. We observed that the decision tree performed well on this problem, better than the SVM by a significant margin. That said, both methods did not obtain the accuracy level attained by our methods on the normal dataset. We hypothesised that the SVM was having difficulty due to the large feature vector (165 features) and we performed an additional feature selection (FS) for the SVM using a genetic algorithm[3] (the objective function was the average 5-fold cross-validation accuracy), followed by an additional optimisation of algorithm parameters. Only after this computationally-heavy optimisation did the SVM attain high scores similar to our OIESGP classifiers.

5.4.2 *Early Classification*

One additional benefit of our approach is the ability to predict object labels before the grasping motion is complete. To, illustrate this point, Fig. 5.8 shows the "early" classification accuracies on the test set (using both tactile and encoder features) *during* the grasp. The scores are shown versus sequence proportion since the actual temporal lengths varied from one grasp to the next. Even with 10% of the data, both classifiers achieved a high accuracy of $\approx$ 98%. Indeed, both models achieve perfect accuracy when using only 30% of each sequence,

2 More information at http://www.rapid-i.com.
3 We also attempted to use PCA for feature reduction but this yielded suboptimal results.

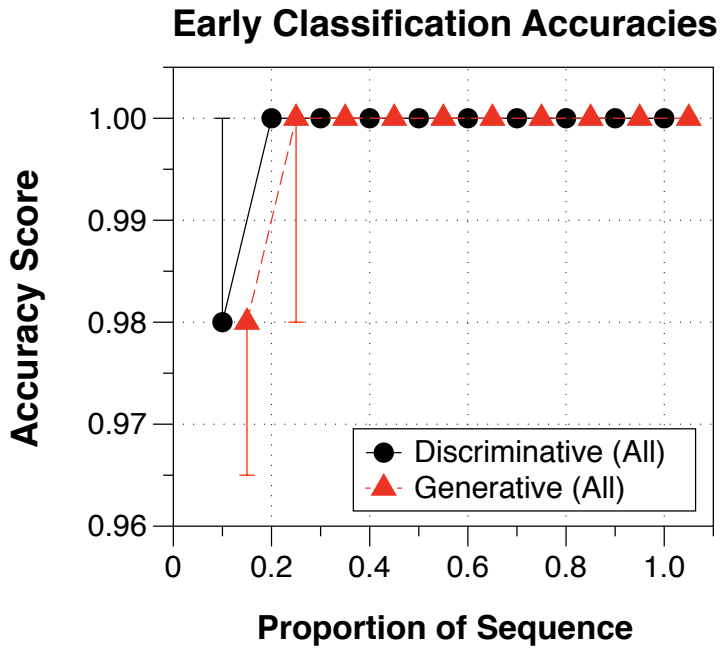
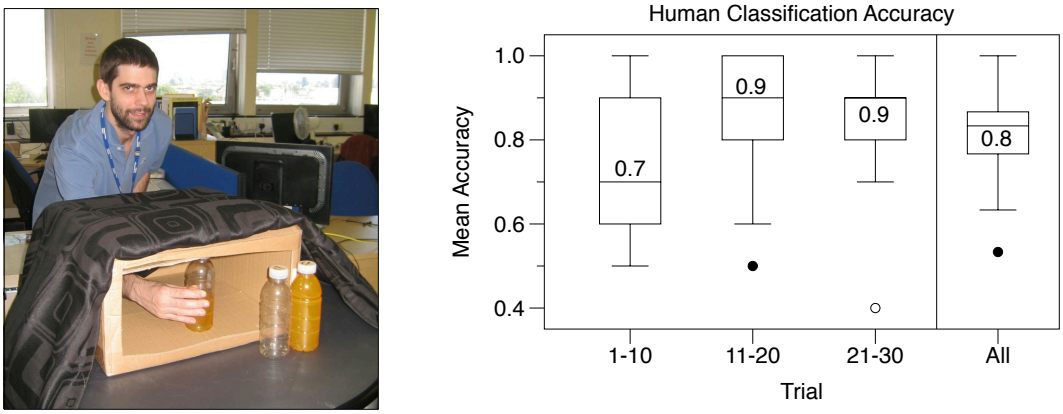


Figure 5.8: Median classification accuracies at different stages of the pressing motion. High accuracies are achievable even before the motion is complete — perfect identifications are made using only the first 20% and 30% of the sequence for the discriminative and generative classifiers respectively.

demonstrating that excellent object recognition is achievable without having to complete the entire motion.

5.4.3 *Computational Costs*

Our computational test-bed was a Apple Macbook laptop with a 2.6Ghz Intel Core-i7 processor and 16GB of RAM. For the discriminative classifier, each prediction iteration (for each observation) required an average of 0.0093s. Training was more expensive at 0.02s given the need to compute derivatives for hyper-parameter optimisation. Prediction and training times were marginally higher for the generative model at 0.0097s and 0.0265s respectively. Nevertheless, both models were capable of updating their internal states in less time than required to execute the grasping motion. In short, the training and predictions can be made in real-time given reasonable limits on the number of classes.



(a) (b)

Figure 5.9: Human online classification experiment with the three plastic bottles (of varying fullness). Participants were provided true-label feedback after each sample, giving them the opportunity to learn from mistakes. (5.9b) Human Classification Accuracy as trials progressed (with medians shown in the boxes) show an improvement in accuracy between trials 1-10 and 11-20. Differences between the first and second segments are statistically significant at the 95% level with $p = 0.033$.

5.4.4 *Performance Comparison to Human Subjects*

In addition to comparisons with offline classifiers, we sought to understand how human subjects would perform given the online-learning experimental setup with true-label feedback after each attempt. From preliminary tests, we discovered that humans easily achieved 100% accuracy when distinguishing the (easily deformed) soda cans, the soft toys (through texture), the lotion bottle and the hard-cover book. However, they appeared to have difficulty discriminating between the plastic bottles. Similar findings were reported by Chitta *et al.* [Chitta et al., 2011], where human subjects achieved a classification accuracy score of 75.2% when asked to discriminate between full and empty (both open and closed) plastic bottles — note however that in that study, the humans trained before-hand (until they were confident about their abilities) and were not offered the correct labels during the testing stage.

We invited 15 human subjects (ages 21-38 years, mean 28.5 years, 13 males) to participate in an experiment similar to the classification task described in Section 5.3 but limited to the three plastic bottles (Fig. 5.9a). In the first phase, each

	Predicted		
True Label	Empty	Half-Full	Full
Empty	111	37	2
Half-Full	17	110	23
Full	0	9	141

Table 5.3: Confusion Matrix for Full, Half-Full and Empty Plastic Bottles (Human Study)

participant was allowed to grasp each of the items once and told the object's class; they were not allowed to lift or otherwise move the bottles. In the second phase, they were asked to classify a randomly selected bottle (30 trials with stratified sampling) hidden in covered box. After each attempt, the participants were provided feedback (the true-label), giving them the opportunity to learn from mistakes.

The average human score was 80.4% (sd. 12.4%), lower than that achieved by iCub with OIESGP classifiers (which achieved 100% accuracy across 20 repeated tests with the order of test samples shuffled). From the confusion matrix (Tbl. 5.3), we observed that the full bottle was the easiest to classify. The empty and half-full bottles were more difficult to tell apart; we observed our participants frequently misclassified the empty bottle as half-full, particularly during earlier trials. The one participant who achieved perfect accuracy revealed to us after the experiment that he used a combination of temperature and bottle deformation to help him with the task. Since our methods can easily accommodate other sensor-streams, future work may fuse sensor data to better classify items.

To determine if the subjects improved as the trials progressed, we segmented the data into three portions, i.e., the first ten, second ten and final ten trials. As can be seen in Fig. 5.9b, the median accuracy scores rose from 70% to 90% from the first to the second segments and remained relatively stable after. The difference in the first and second segments were statistically significant at the 95% level ($p = 0.033$), in favour of the hypothesis that human beings learnt continuously from feedback. We also noted that the human subjects repeated their grasps in order to make a decision (especially when they were unsure). As

future work, we plan to investigate using the model's uncertainty estimates in a similar "active-touch" strategy.

5.5 SUMMARY AND CONCLUSIONS

In this chapter, we have demonstrated how the OIESGP can be used in discriminative and generative classification models, applied specifically to the task of distinguishing objects by touch. Our experimental results using the iCub anthropomorphic hand and ten object classes show that our online methods are capable of real-time classification and attained high accuracies comparable to highly-optimised offline classifiers such as the C4.5 decision tree and the SVM — when using both tactile and joint angle features, our methods achieve perfect accuracy on the test portion of our dataset.

Despite the excellent performance, our object set remains relatively small (with a substantial number of training samples) and our classifiers were forced to make a choice between known objects. In contrast, humans are able to discriminate between a very large number of familiar and novel items. Although our classifiers outperformed human subjects on the plastic bottle experiment, there may be inherent trade-offs between being able to positively perceive small differences in tactile feedback and scaling up to meet the demands of real-world object classification. Indeed, humans are able to vary palpations to better obtain sensory input and can leverage on additional features (such as weight) to better classify objects, relaxing the need to rely simply on tactile information obtained through a grasp. These reasons may explain why the human subjects did not achieve perfect accuracy in our experiment, but more importantly, suggest means of extending our work.

First, we may incorporate additional sensory data (e.g., heaviness and temperature) to better classify objects. The second, potentially more interesting, research thread is to explore the use of active classification, i.e., gathering additional sensory data when confidences are low. In a similar vein, active learning can be used to minimise the amount of samples required for training. These

issues, as well as ideas for scaling up through hierarchical modelling, are discussed further in Chapter 7.

Moving on, in the next chapter, we will combine discriminative and generative elements into a single cohesive model for learning assistance by demonstration. We demonstrate that this model can be trained online to provide real-time assistance to wheelchair users.

6

LEARNING ASSISTANCE BY DEMONSTRATION

Over the years, the robotics research community has propelled the development of the intelligent robot assistant, making important scientific and technological advances that have resulted in (among many other examples) smart wheelchairs that predict user intent [Carlson and Demiris, 2012] and robotic laparoscopic camera assistants that help doctors perform complex surgery [Nezhat et al., 2009]. That said, robotic assistants have yet to achieve widespread use and remain largely confined to laboratory and other controlled environments.

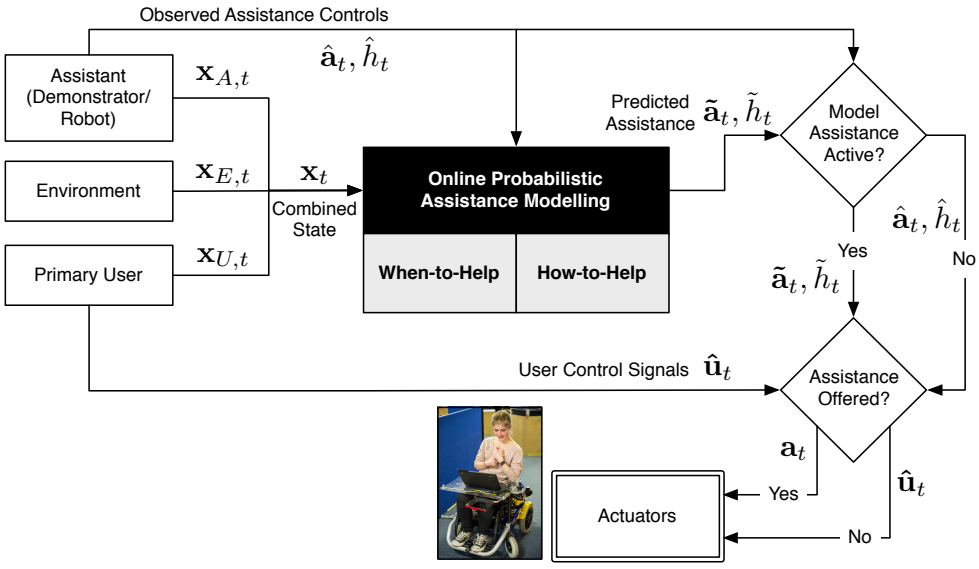


Figure 6.1: Learning-Assistance-by-Demonstration (LAD) System. Our model learns both *when* and *how* to assist iteratively from an assistant (demonstrator) helping a user accomplish a task.

One reason for this lack of expansion into real-world settings is that *proper* assistance (by human or robot) is challenging. By its nature, assistance is contextual, dependant not only on the current task, but also on the state of the user, the environment and the assistant's capabilities. Furthermore, competing goals may demand for different degrees of assistance. In educational settings for example, the primary goal is long-term user development rather than short-term task completion [Earwaker, 1998]. Although it may be possible to derive assistive policies from "first principles", it requires an inordinate amount of prior knowledge to be integrated into a robotic system.

In this chapter, we adopt a novel perspective on the problem, approaching it from a Learning-by-Demonstration (LbD) standpoint [Dillmann, 2004; Argall et al., 2009]. As compared to developing policies by-hand, LbD methods derive policies from teacher demonstrations. In a similar fashion, we aim to derive assistive policies by *observing an assistant*; an approach we term as Learning Assistance by Demonstration (LAD) [Soh and Demiris, 2013].

Framing assistive learning in this way allows us to retain many of the benefits associated with LbD; for complex tasks, demonstration is often more intuitive than hand-coding specific behaviours, allowing non-roboticists to participate in policy development [Argall et al., 2009].

In addition, LAD augments LbD by focussing on the assistive element. Current LbD systems are *task-centred* in that they focus on deriving a policy for completing the demonstrated task with or without a human-in-the-loop. For example, to teach a smart wheelchair to navigate, we would provide it with driving demonstrations to derive a "how-to-drive" policy (e.g., [Chow and Xu, June]). When the smart wheelchair is assisting a user, the robot faces the difficult decisions of not only *how* best to assist but also *when* it is appropriate to intercede. One approach to solve this is by inferring the user's intent from (potentially noisy) observations [Carlson and Demiris, 2008; Demiris, 2007], using the learned policies and failure models. In contrast, LAD is a *direct approach*. Instead of deriving a policy for "how-to-drive", we extract a policy for "how-to-help-a-user-drive".

A natural follow-up question is how to learn assistive policies from demonstrations? As a solution, we present a novel probabilistic mixture model using our developed OIESGP regressor and classifier. Described in Section 6.2, this model is trainable online during the demonstration process (which facilitates interactive teaching) and more importantly, places emphasis on capturing both when and how to appropriately assist.

To demonstrate the feasibility of the LAD approach and our associated model, we focus on the problem of smart assisted mobility. We conducted experiments using a simulated robot and on a real-world smart wheelchair. Section 6.3 describes our simulation experiment and presents model accuracy results. Section 6.4 extends this discussion to a real-world experiment on the ARTY smart wheelchair platform [Soh and Demiris, 2012b, 2011] with human participants using haptic controllers. Finally, Section 6.5 concludes this chapter with a summary and ventures research questions arising from this work.

6.1 LEARNING ASSISTANCE BY DEMONSTRATION: PROBLEM STATEMENT

In this section, we formally introduce LAD and detail its elements and distinguishing features. To help guide this discussion, Fig. 6.1 illustrates an overview of our system model.

Recall that we are principally interested in robots that learn how to assist humans accomplish tasks through observation of an assistant. For simplicity, let us consider a scenario where there is one assistant (U_A) helping a primary user (U_P) complete a primary task (T_P)[1]. In effect, U_A is engaging in an *assistive task* (T_A) and our robot's goal is to extract an assistive policy, i.e., a mapping from states $\mathbf{x}_t \in \mathcal{X}$ to assistive actions $\mathbf{a}_t \in \mathcal{A}$, denoted $\pi_A(\mathbf{x}_t) : \mathcal{X} \to \mathcal{A}$. This should be contrasted against the typical LbD approach which is to model the primary task and obtain a task policy $\pi_P(\mathbf{x}_t) : \mathcal{X} \to \mathcal{U}$ where $\mathcal{U}$ is the set of control actions.

To enable us to extract this policy, we have access to (partial and noisy) observations of the state $\mathbf{x}_t = (\mathbf{x}_{A,t}, \mathbf{x}_{U,t}, \mathbf{x}_{E,t})$ where $\mathbf{x}_{A,t}$, $\mathbf{x}_{U,t}$, $\mathbf{x}_{E,t}$ are the state of

1 Our method can be extended to scenarios involving multiple assistants and primary users.

the assistant, the user and the environment. Note that $\mathbf{x}_{A,t}$ represents either the assistant or the robot, depending on whether the robot is observing or executing the policy. We also observe the actions performed by U and A, denoted as $\hat{\mathbf{u}}_t$ and $\hat{\mathbf{a}}_t$ respectively. In addition, we assume a visible signal $\hat{h}_t$ which indicates when the assistance is given.

To make these concepts more concrete, let us consider a specific intelligent wheelchair training scenario where an occupational therapist U_A is teaching a young boy U_P with special needs to navigate his environment. In addition, she wishes to train the smart wheelchair R to assist the child when she is not present. To do so, whenever U_P is unable to complete a particular sub-task or is having difficulty, U_A engages a "guidance-mode" where she takes control and guides U_P appropriately with actions $\hat{\mathbf{a}}_t$. R is able to observe these actions as well as the states $\mathbf{x}_t$ when interventions occur, and aims to extract the policy $\pi_A(\mathbf{x}_t)$ used by U_A.

An important element of our approach is the recognition that assistance may not be needed (or desired) continuously but only at *key* points during the process [Demiris, 2009a]. This is relevant in many real-world applications where constant intervention may interfere with task completion or competing objectives. Using our example, if U_A does not provide sufficient assistance, the child may become frustrated with the activity. On the other hand, if U_A assists too much, the child learns to rely on her assistance, which negatively impacts his long-term development (a phenomena known as "learned helplessness" [Seligman, 1972]). As such, deciding when to assist is paramount. In the next section, we discuss our probabilistic assistive model which formally captures these elements.

6.2 AN ONLINE PROBABILISTIC MODEL FOR LAD

At a high-level, our model is a representation of an assistant making two choices at each time-step; 1) whether she should help and if so, 2) what she should do.

Let us begin by assuming that the assistive actions are continuous random variables $\mathbf{a}_t \in \mathbb{R}^D$, which can represent motor commands to an actuator for example. In addition, we define a "null" or zero action $\mathbf{a}_0 \triangleq \mathbf{0}$ (the assistant is not helping in the task). Our Bayesian model considers that assistive actions $\mathbf{a}_t$ are generated from an "assistance component" or a "do-nothing component" (where the action probability is a Dirac delta centered at $\mathbf{a}_0$):

$$p(\mathbf{a}_t|\mathbf{x}_t) = p(h_t = 1|\mathbf{x}_t)\mathcal{N}(\mathbf{a}_t|f(\mathbf{x}_t), \mathbb{V}[f(\mathbf{x}_t)]) + (1 - p(h_t = 1|\mathbf{x}_t))\delta_0(\mathbf{a}_t). \tag{6.1}$$

A critical part of our model is the probability of assistance (when-to-help) represented by a discrete/binary random variable $h_t \in \{-1, 1\}$ conditioned on the current state $\mathbf{x}_t$. In this work, h_t and $\mathbf{a}_t$ are modelled using a Gaussian Process classifier (GPC) and regressor (GP) respectively:

$$h_t|\mathbf{x}_t \sim \mathcal{GPC} \tag{6.2}$$

$$f(\mathbf{x}_t) \sim \mathcal{GP} \tag{6.3}$$

Since standard GPs are computationally expensive and we prefer our robotic assistants to learn iteratively from demonstrations, we make use of the OIESGP developed in the previous chapters. To increase modelling power without having to increase the BV set size (avoiding the additional expense associated with the matrix inversions needed for hyperparameter optimisation), we re-introduce the local modelling concepts discussed in Chapter 2. Here, the binary variable h_t is modelled with a mixture of OIESGP discriminative classifiers (using the probit likelihood introduced in Chapter 5), and the assistive actions are modelled using a mixture of OIESGP regressors.

6.2.1 *Mixtures of OIESGP Experts*

Recall from Chapter 2 that local models maintain a collection of smaller submodels that partition the space. Similarly, Mixture of Experts (ME) models [Jor-

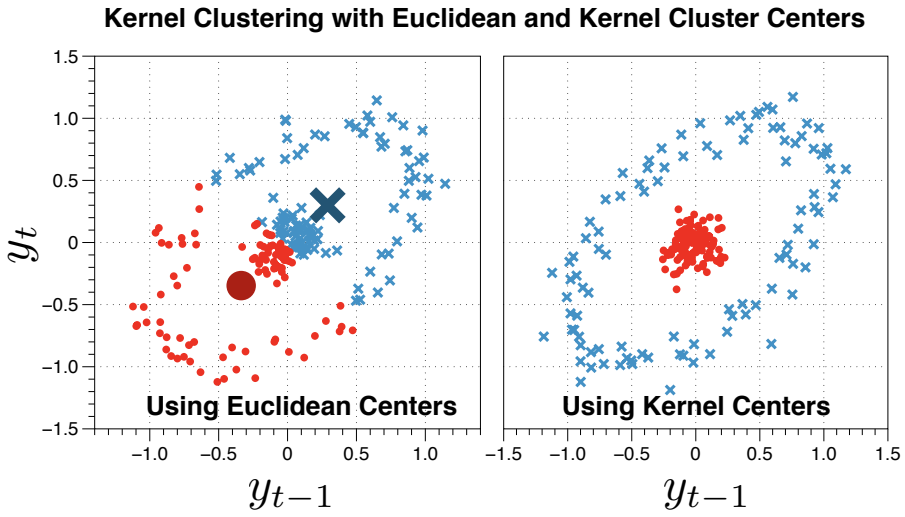


Figure 6.2: Kernel Clustering using Euclidean centers produced a linear and incorrect partitioning of the space, which was resolved by using the kernel centers in the projected space.

dan and Jacobs, 1994] consider the probability of the observed output comes from a linear mixing of K base experts:

$$p(y_t|\mathbf{x}_t, \boldsymbol{\Theta}) = \sum_{i=1}^{K} \pi_k p_k(y_t \mid \mathbf{x}_t, \theta_k) \tag{6.4}$$

where $\boldsymbol{\Theta}$ represent all the model parameters, θ_k are the parameters for the k-th GP, and π_k are the mixing weights, $\pi_k = p(z_t = k \mid \mathbf{x}_t, \zeta)$ where ζ are the *gating function* parameters and z_t are the indicator variables assigning samples to experts. For an offline LAD mixture model, we may apply the GP mixtures proposed in [Tresp, 2001; Rasmussen and Ghahramani, 2002].

Here, we present an online approximation using OIESGP experts where we apply online kernel k-means to partition the space. Our approach differs from LWPR and LGP, which can produce pathological clusterings due to the use of Euclidean cluster centers. Fig. 6.2 shows a concrete example of this where we applied online clustering on data generated from a simple switching dynamical system[2]. Here, kernel clustering using the *centers in the projected space* generated a more accurate and intuitively appealing partitioning of the space (given the underlying system).

2 The dynamical system employed was similar to the simple linear rotating system discussed in 4.4, but with two modes of different radii and switching between the modes every 50 time-steps.

6.2.1.1 *Kernel k-means Clustering*

Unlike the supervised learning methods we have discussed up to this point, k-means is an *unsupervised learning* method. In contrast to regression and classification, we are not given the "true" outputs. Instead, unsupervised methods "discover" structure in data [Murphy, 2012]. For *clustering* tasks in particular, we seek to partition the data into coherent parts, i.e., groups with similar samples.

The k-means algorithm is an iterative (offline) method that clusters samples $\mathbf{x}_i$ into K clusters to minimise the cost (loss) function:

$$\mathcal{L}(M, Z) = \sum_i ||\mathbf{x}_i - \mathbf{m}_{z_i}||^2 \tag{6.5}$$

where $M = \{\mathbf{m}_k\}$ for $k = 1, \ldots, K$ is the set of the cluster centers and $Z = \{z_i\}$ is the set of assignment variables indicating which cluster the sample $\mathbf{x}_i$ belongs to. The solution to this optimisation problem is achieved by iteration of two steps:

1. Assign the points to the closest cluster: $z_i = \arg\min_k ||\mathbf{x}_i - \mathbf{m}_k||^2$
2. Re-compute the cluster centers: $\mathbf{m}_k = \sum_{\mathbf{x}_i \in \mathcal{C}_k} \mathbf{x}_i / |\mathcal{C}_k|$ where $\mathcal{C}_k$ is the set of samples assigned to cluster k.

For kernel k-means, the overall structure of the algorithm remains unchanged. The difference is in the distance computation; instead of using the Euclidean distance in input space, we find the distance to the cluster center in projected space:

$$d(\mathbf{x}_i, \mathbf{m}_k^\phi) = ||\phi(\mathbf{x}_i) - \mathbf{m}_k^\phi||^2 \tag{6.6}$$

where the cluster center is given by:

$$\mathbf{m}_k^\phi = \frac{\sum_{\mathbf{x}_i \in \mathcal{C}_k} \phi(\mathbf{x}_i)}{|\mathcal{C}_k|}. \tag{6.7}$$

Using the property of Mercer kernels, we can apply kernel evaluations instead of dot products:

$$d(\mathbf{x}_i, \mathbf{m}_k^{\phi}) = k(\mathbf{x}_i, \mathbf{x}_i) - \frac{2\sum_{\mathbf{x}_j \in \mathcal{C}_k} k(\mathbf{x}_i, \mathbf{x}_j)}{|\mathcal{C}_k|} + \frac{\sum_{\mathbf{x}_j, \mathbf{x}_l \in \mathcal{C}_k} k(\mathbf{x}_j, \mathbf{x}_l)}{|\mathcal{C}_k|^2} \tag{6.8}$$

6.2.1.2 *Online Clustering for OIESGP experts*

When data arrives sequentially, we can approximate the full kernel k-means solution in a greedy fashion to partition the space incrementally. Here, we process samples $\mathbf{x}_t$ one at a time to update a set of OIESGP experts.

Given the maximum number of experts K_{max}, the model experts m_k for $k = 1, \ldots, K_t$ where $K_t \leqslant K_{max}$ and K_{max} is user-defined. Each expert has a basis vector set $\mathcal{B}_k$. Similar to LWPR and LGP, we employ a cluster creation threshold w_{gen} to determine when a cluster should be spawned — however, the number of experts is bounded by K_{max}. Updating the model proceeds as follows:

1. Compute the distances, $d_{k,t}$, to each of the k centers via Eq. (6.8) using $\mathcal{B}_k$ in place of $\mathcal{C}_k$.

2. If $\exp(-d_{k,t}) < w_{gen}$ for all k and $K_t \leqslant K_{max}$, we create a new expert with the sample $\mathbf{x}_t$, increment K_t and proceed to the next sample.

3. Otherwise, we assign $\mathbf{x}_t$ to the closest expert, i.e., $z_t = \arg\min_k d_{k,t}$ and update the OIESGP m_k with the sample.

Although the distance computation (6.8) appears to be on the order of $O(|\mathcal{B}_k|^2)$, it is straightforward to maintain and update last term on the RHS for each cluster (on the order of $O(|\mathcal{B}_k|)$ for each sample). Then, calculating the distance to each cluster takes, in the worst case, $O(s_{\mathcal{B}})$ for each expert[3] and hence, $O(K_{max} s_{\mathcal{B}})$ in total. In summary, the total distance computation time grows linearly with total number of basis vectors stored across all experts and is bounded by the total number of stored BVs.

Training can be performed online as the demonstration is being conducted. For example, in our experiments, the when-to-help model is continuously trained

3 Recall that $s_{\mathcal{B}}$ specifies the maximum BV set size.

throughout the demonstration with both positive and negative observed samples $(\mathbf{x}_t, \hat{h}_t)$. The how-to-help model is only trained when assistance is offered $(\hat{h}_t = 1)$ with the observed sample pairs $(\mathbf{x}_t, \hat{\mathbf{a}}_t)$.

6.2.2 *Prediction and Control*

Once both models are trained, we can apply them directly in an assistive policy $\pi_A(\mathbf{x}_t)$. To make a prediction, we used a simple winner-takes-all scheme, i.e., predictions are made only with the "closest" expert:

$$\mu_t = \sum_k \delta_{k,z_t} \mu_{k,t} \tag{6.9}$$

where $\mu_{k,t}$ is the predicted mean of expert k using input $\mathbf{x}_t$, δ_{k,z_t} is the Kronecker delta and $z_t = \arg\min_k d_{k,t}$. Preliminary tests indicated that this approach worked effectively for both the how-to-help regressor and when-to-help classifier. For assistive control, we applied a simple threshold W_h and assistance $(\tilde{h}_t = 1)$ was offered whenever the probability of assistance $p(h_t|\mathbf{x}_t) \geqslant W_h$. The threshold W_h is a user-defined parameter that controls the minimum level of confidence before offering assistance. The assistive control signals, $\tilde{\mathbf{a}}_t$, are taken as the prediction of the OIESGP experts using Eq. (6.9) above. We note that more complex policies can be derived to make use of the variance of the assistive action distribution, and this will be explored in future work.

6.3 SIMULATION EXPERIMENT: ASSISTED MOBILITY

To illustrate how LAD and our probabilistic model can be applied, let us focus on the real-world problem of assisted mobility for individuals with special needs. A recent survey article [Simpson et al., 2008] assessed that 61-91% of all wheelchair users (1.4 to 2.1 million people in the US) could gain from an assistive smart wheelchair. Research in this area has produced a multitude of obstacle avoidance algorithms and real-world platforms that help wheelchair

users avoid collisions [Tsotsos et al., 1998; Yanco, 1998a; Gomi and Griffith, 1998; Miller and Slack, 1995; Levine et al., 1999; Simpson et al., 2004; Carlson and Demiris, 2008, 2010b]. For a more complete review of smart wheelchairs, we refer readers to [Simpson, 2005]. Recent research work has applied LbD to driving wheelchairs in cluttered environments [Chow and Xu, June] but the focus of this study was the transferability of human *driving* skills. In contrast, we are interested in transferring *assistive* skills.

As a prelude to our real-world trials, we conducted a simulation study designed to test if our model was capable of capturing when and how to help a wheelchair user after a minimal set of demonstrations.

To simplify analysis, let us consider a first-cut scenario that assumes that the user lacks a specific fine motor control skill which inhibits his ability to make sharp right turns. This loss is permanent and thus, assistance should be offered to prevent user frustration. Since the user is otherwise capable of controlling the joystick, assistance should withheld during "soft" right turns (desired turning angle was less than 60 degrees), left turns or forward/backward movements to encourage the development of wheelchair driving skills. The efficacy of our approach was measured using the model accuracy (using the human demonstrator signals as the "gold standard") and the time required to complete a lap around a driving course.

6.3.1 *Experimental Setup*

Based on the real-world environment shown in Fig. 6.3, we created a simulated world using the Stage simulator [Gerkey et al., 2003]. Our test subject was a autonomous robot driver (RD) developed using the Robot Operating System (ROS) [Quigley et al., 2009] navigation stack[4]. For this test, RD's maximum forward and rotational velocities were set at 0.7 m/s and 0.5 rad/s and local obstacle avoidance was performed using Dynamic Window Approach (DWA) [Fox et al., 1997].

4 More details on developing autonomous mobile robots can be found at http://www.ros.org/wiki/navigation.

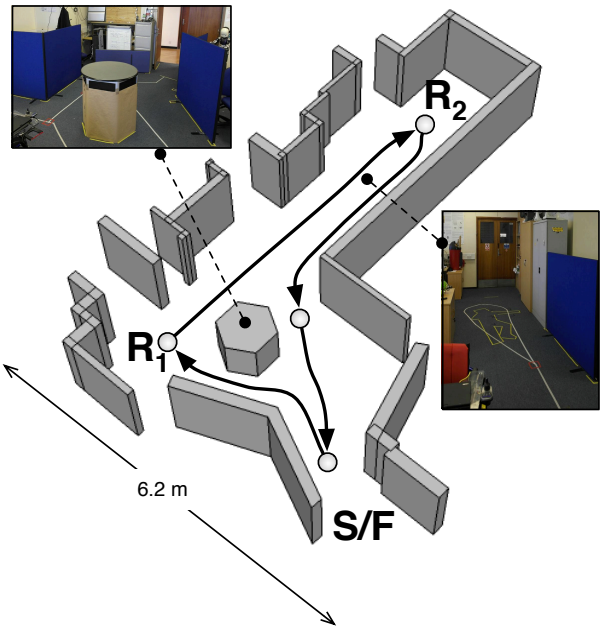


Figure 6.3: Driving course used in our simulation and preliminary trials. Our simulated robot tasked to drive from S to F, passing through and making four sharp right turns at R_1 and R_2.

We tasked RD to drive from the start point (S), through R_1 and R_2 and back to the finish point (F). Note that RD always made right turns at both R_1 and R_2. Under normal operation, RD was able to complete a lap in an average of 44.0 s. We induced the control limitation by scaling right turns to a maximum of 0.15 rad/s. This simple constraint had a drastic impact on RD's lap performance, increasing lap times to 61.6 s.

6.3.1.1 *Assistance Demonstration*

Assistance was provided in the form of a control takeover (in the spirit of "hand-over-hand" control used by occupational therapists) during the right turns at R_1 and R_2. Assistance was not offered at other portions of the map. Demonstration was performed using a wireless joystick controller which provided an observable takeover signal, $\hat{h}_t$ (1 whenever assistance was provided and -1 otherwise) and the assistive command velocities, $\hat{\mathbf{a}}_t = (\hat{a}_{x,t}, \hat{a}_{\theta,t})$. During takeover, the assistive control signals replace those sent by RD (an alternative approach would be to augment RD's control velocities).

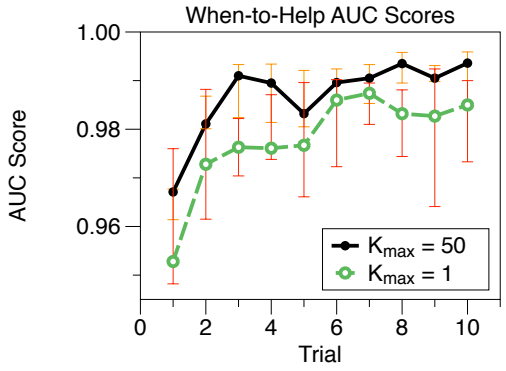
Parameters	When-To-Help	How-To-Help
Capacity (C)	50	50
Recursion Depth (τ)	10	10
Temporal scale (σ_ρ^2)	1.01	1.01
Lengthscale (l_U)	1.0	0.1
Lengthscale (l_A)	1.0	0.1
Lengthscale (l_E)	2.0	2.0
Signal Variance (σ_f^2)	0.1	0.05
Noise (σ_n^2)	0.1	0.01
Assistance Threshold (W_h)	0.5	—
Expert Creation Threshold (w_{gen})	0.4	0.3
Maximum Number of Experts (K_{max})	50	50

Table 6.1: Assistive Model Parameters. The same parameters were used for the simulation and the real-world trails.

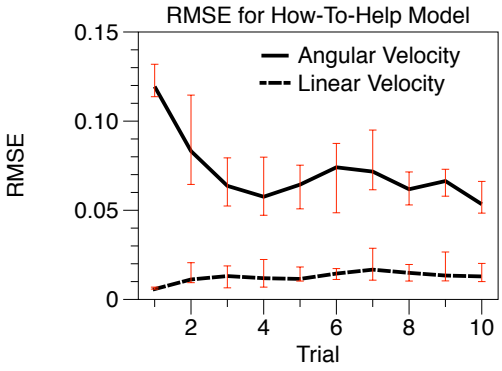
6.3.2 *Model Setup and Parameters*

In this experiment, the (partially-observable) state of the system was modelled as $\mathbf{x}_t = (\mathbf{x}_U, \mathbf{x}_A, \mathbf{x}_E)$ where the state of the robot was represented by its current velocities $\mathbf{x}_A = (v_{x,t}, v_{\theta,t})$ and the environmental state was captured by forward laser scan readings (separated into $M = 15$ segments), $\mathbf{x}_E = \mathbf{s}_t = (s_{1,t}, s_{2,t}, \ldots, s_{M,t})$. Since we did not employ "user-specific" sensors, we used the user's desired translational and rotational velocities as a proxy for intent (an internal state), i.e., $\hat{\mathbf{u}}_t = \mathbf{x}_U = (u_{x,t}, u_{\theta,t})$. More state complexity can be accommodated by our model but we found this representation to be sufficient for the task. Note that, in the limited control state, the model is only able to observe the scaled right turns (instead of the full commanded rotational velocity of 0.5 rad/s).

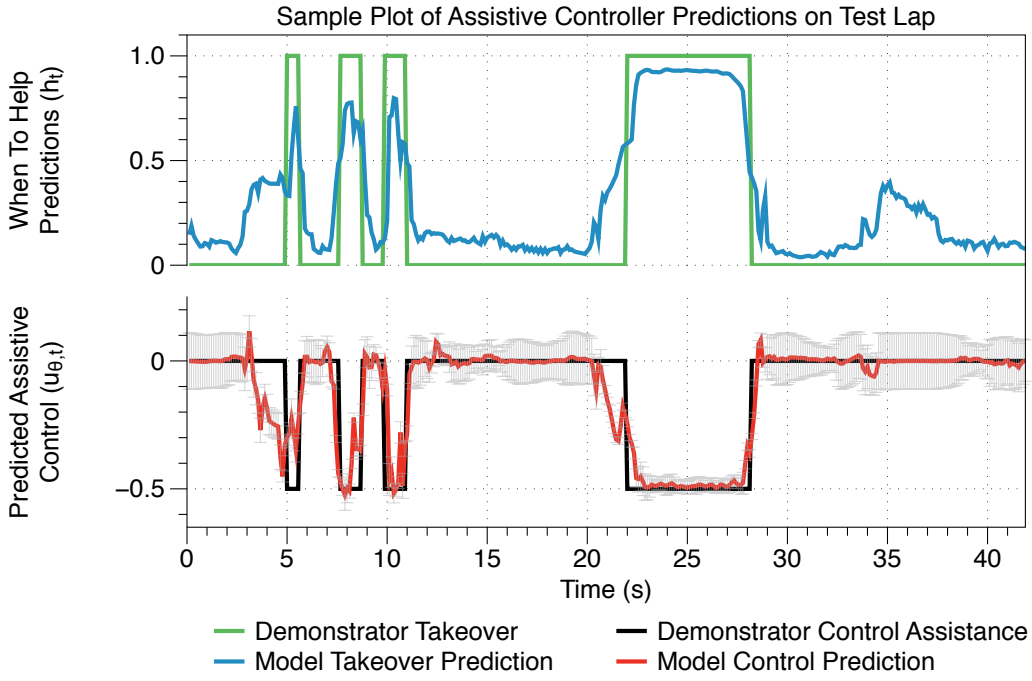
The parameters for our probabilistic assistive model are shown in Table 6.1. In addition to the default $K_{max} = 50$ setting, we experimented with $K_{max} = 1$ to investigate the differences when the LAD model consisted of only one OIESGP regressor and classifier.



(a) When-to-Help AUC



(b) How-to-Help RMSE



(c) Model Predictions during a Sample Test Trial

Figure 6.4: Performance measures summarising LAD model accuracy. The When-To-Help model achieved a high AUC score (0.97 – 0.99), which progressively improved as more samples were provided. This was higher than the single-expert model (0.95 – 0.98). In addition the How-To-Help model attained relatively low ($\approx$ 0.07) errors for the turning command velocities.

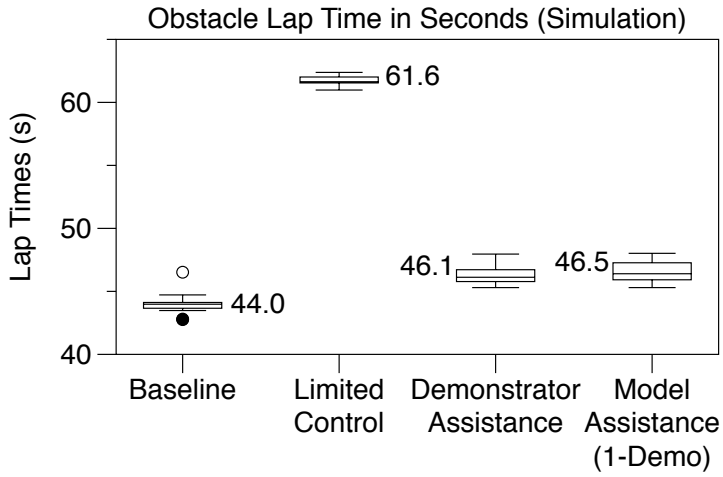


Figure 6.5: Lap times in seconds. Without assistance, the control limited robot (RD) completed a lap in 61.6 s. With human demonstrator assistance, RD was able to complete a lap in approximately 46 s. With a *single* demonstration, our learned model was able to improve lap times to 46.5s (a speedup improvement of 133%), statistically similar to a human demonstrator.

6.3.3 *Simulation Results*

To better analyse the learning capability of our model, we used ten supervisor demonstrations as a single dataset to train and test our model. The model learned in an online fashion, i.e., trained/tested at each time step during nine trials. The final tenth trial was used for testing (no training was applied). Fig. 6.4 summarises the performance of both the classifier and regressor averaged over fifteen runs (with the ten supervisor demonstrations randomly permuted).

The AUC scores obtained by varying the assistance threshold W_h, are shown in Fig. 6.4a. Our trained model achieved good scores (0.97 - 0.99), indicative of high true positive to false positive ratios and strong classifier performance. The median AUC scores were 0.990 by the ninth trial and 0.993 for the final test trial, higher than that when using only a single model (0.985); the number of experts grew to 10 experts[5] for the classifier but remained constant at one for the regressor.

Turning our attention to assistive control accuracy, we see from Figs. 6.4b and 6.4c that the OIESGP is able to generate assistive actions that are very similar to those generated by the demonstrator. The rotational command velocities have

5 The number of experts varied from 9 to 11 depending on the trial.

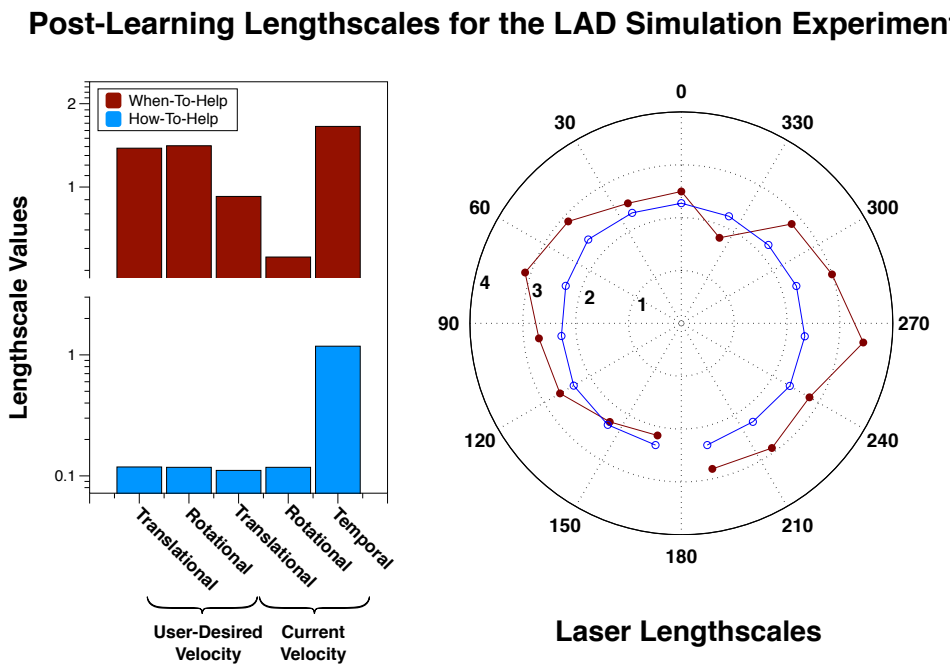


Figure 6.6: Post-Learning Lengthscales for LAD Simulation Experiment. The When-To-Help component manifested more change in the lengthscale values, suggesting that the most relevant features (with minimum lengthscales) were the wheelchair velocities and the front laser-right reading.

higher RMSE as compared to the linear controls; this may be explained by the fact that the help was principally rotational (and thus more varied) in nature. Interestingly, Fig. 6.4c shows that the model is able to "anticipate" when assistance should be given — we see a rise in the model's predicted $p(h_t)$ and a decrease in the variance of $p(\mathbf{a}_t)$ well before the assistance is actually performed.

Fig. 6.5 illustrates the lap times attained by RD when assisted by a human demonstrator and by our learned assistive model (over 10 independent trials). After only a single demonstration, our model is able to improve RD's lap time to 46.5 s; a speedup of 133% over the control limited scenario and statistically similar to that of the human demonstrator (Kolmogorov-Smirnov test with $p = 0.67$ and test statistic $k_s = 0.30$).

6.3.3.1 *Lengthscales*

The lengthscales obtained after the training laps are shown in Fig. 6.6. The When-To-Help component exhibited more change in the lengthscale values, and the most relevant features (with smallest values) were the wheelchair velocities

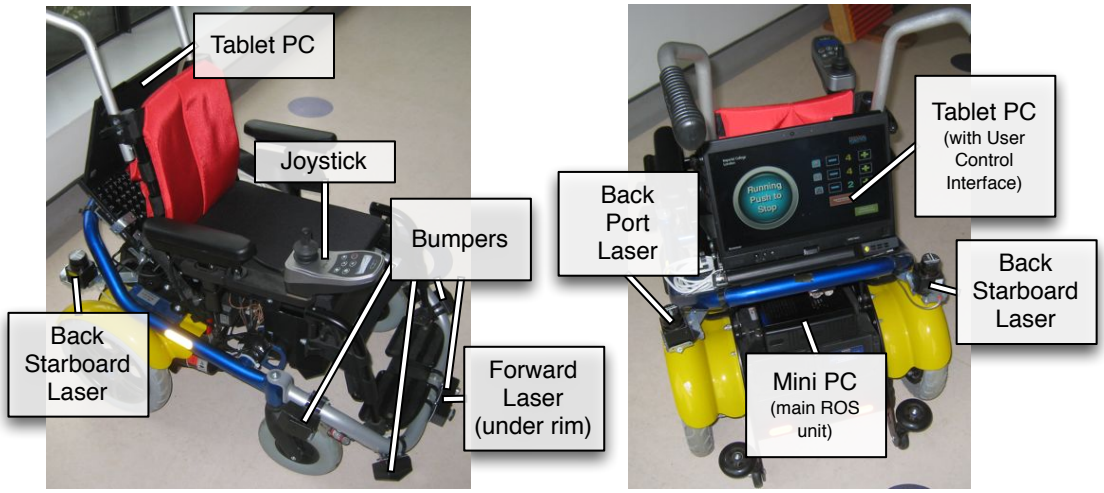


Figure 6.7: The Assistive Robotic Transport for Youngsters (ARTY) Smart Wheelchair [Soh and Demiris, 2011, 2012b].

and the front-right laser reading. This was surprising since we expected the user input to have the most relevance. We posit that since the classifier was only able to observed the scaled command values, the wheelchair velocities (over time, with the laser scan values) were found more informative of when to start and stop assistance.

6.3.3.2 *Computational Times*

The How-to-Help component took an average of 0.021 s (sd. 0.0072 s) per iteration, with a maximum of 0.0695 s when the GP was being rebuilt using new hyperparameters. The When-To-Help classifier's processing time was similar: an average of 0.0128 s (sd. 0.0145) per iteration, with a peak 0.0730 s. As such, our LAD model was capable of operation at $\approx$ 10Hz. Note that using the mixture of 10 experts (with a maximum of 50 BVs each) was computationally cheaper than using a single model with 500 basis vectors. On a separate test, the single model variant with $s_{\mathcal{B}} = 500$ required approximately 2.5 s per iteration (with hyperparameter optimisation enabled).

6.4 REAL-WORLD EXPERIMENT WITH ARTY AND HAPTIC CONTROLLERS

Based on the positive simulation and preliminary results with the joystick controller (detailed in Appendix F), we conducted an real-world experiment using the ARTY smart wheelchair (Fig. 6.7) [Soh and Demiris, 2012b, 2011] with 15 human participants using paired haptic controllers. In this section, we describe our experimental platform, set-up and obtained findings.

6.4.1 *Robot Platform: ARTY Smart Wheelchair*

Our experimental platform was the Assistive Robotic Transport for Youngsters (ARTY), designed to enable more children to benefit from independent mobility. ARTY was built "on-top" of the Skippi, an electronic powered indoor/outdoor chair (EPIOC) designed specifically for children. We chose the Skippi as our base unit because it fulfilled some of the requirements identified by Orpwood's study [Orpwood et al., 2005]; in addition to working both indoors and outdoors (maximum speed of 6 km/h and a maximum range of 30 km), the Skippi is colourful, easily transportable, has adjustable seats, is relatively lightweight, has batteries that last for more than a day and was not prohibitively expensive.

An attractive feature was that Skippi uses a controller-area network (CAN) based electronic system. CAN is a message-based protocol originally designed for automotive systems but is now used in a variety of devices from powered wheelchairs to the iCub humanoid robot [Metta et al., 2008]. All interacting modules (e.g., joystick, motor system) on the wheelchair are identified as CAN nodes and exchange information via messages. Our "smart" components interface with the base-wheelchair via the CAN network.

6.4.1.1 *Sensors and Computational Units*

To sense its environment, ARTY is equipped with three Hokuyo URG-04LX laser scanners and five bump sensors. These sensors are connected via USB to a mini-PC powered by an Atom processor. This "lower-level" unit is responsible

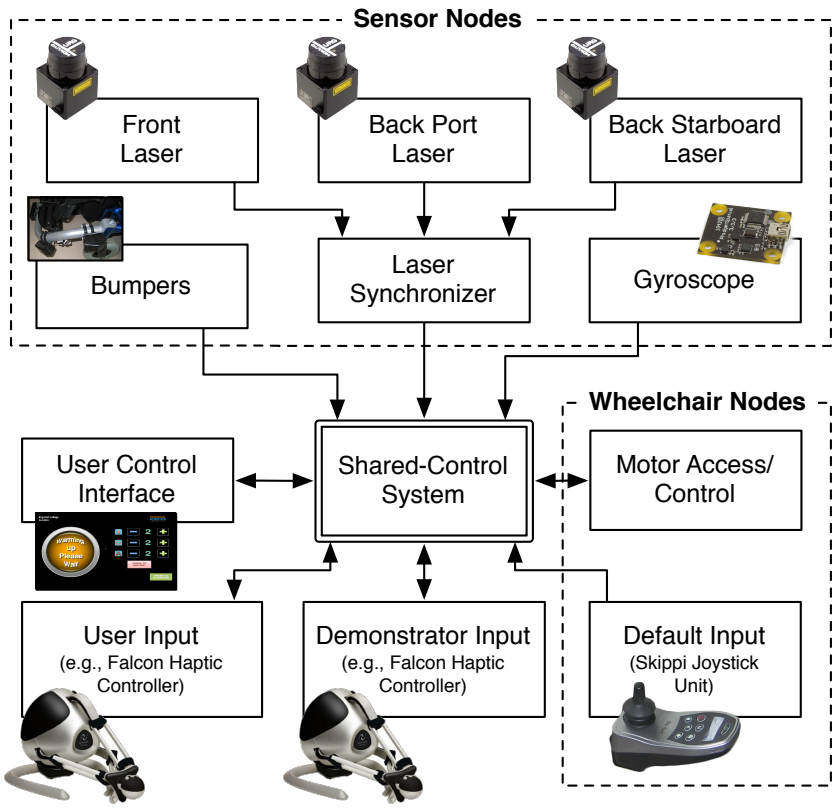


Figure 6.8: High-level schematic of ARTY's Software Components (ROS Nodes). In addition to the sensor and wheelchair control nodes, the system accomodates LAD via the demonstrator and user input nodes for alternative inputs such as haptic devices.

for integrating sensory information. Higher-level path planning and obstacle avoidance is performed by a Tablet PC connected via Ethernet. Splitting the processing tasks allowed us to decrease response time (since the tablet PC had higher computational capability) and to accommodate future expansion. The tablet-PC also presents a more natural touch-interface for users to change the wheelchair's basic settings.

6.4.1.2 *Software System*

ARTY's software system (high-level schematic shown in Fig. 6.8) was developed using the Robot Operating System (ROS) [Quigley et al., 2009], an open-source, thin, robotic platform that supports distributed processing.

In our system, each sensor is managed by its own ROS node running on the mini-PC. Since ARTY has three lasers, it was necessary to synchronise them to obtain a coherent obstacle map. Communication with the base wheelchair is

the responsibility of two interface nodes: the motor access/control (MAC) node and the joystick (default input) reader. Both nodes perform the necessary translations from CAN messages to desired commands velocities and vice-versa. In addition, the MAC node provides odometry information for mapping/localisation. The user interface node on the Tablet PC provides a simple means of turning on/off the wheelchair and changing basic settings such as the maximum translational and rotational velocities.

Finally, the central component of our system is the shared-control (SC) module which changes the user's control commands with the LAD signals and obstacle avoidance to prevent collisions. As can be seen in Fig. 6.8, the SC module takes in information from the sensor, user input and underlying wheelchair components, and generates appropriate motor controls.

6.4.1.3 *Safeguarding with DWA*

To prevent damage to the wheelchair and potential injury to participants, safeguarding using a limited form of the Dynamic Window Approach (DWA) [Fox et al., 1997] was employed to prevent hard collisions with obstacles. DWA is widely-known in robot navigation and collaborative versions have been used in smart wheelchairs such as Sharioto [Demeester et al., 2007] and the Bremen Wheelchair Roland III [Rofer, 1998]. Here, we used DWA to slow the wheelchair to prevent collisions by scaling the commanded velocities. To maintain the overall flow of this chapter, details on our obstacle avoidance methods and experiments with eight able-bodied children and one child with special-needs have been deferred to Appendix H.

6.4.1.4 *Paired Haptic Controllers*

The haptic controller used in this study was the low-cost[6] Novint Falcon, a three-DOF joystick with 400 dpi ($\approx$ 0.06 mm) position resolution (Fig. 6.9). The Falcon is capable of providing a maximum of approximately 9 N of force, via three armed motors, updated at 1 kHz [Novint Technologies, 2013]. In comparison,

6 Recommended retail price of $\approx$ GBP 160.00.



Figure 6.9: Placement of the Novint Falcon Haptic Controller on the ARTY wheelchair. The user provides an input by moving the ball grip, with standard axes shown. For driving the wheelchair, only the x and y axes are relevant.

the professional-grade ForceDimension Delta 6 controller provides 20 N of force with < 0.01 mm resolution, but costs approximately GBP 23,000.

We used two Falcon controllers[7], with the user unit (U) fixed on the smart wheelchair as shown in Fig. 6.9 and demonstrator unit (D) placed on a table marked by D in Fig. 6.10. Each controller was driven by a three-dimensional PD controller[8]:

$$\mathbf{u}_t^f = \mathbf{K}_p \mathbf{e}_t + \mathbf{K}_d \frac{d}{dt} \mathbf{e}_t \, , \tag{6.10}$$

where $\mathbf{u}_t^f$ is the control signal applied to the Falcon, $\mathbf{K}_p$ and $\mathbf{K}_d$ are the 3×3 position and derivative diagonal gain matrices, $\mathbf{e}_t = \mathbf{b}_t - \mathbf{c}_t$ is the error at time t, i.e., the difference between the desired ($\mathbf{b}_t$) and current ($\mathbf{c}_t$) positions. Hence, each PD controller was parameterised by the set $\theta^f = \{\mathbf{K}_p, \mathbf{K}_d\}$.

For LAD, the PD controllers operated in one of two modes: (S)tandard or (A)ssistance takeover with a corresponding parameter set for each controller, denoted θ_U^S, θ_U^A, θ_D^S and θ_D^A, where the subscript and superscript label the unit and mode respectively. These parameters were manually tuned and are shown in Table 6.2.

7 The two units used were donated by Novint Technologies.

8 We had initially used a PID controller but the integrative component would accumulate errors as the ball grip was consistently held in place to control the wheelchair, which led to large forces and instabilities.

Controller	Mode	$\mathbf{K}_p$ (x, y, z)	$\mathbf{K}_d$ (x, y, z)
User	Standard	(15, 5, 10)	(0.1, 0.1, 0.1)
	Assistance	(50, 50, 50)	(0.1, 0.1, 0.1)
Demonstrator	Standard	(30, 10, 20)	(0.1, 0.1, 0.1)
	Assistance	(15, 5, 10)	(0.1, 0.1, 0.1)

Table 6.2: PD parameters for controlling the user and demonstrator Falcon haptic devices. Note that the gain variables are diagonal matrices and as such, only the diagonal elements are shown.

During standard operation, the user drove the wheelchair without assistance; the desired state of the user Falcon controller was the center state $\mathbf{b}_t^U = (0, 0, 0)$. For the demonstrator controller, the desired state was the current state of the user's controller $\mathbf{b}_t^D = \mathbf{c}_t^U$. In other words, the demonstrator felt the user's controlling signals. During assistance, the signal flow was flipped and the user felt the demonstrator's control. More precisely, the demonstrator's Falcon had desired state $\mathbf{b}_t^D = (0, 0, 0)$ and the user's Falcon had desired state $\mathbf{b}_t^U = \mathbf{c}_t^D$. In preliminary tests, we experimented with bi-directional control during the assistance phase but this resulted in a coarse feel to the controller. We posit this was due to communication delays and future work would look into resolving this issue. The operating mode was controlled by the demonstrating assistance using the when-to-help h_t signal, toggled via a button press on the demonstrator Falcon grip.

6.4.1.5 *Limited Control with Haptic Feedback*

To simulate the turning control loss, we limited both the turning capability of the wheelchair as before and also set the user's Falcon to provide a greater counteracting force when the user attempted right turns. This was achieved by setting the y-axis component of the positional gain to a larger value $\mathbf{K}_{p,y} = 100$ when the ball grip position was negative. During assistance takeover, the original gain value was restored.

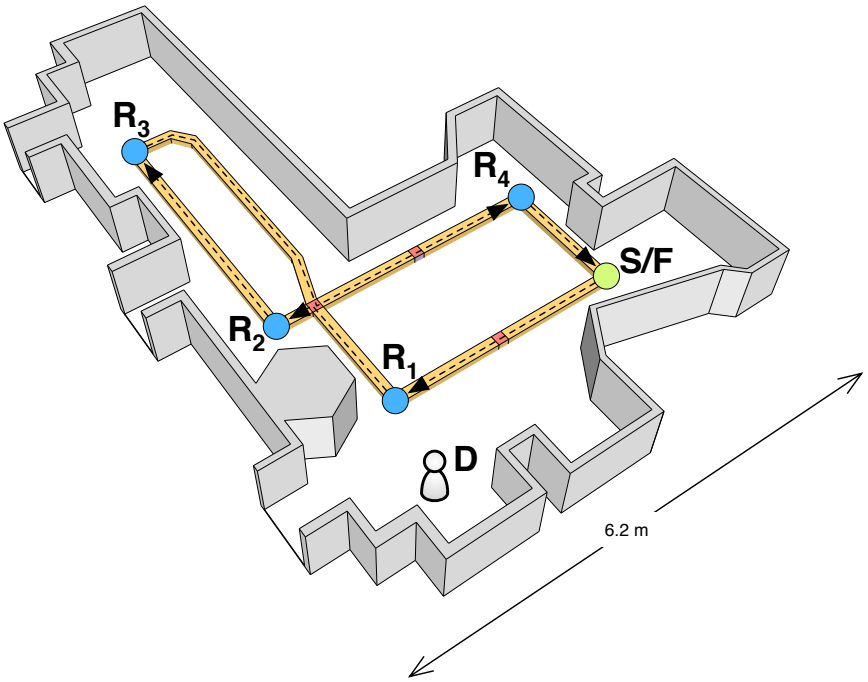


Figure 6.10: Driving course used in our haptic-control LAD experiment. Participants were tasked to drive from S to F, passing through and making four sharp right turns, labelled R_1 through R_4. The demonstrator was located at D.

6.4.2 *Experimental Setup*

Our experimental route was similar to the simulation setup but with four hard right turns instead of two (labelled as R_1 to R_4 in Fig. 6.10). We invited 15 participants (six female) ages 23-38 (mean: 27.3, sd: 4.22) to drive the wheelchair along the specified track (from S to F) a total of five times. Of the 15 participants, only four had previous experience using the Falcon haptic controller. As such, before beginning the trial, the participants drove around the experimental area and along the route until they felt confident about their driving ability.

During the first timed lap, full control was conferred, allowing us to obtain the baseline performance of an average user. The control limitation was applied during the latter four laps. Human demonstrator assistance was offered during the third lap, during which the model was simultaneously trained. For the last two laps, each participant was provided with human or model assistance, the ordering of which was randomised to account for driver improvement. Eight

Survey Questions:

1. I found navigating the obstacle course with the wheelchair easy.
2. I found navigating the obstacle course frustrating.
3. I performed well on the task.
4. The task was difficult at times.
5. I found the driving assistance to be helpful.
6. The driving assistance interfered with my driving in a negative way.
7. The driving assistance enabled me to complete the task faster.
8. I found the driving assistance to be timely.
9. The driving assistance negatively impacted my driving ability.
10. I would have completed the task easily even without the driving assistance.
11. I liked having driving assistance for this task.

Figure 6.11: Survey questions for the LAD Experiment. Participants were asked to indicate how much they agreed with the eleven statements on a 5-point Likert scale. For the first two trials where assistance was not given, surveys consisting of only questions 1-4 were given.

of the fifteen participants underwent the model assistance first. After each trial, the participants were given a survey (shown in Fig. 6.11) to complete indicating how much they agreed with eleven statements on a 5-point Likert scale.

6.4.3 *Results: Lap Performance and Survey Results*

The box-plot shown in Fig. 6.12 summarises the lap completion times under the different conditions. Similar to the simulation experiment, we observed that the control limitation hampered performance, doubling lap times to an average of 90.97 s. With demonstrator assistance, the participants were able to complete the task with performance similar to the baseline (average lap times of 43.30 s and 43.43 s for the learning and test trial respectively). Most importantly, participants using LAD-model assistance attained similar performance (average lap time of 47.49 s) to demonstrator assistance and the baseline, corresponding to a speedup of 191% over the limited control condition.

Fig. 6.13 illustrates the empirical distributions of when assistance was offered. We see that both human and model assistance distributions are highly similar; the model only provided assistance at the sharp right turns, marked R_1 to R_4,

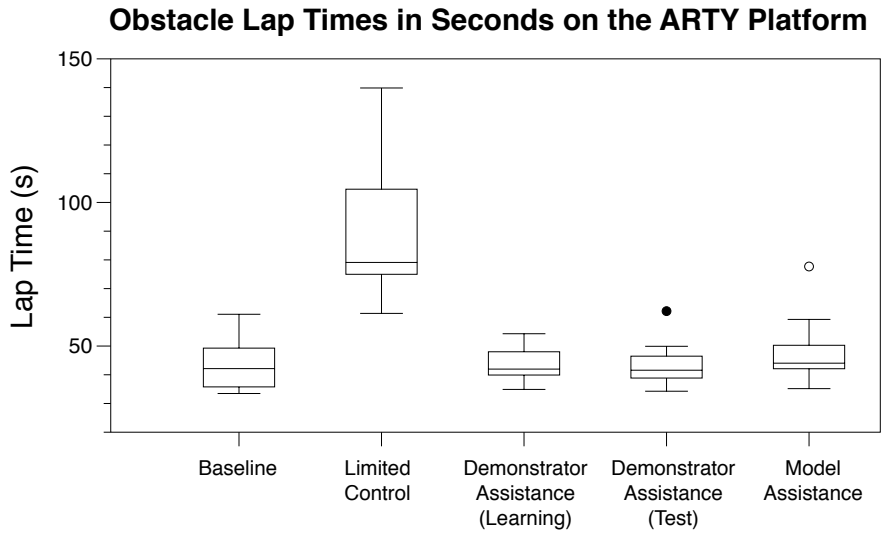
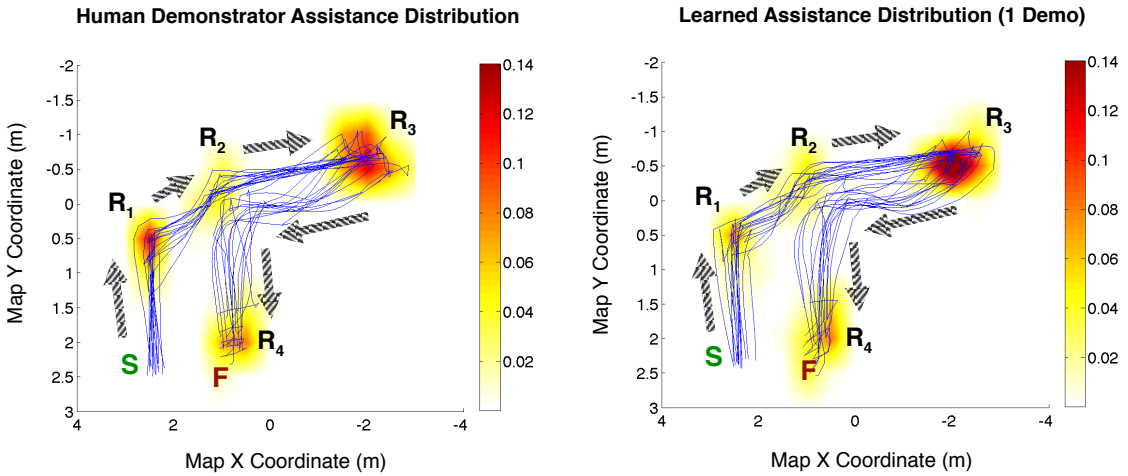


Figure 6.12: Lap times in seconds for all fifteen participants categorised by mode. The control limitation increased lap times from an average of 43.04 s (sd. 8.32 s) to 90.97 s (sd. 26.13 s). Demonstrator assistance enabled drivers to complete the task more quickly, reducing lap times back to the baseline, i.e., 43.30 s (sd. 5.64) and 43.43 s (sd. 6.89) for the learning trial and test trial respectively. The model assistance (only after a single demonstration) performed similarly to the demonstrator assistance, with slightly higher average lap times of 47.49 s (sd. 10.99).



(a) Demonstrator (b) Model (1-Demo)

Figure 6.13: Driven paths and normalised smoothed density plots illustrating when assistance was offered by the human demonstrator and LAD model. Both distributions are highly similar — assistance was only offered at the marked sharp right turn points — indicating the learned model was able to capture when to appropriately assist. In particular, note the lack of assistance during the right turns made between R_3 and R_4.

despite the user attempting right turns at other portions of the course (e.g., as indicated by the path driven between R_3 and R_4). These results show that the model provided *contextual* assistance, based not only on the user's desired movements, but also the environment (via the laser readings).

From the survey results for the first two laps (shown in Fig. 6.14), we observed that more participants found navigating the course more difficult and more frustrating (Q1 and Q2) during limited control as compared to the other conditions. Specifically, 33% of the participants disagreed (or strongly disagreed) with Q1 and 40% agreed that the task was frustrating under limited control. In addition, the participants considered they performed better on the task for the baseline (Q3) and felt the task was more difficult under limited control (Q4). To summarise, the control limitation was sufficient to make the task more challenging for our able-bodied participants. Notably, the response distributions for the assistance laps were comparable to the baseline, suggesting that the help offered alleviated the difficulty effected by the limited control.

Fig. 6.15 shows the responses for laps when assistance was offered. The response distributions between the human and LAD model assistance laps are similar, which suggests that to the participants, the model assistance performed comparably to the human demonstrator — using a χ^2 goodness-of-fit test, we were unable to reject the hypothesis that the distributions were the same for all questions (at the Bonferroni-corrected $\alpha = 0.05$ level[9]). Full p-values are available as Table G.3 in the Appendix.

Focussing on the responses during the model assistance laps, we found participants answered positively with regards to the assistance; 14 out of the 15 participants found the assistance to be helpful (Q5) and timely (Q8), enabling them to complete the task faster (Q7); the remaining one participant was neutral to both statements. All participants who drove with model assistance preferred help for the task (Q11)[10]. These responses were also largely consistent, with only

9 We note that Bonferroni-correction is conservative and future experiments with additional participants would provide firmer empirical evidence.

10 The one participant who disagreed with Q11 during the *human assistance* lap drove too close to the wall during the return leg from R_3 and R_4 and was momentarily immobile due to the obstacle avoidance method. Assistance was not offered since it was possible for the participant to resolve the situation.

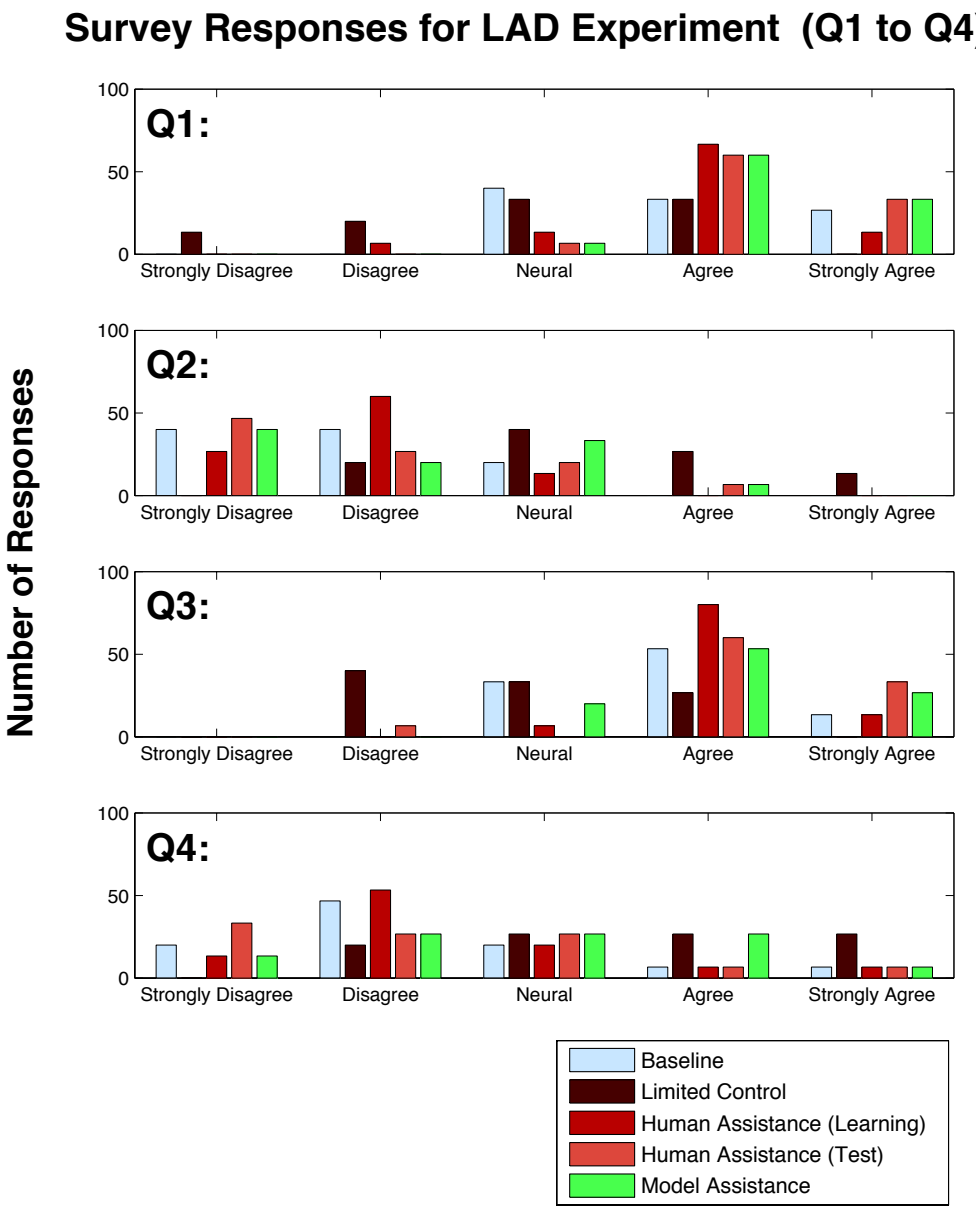


Figure 6.14: Survey Responses comparing the different lap conditions (Q1 to Q4). Participants found the course more difficult and frustrating under the limited control condition. Responses during assistance (both human and model) were similar to the baseline condition (without control limitation).

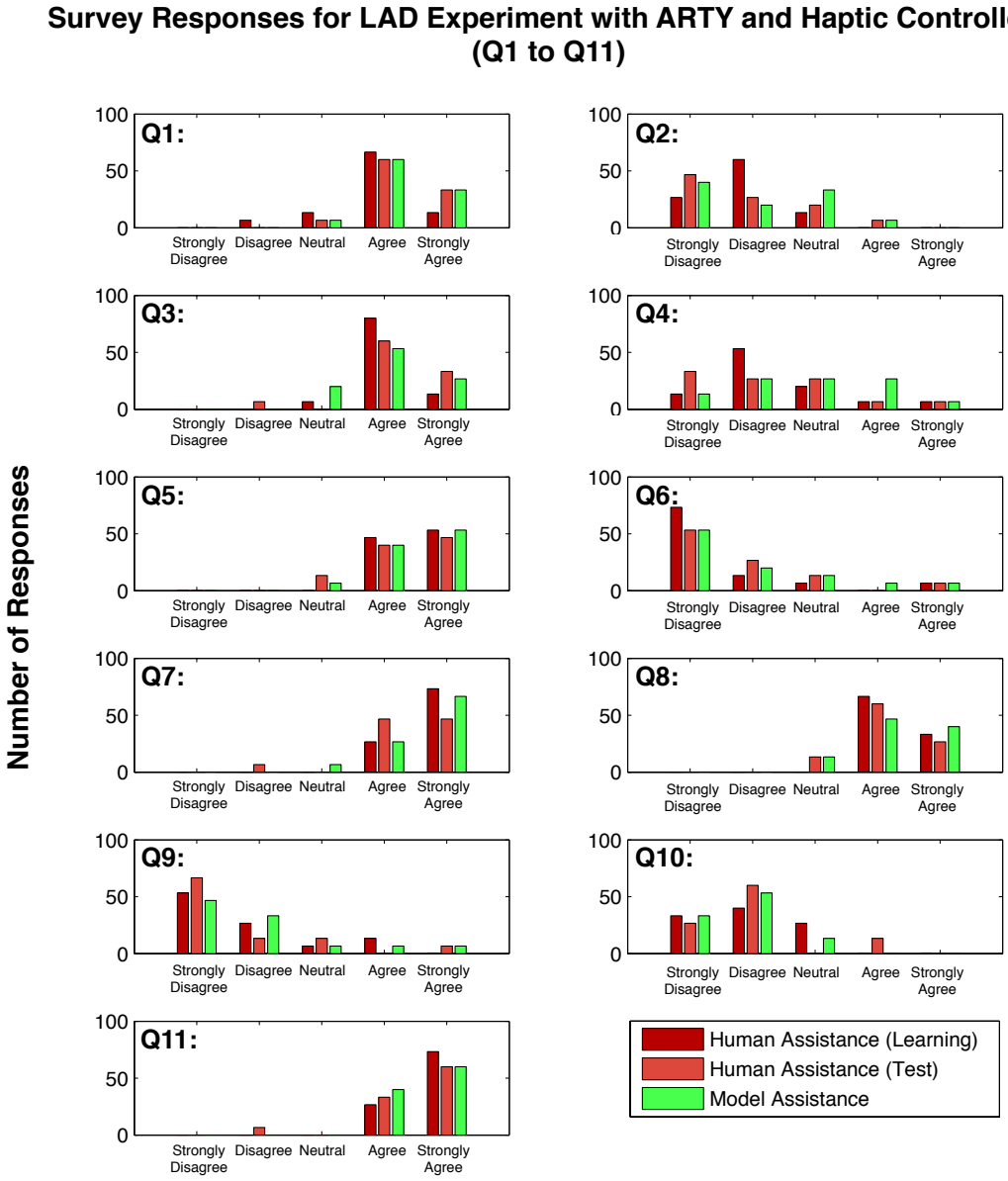


Figure 6.15: Survey Responses from 15 participants for Experiment with Haptic Controllers comparing the two laps with human assistance (both demonstration/learning and test laps) and the LAD model assistance. The distributions are similar indicating that participants felt they were appropriately helped in all three laps (irrespective of whether it was human or model-based assistance).

a minority (2 participants[11]) agreeing that the assistance negatively affected in their task (Q6 and Q9). In summary, participants not only performed better quantitatively with assistance (with lower lap completion times), but also felt that the human/model assistance helped them complete the task.

6.5 CONCLUSIONS

This chapter proposed and discussed Learning Assistance by Demonstration (LAD); a LbD approach to the problem of deriving assistance policies. In addition, we have contributed a novel probabilistic model which captures both when and how to assist, and efficient OIESGP methods for *online* training.

Both simulation and real-world experiments demonstrate the efficacy of our approach; our LAD-enabled smart wheelchair is able to learn rapidly to provide contextual assistance where needed, speeding up lap times by almost double (speed-up of 191% after only one demonstration). Survey responses by 15 human participants support the notion that the LAD-model performed similarly to the human demonstrator and in general, participants appreciated the assistance given, enabling them complete the task faster.

We believe LAD to be fertile ground for future research. From an experimental perspective, it is important to validate the model on more complex scenarios with the intended target population. In addition, while fully capable of providing haptic feedback, we observed that the force capabilities of the Falcon were not sufficient to overcome the forces issued by our able-bodied participants. As such, future studies may look into using higher-grade haptic devices. Moreover, it would also be interesting to apply our model to other robotic platforms such as the humanoid iCub, which would enable assistance with object manipulation tasks. Finally, we would like to highlight interesting research questions particular to the LAD approach:

11 The two participants who provided inconsistent responses were not native English speakers and may have misunderstood the statement.

- **How can we train the model efficiently when the assistive signal is latent?** In this work, we have assumed that h_t is visible. In certain scenarios, h_t is hidden or latent, complicating the training process.

- **What if there are multiple assistants and multiple users?** For simplicity, we have limited our model description to a single user and single assistant. However, this can be extended to teams of assistants and users.

- **How am I (the robot) being helped?** In this paper, we have assumed that the robot is observing the demonstrator assisting another human. Can our model be combined with a task-oriented LbD approach such that the robot learns how-to-assist from observing how the demonstrator corrects the mistakes it makes at a given task?

- **How can the model adapt to improvements on the part of the assisted user?** The simplest, albeit less scalable, solution is simply for the demonstrator to provide additional demonstrations. A potentially more interesting solution may come from combining ideas from human-robot cross-training for collaborative robots [Nikolaidis and Shah, 2013] with our approach.

- **Can we extract more than control from the learnt assistive policies?** In this work, we have illustrated how LAD can be applied to derive assistive control policies. However, the learnt policy can also be used for studying how humans assist under various circumstances, which may supplement recent studies [Dragan and Srinivasa, 2012].

In the final following chapter, we conclude this thesis with a summary of our key contributions and explore potential solutions to several of these issues.

Part IV

CLOSING

Over the course of the six preceding chapters, we presented the OESGP and its infinite variant, the OIESGP, along with simulation and real-world robotics experiments. In this final chapter, we review the main contributions of this thesis and point out avenues for future work.

7

SUMMARY, CONCLUSIONS AND FUTURE WORK

Our main goal in this thesis was to address online-learning with multivariate temporal data-streams for robotics systems. To this end, we contributed the OESGP and OIESGP; combinations of the sparse online Gaussian process with reservoir-inspired kernels. Both are fixed-budget methods (with maximum computational and storage costs limited by a user-defined parameter) and deliver probabilistic distributions conveying uncertainties (instead of point predictions). Extensive experiments on benchmark dynamical systems and real-world robotics problems demonstrate that they achieved high accuracies relative to current state-of-the-art methods.

7.1 SUMMARY OF CONTRIBUTIONS

We began with the OESGP, which used finite echo-state reservoirs coupled with a sparse online GP. Although this method attained higher accuracy than its online ESN counterparts, it required the creation and maintenance of the reservoir. This in turn required tedious, manual-tuning of parameters such as the spectral-radius and reservoir connectivity.

As a solution to this problem, we considered the use of recursive kernels which were formulated by considering reservoirs of infinite size. Specifically, we proposed a new recursive kernel with automatic relevance determination that enables feature selection and weighting. Through stochastic natural gradient descent, the kernel hyperparameters are learnt in an online manner as

more data becomes available — this makes offline hyperparameter tuning unnecessary as long as a reasonable starting position is given. Furthermore, we demonstrated that inspection of the hyperparameters revealed aspects of the underlying system such as the importance of features and the relevancy of the past. Experimental results on a diverse set of benchmarks showed that the OIESGP achieved better performance than competing state-of-the-art methods, particularly on problems with irrelevant dimension.

We then turned our attention towards robotics applications. Our first OIESGP-based contributions were online generative and discriminative tactile classifiers capable of refining existing models of known objects and creating new models as needed. A classification experiment involving nine everyday objects with the iCub humanoid robot showed that our methods (with only a single pass through the data) attained perfect accuracy on the test portion of the dataset. Moreover, our methods surpassed the performance of human subjects tasked to determine the fullness of plastic bottles. This work provides evidence that the OIESGP can be readily applied to real-world signals and also that online-learning of objects through tactile feedback can be performed efficiently.

Second, we developed the Learning Assistance by Demonstration (LAD) framework — a novel approach for developing assistive controllers. Unlike current robotic learning by demonstration, this paradigm places importance on the assistive aspect of tasks by directly extracting an assistive policy. In practical terms, this allows robots to learn by watching assistants (e.g., scrub nurses, occupational therapists) as they perform their daily activities. To learn these policies, we contributed an OIESGP-based mixture model that addresses the difficult problems of "when" and "how" to assist and demonstrated its efficacy with experiments with human subjects using the ARTY smart wheelchair; we obtained positive performance and survey responses indicating that model assistance was comparable to human assistance on a mobility assistance task.

7.2 TOWARDS THE FUTURE

The work presented in this thesis is a step towards adaptive robots that learn and act in noisy, unconstrained environments. In particular, we have developed and empirically tested — using both simulation and real-world experiments — temporal learners that can be integrated into higher level structures (such as Bayes classifiers and mixture models). In this section, we discuss possible extensions and unresolved issues.

7.2.1 *Changing Distributions and Non-stationarity*

An important issue not fully addressed in this work is that of non-stationarity — where the underlying joint probability distribution changes over time. For example, robot motors wear down with use and capacitive tactile sensors experience drift. This leads to changes in the underlying target function that we aim to learn.

Although the OESGP/OIESGP can be applied to learning non-stationary time-series (since SNGD with proper learning rates partially alleviates the problem), using an out-dated basis vector set may restrain adaptation to change. One potential solution is to incorporate an *ageing* component into the scoring function for BV removal:

$$\epsilon^{a}_{i,t} = (1 - w_a)\epsilon_i + w_a \zeta_{i,t} \tag{7.1}$$

where ϵ_i is the original scoring value (Eq. 3.13), w_a is a weighting parameter that controls the contribution from each score component and $\zeta_{i,t}$ computes the "age" of the i-th BV at a given time t:

$$\zeta_{i,t} = \sum_{j=t_i}^{t} \omega_j \frac{k(\mathbf{b}_i, \mathbf{x}_j)}{k(\mathbf{x}_j, \mathbf{x}_j)}. \tag{7.2}$$

Here, ω_j is a weighting parameter, t_i was the time the i-th BV $\mathbf{b}_i$ was inserted into the BV set. As such, $\zeta_{i,t}$ is the weighted average of the normalised kernel values over time. Preliminary experiments show promising results: with proper weighting parameters, adaptability to changing distributions is improved (in terms of error reduction speed). However, this comes at a cost of higher errors on stationary distributions and future work is needed to examine trade-offs.

Another potential solution that can be easily assimilated into our methods was proposed in [Van Vaerenbergh et al., 2012]; the principal idea is to use a *forgetting factor* to force the posterior to lose information during updates. This approach was shown to be effective on two non-stationary toy problems and a real-world attitude control system. Hence, future work may involve comparing or combining the two aforementioned techniques, possibly with "freely-moving" basis vectors described in the next section.

7.2.2 *Unconstrained Basis Vectors*

In this work, we have worked solely with basis vectors centred upon actual observations. However, the BVs — also called *inducing inputs* [Quiñonero Candela and Rasmussen, 2005] — can be considered as parameters to be freely optimised. This approach was shown in [Snelson and Ghahramani, 2006] to produce high accuracies (compared to observation-centered BVs) when using only a small number of inducing inputs.

To learn the inducing inputs iteratively, we can apply an online optimisation algorithm such as SNGD. However, the associated computational cost is $O(ds_{\mathcal{B}}^3)$ per update where d is the input dimension and $s_{\mathcal{B}}$ is the number of inducing inputs. For large d (multi-dimensional with a long history), this method can be prohibitively expensive. Instead, a way forward may lie in a very recent GP for "big data" employing stochastic variational inference, which has demonstrated promising results both in terms of accuracy and computational cost [Hensman and Lawrence, 2012; Hensman et al., 2013]. Combining the variational method

with our recursive ARD kernel may yield a faster, more adaptive online learning method for spatio-temporal data streams.

7.2.3 *Noisy Inputs, Iterated Predictions and Heteroscedasticity*

Although our experiments show that our methods can perform well even when inputs are noisy, GPs in their theoretical formulation assume *noiseless inputs*. This has consequences for both learning and prediction. Learning with inputs contaminated with Gaussian noise and a squared exponential kernel leads to a "corrected" kernel with increased lengthscales [Girard, 2004]. Intuitively, the same outcome should hold for our recursive ARD kernel but this has yet to be rigorously proven.

In addition, one obtains underestimated predictive variances when performing iterated one-step predictions (using the previous predicted output as the next input) because input noise is not taken into account. Batch GP models that address this issue (for Gaussian noise) are described in [Girard et al., 2002] and [McHutchon and Rasmussen, 2011], but finding efficient online variants remains an open problem. One potential way forward is via a Monte-Carlo approach, such as the well-known particle-filter algorithm, but this incurs additional computational cost.

A related problem is that of *heteroscedasticity*, i.e., the output noise is local and input-dependent. The GP models considered in this thesis assume uniform finite variance throughout the domain. A popular method of resolving this issue is to use another GP to capture the variances [Goldberg et al., 1997; Titsias and Lázaro-gredilla, 2011]. This strategy can be applied in an online setting by constructing a second OIESGP using an intermediate exponential function $\exp(\mathsf{f}(\mathbf{x}))$ to estimate the noise. Training can be performed in-tandem with the primary OIESGP. Our preliminary experiments with this heteroscedastic OIESGP have thus far provided positive outcomes, but more experimental work is necessary to fully evaluate the approach.

7.2.4 *Complex Hierarchical Architectures*

In this thesis, we have shown how the OIESGP was an effective building block in a Bayes classifier and a mixture of experts model. A fruitful line of enquiry would be to investigate OIESGP as a base-model in more complex architectures. In particular, using the OIESGP as a "computational unit" in the biologically-inspired Hierarchical, Attentive, Multiple Models for Execution and Recognition (HAMMER) architecture [Demiris and Khadhouri, 2006] would result in an adaptive model integrating attention for imitation and control of robots. Here, each OIESGP would function in forward-inverse model pairings capable of both action recognition and execution.

Another interesting avenue comes from the field of "deep" models [Bengio, 2009]. Potentially, one could extend the mixture model introduced in Chapter 6 to construct "layers" in a unsupervised fashion. The layers would correspond to a hierarchy of concepts, whereby higher level notions are constructed from lower level ones. If the layers can be linked in a suitably efficient manner, the resulting Hierarchical Mixture of OIESGP models (HM-OIESGP) model would be capable of capturing very long-term spatio-temporal correlations.

7.2.5 *Large-scale Learning of Objects via Touch and Other Modalities*

Chapter 5 presented an OIESGP-based classifier for recognising objects with tactile feedback. Although effective, the model is composed of one expert model per object. This implies a significant computational and storage overhead when dealing in real-world situations with thousands, if not millions, of different objects. How we might scale up our model to such scenarios?

A prospective solution was introduced in the previous section, i.e., hierarchical modelling. Through a layered model, we may identify re-usable "tactile feature elements" — common multivariate time-series snippets — that can be used in higher layers. Although this approach may not reduce the number of OIESGPs required, it could drastically reduce the complexity of each model, thus

allowing for very small BV sets. Viewed in a different way, the HM-OIESGP may better compress available information and in doing so, acquire processing and storage savings. In addition, the HM-OIESGP may simplify the incorporation of other input modalities (e.g., vision and sound) with different timescales, such as sensory data arising from autonomous exploration and learning [D. Schiebener and Ude, 2013].

When scaling-up the model, it is important to keep the required number of training samples manageable. As previously proposed in Chapter 5, active learning [Settles, 2010] may be used to better choose objects to learn from. In particular, the uncertainty estimates produced by our model may be used in an active class selection scheme [Lomasky et al., 2007] to determine target objects expected to cause maximal classifier improvement.

7.2.6 *Learning Complex Assistive Policies*

We believe the LAD approach discussed in Chapter 6 to be a rich and promising area for future exploration. Our work is only the first step in this direction and there remain many interesting, open questions.

Certainly, additional experiments are needed to resolve outstanding issues such as model generalisability (for example, to test if policies learned in one location extend to other environments). Moreover, the robot's physical embodiment and observations may be different to that of the human demonstrator. To derive a proper policy, a *correspondence mapping* between the robot and demonstrator needs to be identified [Argall et al., 2009]. These issues are common to the larger class of LbD methods and are actively investigated within the research community.

Focussing on elements particular to LAD, here we have considered the (simpler) assistance scenario where the training signal h_t is visible throughout the demonstration process. Training the model when h_t is latent is decidedly more complicated. One solution is to rely on proxy observables and estimate the probability of assistance. Then, the How-to-Help model can be trained via a

sampling process or by weighting the training instances with the (estimated) probability that assistance is offered. However, how to perform these updates efficiently remains an unanswered question.

A core issue with assistance is that the user and assistant are likely to change and develop over time. For example, our wheelchair driver would hopefully improve as he gains experience. Moreover, his occupational therapist may also learn when and how to better help. These issues relate back to the subject of non-stationarity discussed in Section 7.2.1 and the corresponding resolutions may be applied. That said, more experimental work is needed to explore co-development issues.

Finally, interactions with end-users — the special-needs population and their care-givers — will help keep us on track towards developing assistive robots that will *help real people*. It is essential to move this research beyond laboratory conditions into real-world environments such as hospitals and rehabilitation centres. LAD can be an essential piece of this endeavour: the learnt policy, used for control in our experiments, can also be used for studying how humans assist under various circumstances, which may lead to interesting findings applicable to the general design of assistive robots.

Part V

APPENDIX

A

MATHEMATICAL FORMULAE

A.1 MATRIX INVERSION LEMMA

The Matrix Inversion Lemma — also known as Sherman-Morrison-Woodbury formula or Woodbury Formula — relates a rank-k correction to a matrix to a rank-k correction of the matrix's inverse. The formulae presentation below comes from Chapter 4 of [Murphy, 2012]. Suppose we have a partitioned matrix:

$$\mathbf{M} = \begin{pmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{pmatrix} \tag{A.1}$$

where $\mathbf{E}$ and $\mathbf{F}$ are invertible. Then,

$$(\mathbf{E} - \mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1} = \mathbf{E}^{-1}\mathbf{F}(\mathbf{H} - \mathbf{G}\mathbf{E}^{-1}\mathbf{F})^{-1}\mathbf{G}\mathbf{E}^{-1} \tag{A.2}$$

$$(\mathbf{E} - \mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1}\mathbf{F}\mathbf{H} = \mathbf{E}^{-1}\mathbf{F}(\mathbf{H} - \mathbf{G}\mathbf{E}^{-1}\mathbf{F})^{-1}. \tag{A.3}$$

For the special case of a rank-one update (e.g., in the sparse online GP approximations), we have the Sherman-Morrison formula:

$$(\mathbf{E} + \mathbf{u}\mathbf{v}^{\top})^{-1} = \mathbf{E}^{-1} - \frac{\mathbf{E}^{-1}\mathbf{u}\mathbf{v}^{\top}\mathbf{E}^{-1}}{1 + \mathbf{v}^{\top}\mathbf{E}^{-1}\mathbf{u}} \tag{A.4}$$

B

RECURSIVE ARD KERNEL DERIVATION

In this appendix, we present a derivation of the recursive ARD kernel introduced in Chapter 4. Our approach generalises the derivation for the recursive squared exponential derivation [Hermans and Schrauwen, 2011] to allow for varying spatial lengthscales.

To begin, let $\mathbf{u} = [\mathbf{u}_1|\mathbf{u}_2]$ and $\mathbf{v} = [\mathbf{v}_1|\mathbf{v}_2]$ be concatenations of two vectors each. Then for kernels of the form,

$$k(\mathbf{u}, \mathbf{v}) = f((\mathbf{u} - \mathbf{v})\mathbf{M}(\mathbf{u} - \mathbf{v})) \tag{B.1}$$

where $\mathbf{M}$ is a diagonal matrix, $\mathbf{M} = \text{diag}(\mathbf{l})^{-2}$ with $\mathbf{l} = [l_i]_{i=1}^{d}$, we can separate out the kernel into two parts:

$$k(\mathbf{u}, \mathbf{v}) = f((\mathbf{u}_1 - \mathbf{v}_1)\mathbf{M}_1(\mathbf{u}_1 - \mathbf{v}_1) + (\mathbf{u}_2 - \mathbf{v}_2)\mathbf{M}_2(\mathbf{u}_2 - \mathbf{v}_2)) \tag{B.2}$$

where $\mathbf{M}_1 = \text{diag}(\mathbf{l}_1)^{-2}$ with $\mathbf{l}_1 = [l_i]_{i=1}^{d_1}$ and $\mathbf{M}_2 = \text{diag}(\mathbf{l}_2)^{-2}$ with $\mathbf{l}_2 = [l_i]_{i=d_1+1}^{d_2}$. Let $l_{d+1} = l_{d+2} = \cdots = l_d = \sigma_\rho/\sqrt{2}$, i.e., the same lengthscale is used for all elements of the second portion, and we specify $f(z) = \exp(-z/2)$. Then:

$$k(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{1}{2}(\mathbf{u}_1 - \mathbf{v}_1)\mathbf{M}_1(\mathbf{u}_1 - \mathbf{v}_1)\right) \exp\left(-\frac{1}{\sigma_\rho^2}(\mathbf{u}_2 - \mathbf{v}_2)^2\right) \tag{B.3}$$

Now, suppose that $\mathbf{u}_1$ and $\mathbf{v}_1$ correspond to the current inputs $\mathbf{x}_t$ and $\mathbf{x}'_t$ respectively, and $\mathbf{u}_2$ and $\mathbf{v}_2$ correspond to the recursive maps for each time series. Then,

$$\begin{aligned}\kappa_t(\mathbf{x},\mathbf{x}') = &\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')\mathbf{M}_1(\mathbf{x}-\mathbf{x}')\right)\\ &\exp\left(-\frac{\kappa_{t-1}(\mathbf{x},\mathbf{x})+\kappa_{t-1}(\mathbf{x}',\mathbf{x}')-2\kappa_{t-1}(\mathbf{x},\mathbf{x}')}{\sigma_\rho^2}\right)\end{aligned} \tag{B.4}$$

where we have used the property of Mercer kernels that dot products between feature maps are equivalent to a kernel function evaluation. Since $\kappa_{t-1}(\mathbf{x},\mathbf{x}) = 1$ for $\exp(-z/2)$, we can simplify the above to yield:

$$\kappa_t(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')\mathbf{M}_1(\mathbf{x}-\mathbf{x}')\right)\exp\left(-\frac{\kappa_{t-1}(\mathbf{x},\mathbf{x}')-1}{\sigma_\rho^2}\right) \tag{B.5}$$

which is the form presented in Chapter 4. Note that this construction can be easily extended to the case where $\mathbf{M}_1$ is the factor analysis distance: $\mathbf{M}_1 = \Lambda\Lambda^\top + \text{diag}(\mathbf{l})^{-2}$.

C

PARAMETERS AND GRID-SEARCH BOUNDS FOR THE EXPERIMENTAL STUDIES

This appendix chapter describes the parameter setup used for the benchmark experiments used in Chapter 4. For each memory structure (delay-line or reservoir) and algorithm, we ran a grid-search over parameter combinations. For each combination, the set with minimum RMSE error over 5 runs with sequences of 5000 time-steps (80% training, 20% testing) was selected as the best parameter set and was used for subsequent tests.

C.1 DELAY-LINE AND RESERVOIR MEMORY STRUCTURES

We used standard delay lines of length $1, 5, 10$ and 25. We used a standard reservoir setup: 100 neurons with the hyperbolic tangent activation function. The reservoir weights and connectivities were set at 0.1 and 0.1 respectively. The input weights and spectral radius were varied using the parameter sets $\{0.1, 0.5, 1.0\}$ and $\{0.8, 0.9, 0.99\}$ respectively. For the *sign* of the input weights, we employed the deterministic technique described by [Rodan and Tino, 2011] to ensure *aperiodic* patterns, which was shown empirically to yield better performance. In particular, the input signs are determined by a decimal expansion of π where the individial decimal digits were thresholded at 4.5; if the decimal digit $d_n \geqslant 5$, then the input sign was positive (and negative otherwise).

Algorithm	Memory Structure	Parameter	Value Set
OIESGP	Delay-line	Capacity (C)	100
		Recursion Depth (τ)	$\{5, 10, 25\}$
		Spatial Lengthscale (l)	$\{1, 5, 10\}$
		Temporal lengthscale (σ_ρ^2)	$\{1.01, 1.5\}$
		Signal Variance (σ_f^2)	1.0
		Novelty Threshold (ϵ)	10^{-5}
OESGP	Reservoir	Capacity (C)	100
		Lengthscale (l)	$\{1, 5, 10\}$
		Signal Variance (σ_f^2)	1.0
		Novelty Threshold (ϵ)	10^{-5}
SOGP	Delay-line	Capacity (C)	100
		Lengthscale (l)	$\{1, 5, 10\}$
		Signal Variance (σ_f^2)	1.0
		Novelty Threshold (ϵ)	10^{-5}
LWPR	Delay-line	Kernel	Gaussian
		Distance metric (D)	$\{0.5, 5.0\}$
		Distance metric learning rate (α)	$\{1, 5\}$
		Meta learning rate	250
		Diagonal Distance Metric	{ True, False }
KLMS	Delay-line	Capacity (C)	100
		Lengthscale (l)	$\{1, 5, 10\}$
		Learning Rate (η) (σ_f^2)	$\{0.01, 0.2, 0.5\}$
NORMA	Delay-line	Capacity (C)	100
		Lengthscale (l)	$\{1, 5, 10\}$
		Learning Rate (η) (σ_f^2)	$\{0.01, 0.2, 0.5\}$
		Regularisation (ϵ)	$\{10^{-2}, 10^{-1}\}$

Table C.1: Algorithm parameter sets for the benchmark experiments.

C.2 ALGORITHM PARAMETERS

As discussed in Chapter 4, we included in our benchmark experiments five different methods: OIESGP, OESGP, SOGP, LWPR, KLMS and NORMA. The parameter sets searched over for each of the methods are shown in Table C.1. For the kernel-based methods (OESGP, SOGP, KLMS and NORMA), we used the squared exponential kernel. In addition, we included the ARD kernel for SOGP. The kernel hyperparameters for the GPs were optimised via SNGD (Section 4.2, Chapter 4) with gradient sampling interval $s_g = 20$ and stopping criteria $||\triangle\theta||^2 < 10^{-4}$ over 50 iterations. The initial noise parameter set to the true noise value of the benchmark problem.

D

SUPPLEMENTARY BENCHMARK RESULTS

This appendix collates supplementary results for the benchmark tests performed in Chapter 4, in particular:

- Table D.1 Wilcoxon rank sum test p-values for the RMSE distributions on the One-step Prediction tasks.
- Table D.2 Wilcoxon rank sum test p-values for the RMSE distributions on the One-step Prediction tasks with Irrelevant Dimensions.
- Table D.3 Wilcoxon rank sum test p-values for the RMSE distributions on the System Identification Tasks.
- Table D.4 Wilcoxon rank sum test p-values for the RMSE distributions on the System Identification Tasks with Irrelevant Dimensions.
- Table D.5 Training and prediction CPU time measured in microseconds for benchmarks.
- Table D.6 Training and prediction CPU time measured in microseconds for the irrelevant dimension prediction task.

Mackey-Glass	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	1.313 e-15	1.851 e-13	5.464 e-01	4.060 e-14	1.349 e-16	$<$ 5 e-18
OIESGP	-	-	1.568 e-05	3.959 e-12	1.013 e-17	7.969 e-18	$<$ 5 e-18
SOGP	-	-	-	8.013 e-08	1.952 e-17	1.142 e-17	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	4.095 e-02	2.293 e-06	$<$ 5 e-18
LWPR	-	-	-	-	-	3.692 e-07	$<$ 5 e-18
KLMS	-	-	-	-	-	-	9.540 e-18

Henon	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	3.961 e-02	1.669 e-13	7.394 e-04	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	3.456 e-10	8.295 e-02	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	1.040 e-05	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
LWPR	-	-	-	-	-	$<$ 5 e-18	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Laser	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	1.299 e-05	$<$ 5 e-18	5.897 e-07	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	$<$ 5 e-18	1.842 e-11	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	$<$ 5 e-18	2.071 e-17	6.398 e-10	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	2.422 e-15	1.260 e-11	$<$ 5 e-18
LWPR	-	-	-	-	-	6.319 e-02	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Ikeda	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	1.349 e-16	2.521 e-13	1.349 e-16	3.194 e-15	1.349 e-16	$<$ 5 e-18
OIESGP	-	-	1.120 e-08	4.038 e-05	7.969 e-18	8.484 e-17	$<$ 5 e-18
SOGP	-	-	-	5.053 e-04	1.899 e-03	8.013 e-01	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	1.142 e-17	4.207 e-15	$<$ 5 e-18
LWPR	-	-	-	-	-	2.691 e-16	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Lorenz	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	1.020 e-03	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	5.418 e-01	$<$ 5 e-18	7.504 e-18	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
LWPR	-	-	-	-	-	$<$ 5 e-18	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

NP1	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	7.695 e-01	2.275 e-13	$<$ 5 e-18	1.013 e-17	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	5.050 e-12	$<$ 5 e-18	1.839 e-17	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	6.885 e-15	$<$ 5 e-18	4.960 e-15	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	$<$ 5 e-18	3.647 e-01	$<$ 5 e-18
LWPR	-	-	-	-	-	$<$ 5 e-18	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Table D.1: Wilcoxon rank sum test p-values for the RMSE distributions on the One-step Prediction tasks.

Mackey-Glass	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	3.242 e-04	1.075 e-17	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	2.954 e-17	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	3.995 e-13	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	9.919 e-16	$<$ 5 e-18	$<$ 5 e-18
LWPR	-	-	-	-	-	1.358 e-04	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Henon	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	1.075 e-17	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	3.966 e-17	$<$ 5 e-18
OIESGP	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	1.500 e-03	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	3.566 e-15	1.287 e-17	8.985 e-18
LWPR	-	-	-	-	-	$<$ 5 e-18	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Laser	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	2.647 e-10	3.791 e-16	6.880 e-14	1.134 e-16	5.050 e-12	$<$ 5 e-18
OIESGP	-	-	4.178 e-09	8.188 e-12	5.565 e-12	7.803 e-12	1.349 e-16
SOGP	-	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	1.510 e-02	1.943 e-03	$<$ 5 e-18
LWPR	-	-	-	-	-	1.894 e-12	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Ikeda	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	$<$ 5 e-18	$<$ 5 e-18	7.504 e-18	7.504 e-18	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	1.633 e-17	$<$ 5 e-18
SOGP	-	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	8.985 e-01	9.540 e-18	$<$ 5 e-18
LWPR	-	-	-	-	-	1.539 e-17	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Lorenz	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	$<$ 5 e-18	2.676 e-12	1.691 e-02	$<$ 5 e-18	4.540 e-09	$<$ 5 e-18
OIESGP	-	-	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	2.990 e-09	$<$ 5 e-18	3.022 e-15	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	$<$ 5 e-18	2.225 e-11	$<$ 5 e-18
LWPR	-	-	-	-	-	$<$ 5 e-18	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

NP1	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	1.676 e-11	9.917 e-13	2.867 e-09	$<$ 5 e-18	5.556 e-01	$<$ 5 e-18
OIESGP	-	-	2.860 e-15	9.543 e-15	$<$ 5 e-18	2.444 e-11	$<$ 5 e-18
SOGP	-	-	-	6.043 e-04	$<$ 5 e-18	3.177 e-05	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	1.700 e-16	4.216 e-04	$<$ 5 e-18
LWPR	-	-	-	-	-	1.161 e-13	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Table D.2: Wilcoxon rank sum test p-values for the RMSE distributions on the One-step Prediction tasks with Irrelevant Dimension.

NP2	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	1.500 e-03	2.398 e-08	3.252 e-09	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	2.054 e-04	1.743 e-04	7.504 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	8.767 e-01	1.366 e-17	$<$ 5 e-18	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	4.206 e-17	$<$ 5 e-18	$<$ 5 e-18
LWPR	-	-	-	-	-	1.638 e-14	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

NARMA-10	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	3.162 e-10	3.238 e-11	8.462 e-18	7.504 e-18	8.462 e-18	$<$ 5 e-18
OIESGP	-	-	5.147 e-01	1.585 e-13	1.669 e-13	8.053 e-14	$<$ 5 e-18
SOGP	-	-	-	1.273 e-16	1.273 e-16	7.551 e-17	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	5.510 e-01	2.059 e-01	$<$ 5 e-18
LWPR	-	-	-	-	-	8.419 e-02	$<$ 5 e-18
KLMS	-	-	-	-	-	-	$<$ 5 e-18

Table D.3: Wilcoxon rank sum test p-values for the RMSE distributions on the System Identification Tasks.

NP2	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	2.921 e-08	1.313 e-15	1.288 e-13	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
OIESGP	-	-	2.051 e-15	2.260 e-14	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	5.065 e-02	1.013 e-17	1.075 e-17	$<$ 5 e-18
$SOGP_{ARD}$	-	-	-	-	9.540 e-18	1.075 e-17	$<$ 5 e-18
LWPR	-	-	-	-	-	1.769 e-07	$<$ 5 e-18
KLMS	-	-	-	-	-	-	4.487 e-11

NARMA-10	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
OESGP	-	4.730 e-17	1.691 e-02	3.113 e-14	$<$ 5 e-18	1.733 e-17	$<$ 5 e-18
OIESGP	-	-	8.171 e-11	1.427 e-08	$<$ 5 e-18	$<$ 5 e-18	$<$ 5 e-18
SOGP	-	-	-	1.332 e-03	1.879 e-09	1.546 e-08	9.530 e-17
$SOGP_{ARD}$	-	-	-	-	$<$ 5 e-18	7.969 e-18	$<$ 5 e-18
LWPR	-	-	-	-	-	7.711 e-03	9.540 e-18
KLMS	-	-	-	-	-	-	2.952 e-12

Table D.4: Wilcoxon rank sum test p-values for the RMSE distributions on the System Identification Tasks with Irrelevant Dimensions.

Problem	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
Mackey-Glass	2.8998 (0.3943)	1.4550 (1.4826)	2.5536 (0.4772)	20.1325 (18.3434)	**0.8014** (0.1237)	3.1437 (1.8318)	1.6689 (0.1987)
	0.8448 (0.1196)	0.6955 (0.1850)	0.6088 (0.1499)	0.7119 (0.1145)	**0.3553** (0.0482)	2.1368 (0.2423)	0.6586 (0.0961)
Henon	2.7610 (0.5375)	2.6521 (1.1923)	2.6570 (1.9692)	4.8350 (3.2697)	15.3122 (6.1677)	1.9464 (0.5519)	**1.5067** (0.3399)
	0.8221 (0.1094)	0.6763 (0.1214)	0.5709 (0.1282)	**0.5464** (0.1786)	4.2299 (1.6268)	1.1501 (0.1533)	0.6011 (0.1566)
Laser	2.1810 (0.3957)	2.2189 (0.4335)	**0.8590** (0.2434)	2.5686 (5.9253)	4.1829 (1.2348)	3.1430 (1.8486)	1.4433 (0.4452)
	0.6877 (0.1247)	0.6352 (0.1661)	**0.4749** (0.1400)	0.5846 (0.1597)	1.2863 (0.3601)	2.1665 (0.3062)	0.5601 (0.2106)
Ikeda	2.8714 (0.2640)	2.6669 (0.3028)	2.0365 (0.5481)	3.9376 (0.6030)	**1.1341** (0.2366)	2.0122 (0.5920)	1.5067 (0.2830)
	0.6196 (0.0800)	0.4675 (0.0464)	0.4282 (0.0895)	0.4314 (0.0304)	**0.4134** (0.0486)	1.1976 (0.1546)	0.5815 (0.1406)
Lorenz	1.8808 (0.2011)	1.8072 (0.2687)	0.7180 (0.1293)	0.7481 (0.1492)	**0.2224** (0.0404)	3.5072 (2.0827)	1.6876 (0.4109)
	0.5896 (0.0616)	0.4835 (0.0384)	0.4022 (0.0390)	0.4346 (0.0352)	**0.1482** (0.0343)	2.4063 (0.3555)	0.6429 (0.1694)
NP1	1.3474 (0.3352)	2.4017 (0.5483)	**0.9853** (1.1366)	1.0103 (6.3796)	2.0003 (0.4270)	3.2094 (1.6154)	1.8089 (0.2567)
	0.8368 (0.1142)	0.6675 (0.1769)	**0.4866** (0.1484)	0.5418 (0.1651)	0.7053 (0.1595)	1.9576 (0.5107)	0.6937 (0.1076)
NP2	3.0113 (0.5588)	2.6129 (0.5935)	2.0684 (0.4329)	2.3414 (0.5980)	**1.1612** (1.3450)	1.9813 (0.6593)	1.6196 (0.1707)
	0.8807 (0.1707)	0.6301 (0.2094)	**0.4657** (0.1260)	0.5082 (0.1415)	0.5177 (0.4335)	1.1624 (0.2153)	0.6412 (0.0867)
NARMA-10	2.6316 (0.2790)	2.5610 (1.4616)	1.8376 (0.4006)	16.2201 (18.1534)	4.2921 (1.4488)	3.2615 (1.9033)	**1.4518** (0.3727)
	0.8207 (0.0908)	0.7084 (0.0895)	**0.4400** (0.0810)	0.6695 (0.1999)	0.8597 (0.3538)	2.2554 (0.2715)	0.5772 (0.1732)

Table D.5: Training (top) and prediction (bottom) CPU time measured in microseconds (ms) for benchmarks. Interquartile ranges are shown in brackets and lowest scores are in **bold**.

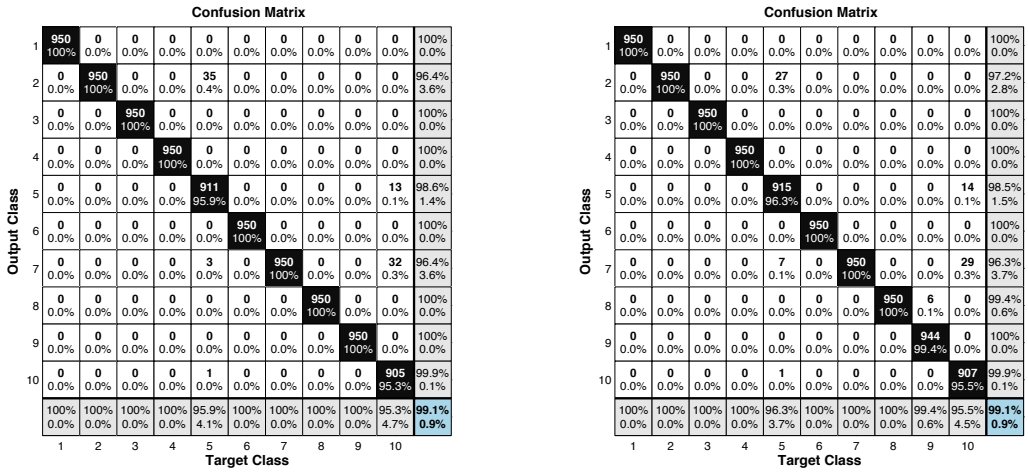
Problem	OESGP	OIESGP	SOGP	$SOGP_{ARD}$	LWPR	KLMS	NORMA
Mackey-Glass	2.6909 (0.4580)	2.5778 (1.9928)	2.3329 (0.4269)	16.7932 (1.6590)	**0.1729** (0.0864)	1.9926 (0.8660)	1.1362 (0.4265)
	0.8090 (0.1064)	0.7530 (0.1724)	0.5602 (0.1581)	0.6881 (0.1291)	**0.1199** (0.0565)	1.2457 (0.1874)	0.4288 (0.1781)
Henon	2.4072 (1.0122)	3.7072 (0.3540)	3.1749 (0.5330)	15.1587 (1.7594)	**0.1751** (0.1638)	2.2800 (0.9700)	1.6036 (0.2719)
	0.6651 (0.1339)	0.5611 (0.0543)	0.6031 (0.1252)	0.6144 (0.1346)	**0.1250** (0.0876)	1.4302 (0.1636)	0.6126 (0.1282)
Laser	1.9730 (1.7989)	2.9922 (0.6310)	1.7103 (0.3626)	6.2890 (0.5663)	**0.1447** (0.0088)	2.4376 (1.0670)	1.5741 (0.3020)
	0.5715 (0.0637)	0.4774 (0.0366)	0.3986 (0.0370)	0.4259 (0.0328)	**0.0976** (0.0133)	1.5163 (0.1965)	0.6057 (0.1416)
Ikeda	3.9925 (0.8327)	2.7859 (0.9596)	2.6993 (0.4703)	7.8095 (1.4388)	2.4411 (2.0879)	2.3571 (1.0353)	**1.5915** (0.3173)
	0.8225 (0.1744)	**0.4695** (0.0558)	0.4777 (0.1418)	0.5062 (0.1814)	1.2229 (0.4635)	1.4672 (0.2044)	0.6127 (0.1524)
Lorenz	2.4580 (0.6490)	3.7796 (0.6282)	2.5075 (0.5365)	2.8084 (6.8110)	**0.1382** (0.0032)	2.4166 (1.0606)	1.4209 (0.4213)
	0.6752 (0.1521)	0.5670 (0.0694)	0.5677 (0.1516)	0.6536 (0.1758)	**0.0930** (0.0022)	1.5172 (0.1958)	0.5654 (0.1859)
NP1	2.0405 (0.4449)	3.7056 (1.5197)	1.9970 (0.3936)	2.1477 (4.1680)	1.5624 (0.5325)	5.0615 (4.3274)	**1.5367** (0.3601)
	0.5534 (0.0445)	0.7417 (0.1152)	**0.3953** (0.0250)	0.4254 (0.0276)	0.4581 (0.1174)	4.5848 (0.3573)	0.6083 (0.1616)
NP2	2.1650 (0.3514)	1.9002 (1.1982)	1.8975 (0.3189)	1.9791 (3.7634)	**1.1332** (1.0986)	2.7909 (1.5284)	1.3944 (0.4697)
	0.5637 (0.0510)	0.4831 (0.0450)	**0.3926** (0.0246)	0.4239 (0.0258)	0.5804 (0.3988)	1.8521 (0.2336)	0.5429 (0.2023)
NARMA-10	2.1708 (0.5031)	2.4389 (0.9105)	2.3027 (0.4999)	30.8567 (2.4362)	**0.1501** (0.0690)	4.5237 (3.7061)	1.1826 (0.4443)
	0.5613 (0.0495)	0.6174 (0.1535)	0.4379 (0.0333)	0.5571 (0.0460)	**0.0971** (0.0516)	4.2440 (0.3474)	0.4527 (0.1895)

Table D.6: Training (top) and prediction (bottom) CPU time measured in microseconds (ms) for the irrelevant dimension prediction task. Interquartile ranges are shown in brackets and lowest scores are in **bold**.

E

SUPPLEMENTARY TACTILE EXPERIMENT RESULTS

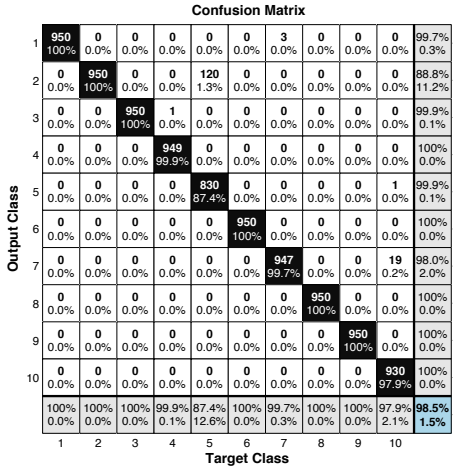
This appendix collates supplementary results for the tactile-based classification experiment performed in Chapter 5. The object labels from one to ten are (1) Empty Plastic Bottle, (2) Half-Full Plastic Bottle, (3) Full Plastic Bottle, (4) Empty Soda Can, (5) Half-Full Soda Can, (6) Teddy-bear, (7) Monkey Soft-toy, (8) Book, (9) Lotion Bottle and (10) None (no object).



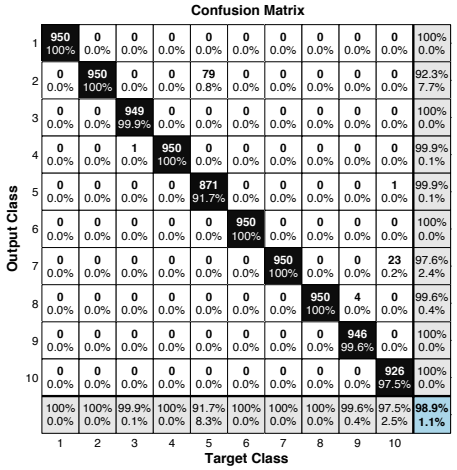
(a) Discriminative (All Features)

(b) Generative (All Features)

Figure E.1: Confusion matrices for the online tactile classification experiment across all samples in Phase 2 using all available features.

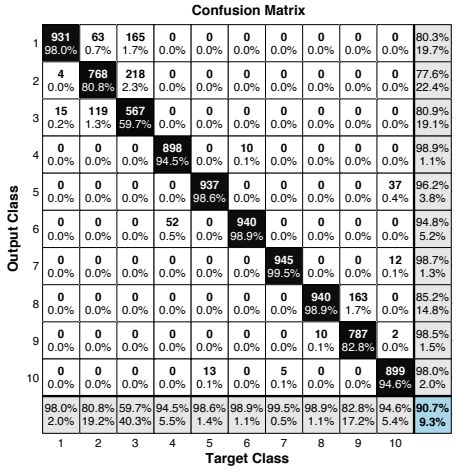


(a) Discriminative (Tactile Features)

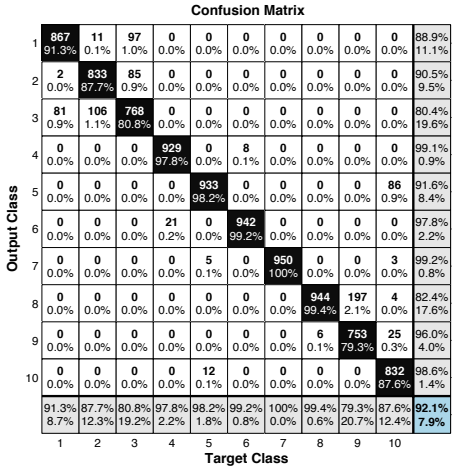


(b) Generative (Tactile Features)

Figure E.2: Confusion matrices for the online tactile classification experiment across all samples in Phase 2 using only tactile features.



(a) Discriminative (Encoder Features)



(b) Generative (Encoder Features)

Figure E.3: Confusion matrices for the online tactile classification experiment across all samples in Phase 2 using only encoder (joint angle) features.

F

PRELIMINARY LAD EXPERIMENT (JOYSTICK CONTROL)

This appendix chapter details our preliminary LAD experiment using joystick control (instead of haptic controllers). Our real-world experiment was set up similar to the simulation experiment; the test environment was a lab office space (Fig. 6.3) and we invited participants to drive the wheelchair along the specified track (from S to F) a total of four times. Before beginning the trials, the participants were allowed to familiarise themselves with the wheelchair by driving around the experimental route until they felt comfortable with its operation. For this experiment, hyperparameter optimisation was disabled and $K_{max} = 1$, i.e., only single experts were used.

During the first trial, full control was conferred, allowing us to obtain the baseline performance of an average user. The control limitation on right turns was applied during the latter three trials; human demonstrator assistance was offered during the third trial (during which the model was simultaneously trained) and LAD model assistance was offered during the final lap[1].

F.1 EMPIRICAL RESULTS: LAP PERFORMANCE

Twelve able-bodied participants (five female) ages 21-38 (mean: 27, sd: 4.91) participated in our experiment. Fig. F.1 shows the lap-times achieved by our participants under the four aforementioned conditions. Under the baseline condition

1 Although the trial ordering may introduce a bias towards lower lap times in the assistance trial, it was only possible to offer the assistance after the model was suitably trained. This was corrected in the experiments presented in Chapter 6 by adding an extra lap and randomising the assistance mode.

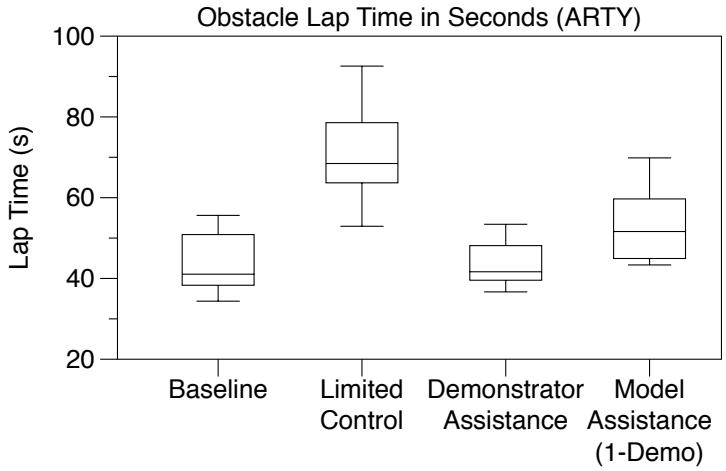
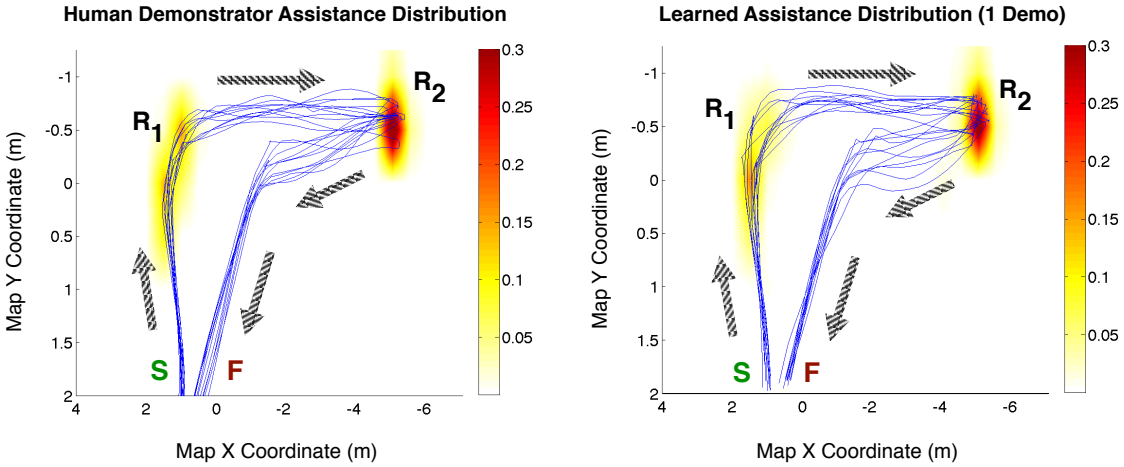


Figure F.1: Lap times in seconds for all twelve participants categorised by driving mode. The learned model improved participant lap performance by 20.6s (speedup of 137%) after only a single demonstration.

(where right turns were normal), participants were able to complete a lap in $\approx$ 48.5s (sd: 20.0s). When right turns were constrained (limited control), lap times increased by 62% to 76.7s (sd: 24.8s). Although the lap times during LAD model assistance was higher than the demonstrator performance (mean: 44.55s, sd: 8s), it is worth noting that the model was only trained using a single demonstration and still improved participants lap times by an average of 20.6s to 56.1s (a speedup of 137% over the limited control setting).

Fig. F.2 shows the paths taken by our subjects and a normalised density plot of when assistance was given. As can be seen, both distributions are similar — we observed the primary difference was that the LAD model would occasionally stop assistance at R_2 earlier than the human demonstrator, resulting in wider deviations. That said, the distribution similarity indicates that assistance was offered by our learned model under the correct situations. In particular, assistance was only offered at R_1 and R_2 and not during the softer right turns in between the waypoints.



(a) Demonstrator (b) LAD Model

Figure F.2: Driven paths and normalised smoothed density plots illustrating *when* assistance was offered by the human demonstrator and LAD model during our preliminary experiment. Both distributions are similar indicating that the LAD model was able to provide appropriate assist at the sharp right turns; in particular, note that assistance was not offered during the return leg from R_2 to the start position despite the moderate right turns.

G

SUPPLEMENTARY LAD EXPERIMENT RESULTS

This appendix contains survey results tables for the Learning Assistance by Demonstration experiment with haptic controllers presented in Chapter 6:

- Table G.1: Survey Responses from 15 participants under the Baseline and Limited control conditions.
- Table G.2: Survey Responses from 15 participants comparing the two laps with human assistance and the LAD model.
- Table G.3: p-values for the aforementioned survey responses.

	Strongly Disagree		Disagree		Neutral		Agree		Strongly Agree	
	B	**L**	**B**	**L**	**B**	**L**	**B**	**L**	**B**	**L**
Q1	0	2	0	3	6	5	5	5	4	0
Q2	6	0	6	3	3	6	0	4	0	2
Q3	0	0	0	6	5	5	8	4	2	0
Q4	3	0	7	3	3	4	1	4	1	4

Table G.1: Survey Responses from 15 participants. Numbers shown are response counts for the (B)aseline and (L)imited control laps.

	Strongly Disagree			Disagree			Neutral			Agree			Strongly Agree		
	H1	**H2**	**M**	**H1**	**H2**	**M**	**H1**	**H2**	**M**	**H1**	**H2**	**M**	**H1**	**H2**	**M**
Q1	0	0	0	1	0	0	2	1	1	10	9	9	2	5	5
Q2	4	7	6	9	4	3	2	3	5	0	1	1	0	0	0
Q3	0	0	0	0	1	0	1	0	3	12	9	8	2	5	4
Q4	2	5	2	8	4	4	3	4	4	1	1	4	1	1	1
Q5	0	0	0	0	0	0	0	2	1	7	6	6	8	7	8
Q6	11	8	8	2	4	3	1	2	2	0	0	1	1	1	1
Q7	0	0	0	0	1	0	0	0	1	4	7	4	11	7	10
Q8	0	0	0	0	0	0	0	2	2	10	9	7	5	4	6
Q9	8	10	7	4	2	5	1	2	1	2	0	1	0	1	1
Q10	5	4	5	6	9	8	4	0	2	0	2	0	0	0	0
Q11	0	0	0	0	1	0	0	0	0	4	5	6	11	9	9

Table G.2: Survey Responses from 15 participants comparing the two laps with human assistance — H1 and H2 for the demonstration and test laps respectively — and the LAD (M)odel. Numbers shown are response counts.

Q1	**H1**	1.000	0.100	0.100	**Q2**	**H1**	1.000	0.121	0.292
	H2	0.197	1.000	1.000		**H2**	0.080	1.000	0.598
	H3	0.197	1.000	1.000		**H3**	0.243	0.605	1.000
Q3	**H1**	1.000	0.100	0.243	**Q4**	**H1**	1.000	0.598	0.035
	H2	0.023	1.000	0.559		**H2**	0.584	1.000	0.114
	H3	0.129	0.584	1.000		**H3**	0.028	0.114	1.000
Q5	**H1**	1.000	0.605	1.000	**Q6**	**H1**	1.000	0.121	0.121
	H2	0.605	1.000	0.605		**H2**	0.080	1.000	1.000
	H3	1.000	0.605	1.000		**H3**	0.080	1.000	1.000
Q7	**H1**	1.000	0.038	0.584	**Q8**	**H1**	1.000	0.559	0.598
	H2	0.020	1.000	0.100		**H2**	0.584	1.000	0.292
	H3	0.559	0.121	1.000		**H3**	0.584	0.243	1.000
Q9	**H1**	1.000	0.273	0.605	**Q10**	**H1**	1.000	0.559	1.000
	H2	0.301	1.000	0.121		**H2**	0.584	1.000	0.584
	H3	0.605	0.100	1.000		**H3**	1.000	0.559	1.000
Q11	**H1**	1.000	0.292	0.292					
	H2	0.243	1.000	1.000					
	H3	0.243	1.000	1.000					

Table G.3: Survey χ^2 p-Values comparing the two laps with human assistance — H1 and H2 for the demonstration and test laps respectively — and the LAD (M)odel.

H

CASE STUDIES WITH THE ARTY SMART WHEELCHAIR

This appendix chapter details case studies involving our prototype intelligent paediatric wheelchair: the Assistive Robot Transport for Youngsters (ARTY), shown in Fig 6.7. ARTY promises to provide an independent lifestyle for disabled children, a training implement for therapists and also an experimental platform for scientists.

Nicholson and Bonsall's 2002 survey of 193 wheelchair services [Nicholson and Bonsall, 2002] showed that 51% of the respondents did not supply wheelchairs to children under 5 years. The top two reasons cited were safety of the child (36%) and safety of others (34%). Although safety is clearly an important factor, for these children to lose independent mobility is a crucial set-back at a critical age. Mobility loss spawns a vicious cycle: the lack of mobility inhibits cognitive, emotional and social development, which in turn further limits personal independence [Butler, 1991; Demiris, 2009b; Nisbet et al., 1996]. Ultimately, this results in a severe long-term deterioration in a child's quality of life.

In our research, we aim to break this cycle by providing a key-enabling technology: a *safe*, intelligent paediatric wheelchair. In contrast to traditional assumptions, powered mobility can be made safe, for the child and others. Risks can be mitigated through the use of robotic technology and shared control systems [Carlson and Demiris, 2012, 2010b,a, 2008; Simpson, 2005]. Safe powered mobility can improve social, emotional and intellectual behaviour [Butler et al., 1983] and has the potential to drastically change lives.

We describe a case-study involving able-bodied children comparing two different shared control mechanisms. We also present one case study involving a

young 5-year boy with special needs, who was considered by healthcare professionals not yet appropriate for a *regular* powered wheelchair.

In the following section, we provide a review of powered mobility for children and research on smart children's wheelchairs. In Section 6.4 of Chapter 6, we presented technical details on ARTY. Here, we supplement this discussion with details on our hybridised shared-control algorithm, given in Section H.2. This is followed by Sections H.3 and H.4 which details our experimental findings with child participants. Finally, Section H.5 presents conclusions and discusses our planned future work.

H.1 BACKGROUND

In this section, we review the current state of powered mobility for children and recent research on smart paediatric wheelchairs.

H.1.1 *Powered Wheelchairs for Young Children*

In his 1987 paper, Hays [Hays, 1987] identified four categories of children who could benefit from powered mobility: those who will never walk, those who cannot efficiently move in a walker or manual wheelchair, those who lose their mobility due to traumatic injury or neuromuscular disorder and those who require temporary assistance (such as after surgery). In the UK alone, there are more than 50,000 disabled children who fall under this category and require mobility assistance [Cox, 2003]. Despite the provision difficulties stated in the introduction, powered mobility advocates consider mobility "an essential component of a child's early intervention program" [Jones et al., 2003; Tefft et al., 1999].

A survey of 1200 members of the National Association of Paediatric Occupational Therapists (NAPOT) [Orpwood et al., 2005] summarised the key features that an ideal young children's wheelchair should possess. For example, the wheelchair should have a "fun" appearance (such that it appears to be a play-

thing) and the ability to be a training aid in early stages. It should be adaptable so that it could be used by children with different disabilities, i.e, accommodate different seating systems and a wide variety of control options (e.g., joystick, sip and puff). A subsequent survey carried out by the charity Whizz-Kidz concluded that no commercial powered wheelchair met these specifications and significant improvements were necessary before commercial systems meet the needs of many disabled children and their caregivers.

H.1.2 *Smart Paediatric Wheelchairs*

Research on smart wheelchairs has grown over the past two decades, fuelled in part by the recognition that standard powered wheelchairs are lacking in many respects. A recent analysis by Simpson [Simpson et al., 2008] showed that 61-91% of all wheelchair users would benefit from a smart wheelchair (approximately 1.4 to 2.1 million people). Early work in (adult) smart wheelchairs [Yanco, 1998b] include MIT's Wheelesley [Yanco, 1998a], the TAO wheelchairs [Gomi and Griffith, 1998], Tin Man II [Miller and Slack, 1995] and NavChair [Levine et al., 1999], with more recent prototypes developed by research teams around the world [Simpson et al., 2004, 2002], including the Personal Robotics Lab at Imperial College [Carlson and Demiris, 2008, 2010b]. For a more complete review of adult smart wheelchairs, we refer readers to comprehensive surveys in [Carlson, 2010; Simpson, 2005].

Of interest to us are smart *paediatric* wheelchairs, a subtype of intelligent wheelchairs that have garnered far less attention. Early seminal work in the area was performed by the Communication Aids for Language and Learning (CALL) Centre at the University of Edinburgh [Nisbet et al., 1996]. They reported that the introduction of twelve smart wheelchairs to three special schools encouraged motivation and developmental improvements in ten disabled participants [CALL Center, 1994]. In fact, most of the children experienced significant improvements in mobility and psychosocial traits. There currently exist two commercial systems based on the CALL Center Smart Wheelchair: the "Smart

Wheelchair" and the "Smart Box" (both distributed by Smile Rehab Ltd.) [Simpson, 2005]. However, the Smart Wheelchair is expensive (USD 14,000) and is only capable of rudimentary abilities (bump and stop/backup/turn, line following). The cheaper Smart Box (USD 5,000) was designed to be fitted onto a standard wheelchair to provide similar capabilities to the Smart Wheelchair.

The falling cost of sensors and computational power has resulted in the incorporation of more sophisticated sensing devices and algorithms than the CALL Centre wheelchair (which was equipped with bump and sonar sensors). Recent smart children's wheelchairs use infra-red (IR) rangers, vision-based methods, shared-control algorithms and obstacle-avoidance algorithms for semi or fully-autonomous navigation [Marchal-Crespo et al., 2010; Ceres et al., 2005; Galloway et al., 2008; Lynch, 2009]. Moreover, researchers have begun incorporating the use of haptic devices to train children, with the goal of supplementing or possibly replacing the hand-over-hand method currently employed in rehabilitation centers. One noteworthy paper by Marchal-Crespo, Furumasu and Reinkensmeyer [Marchal-Crespo et al., 2010] discussed the use of fading haptic guidance. Their study concluded that their system, RO-bot-assisted Learning for Young drivers (ROLY), improved the steering ability of twenty-two able-bodied children and one disabled child with cerebral palsy.

Despite remarkable technological progress, we found a lack of structured clinical trials with disabled users. A notable exception is the PALMA project where Ceres *et al.* [Ceres et al., 2005] performed a small-scale clinical study of their robotic wheelchair involving five children with severe mobility impairments and reduced motor control. Experiments with end-users are challenging for many reasons. For example, the CALL Center wheelchair was not a single entity but multiple variants had to be designed (to accommodate different users) [CALL Center, 1994]. That said, research needs to move beyond laboratory settings with able-bodied children to real-world locations (such as rehabilitation centers) with end-users to be relevant; this is one challenge that requires significant effort, without which, limits smart wheelchairs from gaining widespread acceptance [Carlson and Demiris, 2010a].

H.2 SHARED-CONTROL ON THE ARTY PLATFORM

For this work, we used a hybrid shared-control (HSC) method that combines the merits of the Combined Vector Field (CVF) [Borenstein and Raschke, 1992] (a variant of the Vector Field Histogram (VFH) [Borenstein et al., 1991] for non-point robots) and the Dynamic Window Approach (DWA) [Fox et al., 1997]. In initial versions of ARTY, we have used either VFH *or* DWA. Both presented problems. VFH is computationally low-cost but unfortunately, difficult to tune (particularly since our wheelchair is rectangular and non-holonomic). Furthermore, the relationship between VFH parameters and behaviour was not always obvious, making it difficult to assure ourselves that the method would work in non-tested situations.

On the other hand, DWA, which incorporates the wheelchair's shape and dynamics, was easier to tune and provided greater assurance, but required far greater computation time; the projections and obstacle collision prevention for non-circular robots requires many point-in-polygon checks.

To overcome these limitations, our hybrid approach (HSC) uses CVF to provide an approximate solution, which is refined using (limited) DWA. The idea is straight-forward: we first project forwards in time using the robot's dynamics (using motion equations) and check for future collisions. If there are none, the user's command velocity is left unchanged. However, if a future collision is predicted, we use CVF to generate the principal steering direction, which is then converted into a control velocity. Instead of sending this directly to the motors, we further refine this control by considering *scaled* command velocities. In this work, we generated 11 commands where the scale α was varied from 0 to 1 (inclusive) with 0.1 step size. DWA was then used to search through this space for the optimal solution (and provide assurance that command velocity did not result in a collision). Note that if all the command velocities result in a collision, the default command is zero.

There are three shared-control modes: basic, safeguarding and assisted. For the assisted control mode, HSC is applied. For the safeguarding mode, CVF is

not used; instead, the scaled command velocities are generated directly from the user's control velocities. Early tests showed us that HSC achieves a fast response time (15Hz-20Hz) on our hardware while still providing a smooth driving experience for the user.

H.2.1 *Obstacle Map*

One significant difference in our implementation is in the obstacle map (OM); a representation of potential obstacles in the environment [Thrun et al., 2005]. Building an OM is equivalent to estimating the posterior probability over possible maps m given the observations z and robot poses x thus far, i.e., $p(m|z_{1:t}, x_{1:t})$.

However, this is too computationally expensive to execute in real-time and as such, the problem is usually simplified in three ways. First, the problem is reduced from three-dimensions to two (so, we have a slice instead of a volume). Second, each cell of the map, m_i, is considered independent and as such, $p(m|z_{1:t}, x_{1:t}) = \prod p(m_i|z_{1:t}, x_{1:t})$. Finally, instead of using a true probabilistic model of the sensor data, we resort to a simple binary model; $p(m_i|z_{1:t}, x_{1:t}) = 1$ if the sensor reports a "hit" at m_i and clear, $p(m_i|z_{1:t}, x_{1:t}) = 0$, if no hit is reported or the cell is in the sensor's line of sight between the robot and a detected obstacle.

In our work, instead of the standard cell grid representation, we used a KD-tree which conferred quick rectangular range searches (on the order of $O(\sqrt{n} + m)$ where n is the number of obstacles in the tree and m is the number of reported points). This also permitted us to store obstacle locations with greater precision compared to the cell-grid (which requires a pre-set resolution).

H.3 CASE STUDY WITH ABLE-BODIED PARTICIPANTS

In this case-study, we sought to compare the safeguarding and assisted control modes described in the previous section. Recall that the assisted mode is more intrusive, modulating the user's control to avoid obstacles, and not merely pre-

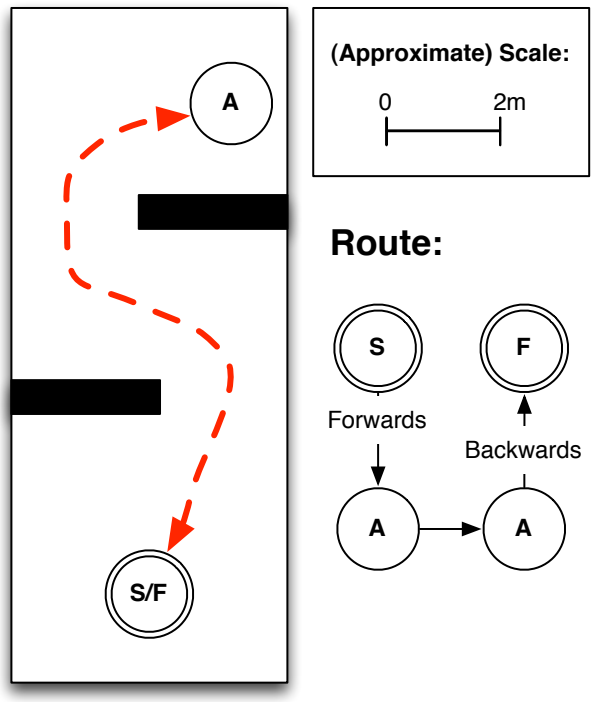
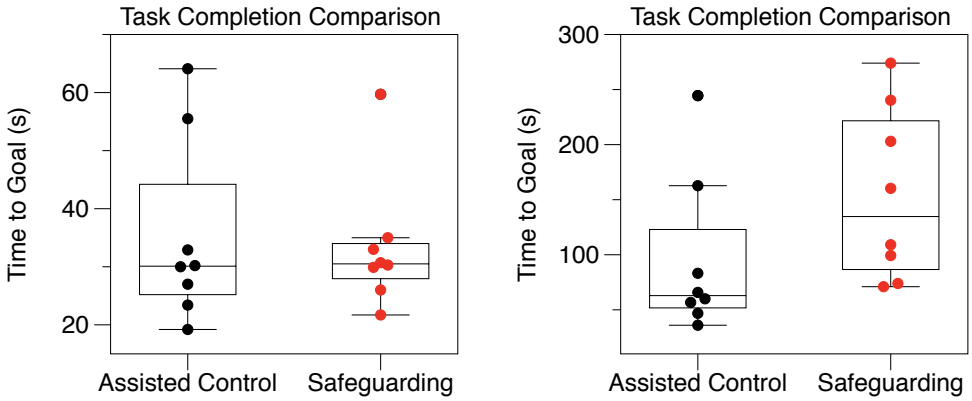


Figure H.1: Obstacle Course for Forwards and Backwards Driving Task (with able-bodied children).



(a) Forwards Driving Task

(b) Backwards Driving Task

Figure H.2: Task-completion times for the forwards and backwards driving portion of the task.

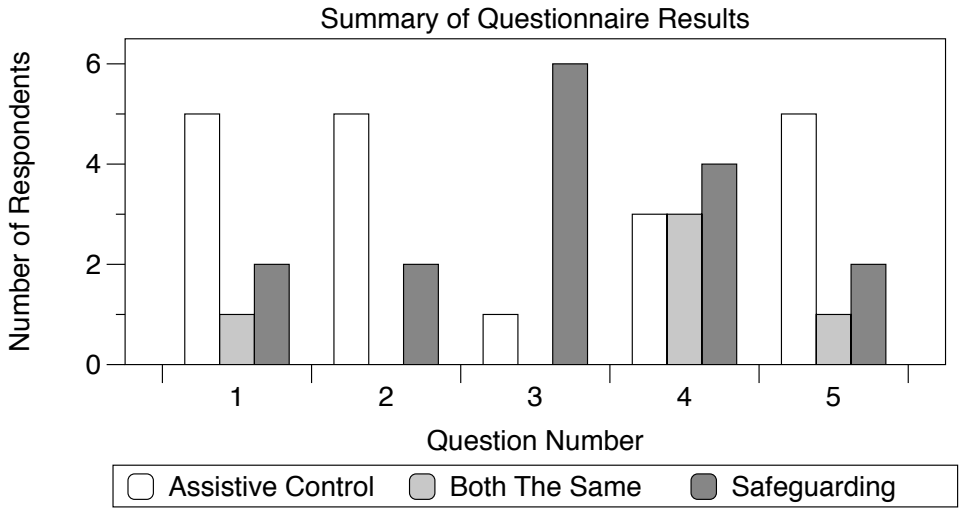


Figure H.3: Summary of Post-trial Questionnaire Results. The bars indicate how many of the respondents indicated higher agreement for each of the modes. Overall, responses for the assisted control was more positive compared to safeguarding.

vent collisions (which safeguarding accomplishes by reducing the scale of the user's command). Previous studies have performed similar comparisons with adults but our experiment was conducted with child participants.

H.3.1 *Experimental Setup*

We set up the obstacle course shown in Fig. H.1 where the task was to drive through the course as quickly as possible — straightforward but with one small complication. Because early trials indicated that forwards driving is relatively easy for able-bodied children, we instructed the participants to drive normally from the start position (S) to the half-way point (A) but *backwards* from A to the finish point (F).

Each child drove through the obstacle course twice, i.e., two *runs* (one for each mode). The mode order was randomised and the participants were not told which mode came first. After each run, they were given a questionnaire (under supervision) asking them how much they agreed with each of the following statements on a five point Likert scale (1 = strongly disagree, 5 = strongly agree):

1. The wheelchair was easy to manoeuvre.

2. The wheelchair behaved as I expected.
3. I had to concentrate hard to drive the wheelchair.
4. It felt natural driving the wheelchair.
5. The obstacle course was easy.

Additionally, after completing both runs, they were asked which run they preferred overall.

H.3.2 *Results and Discussion*

Eight children (aged 11 years) participated in our experiment. As Fig. H.2 shows, the children completed the first portion of the task (forwards from S to A) significantly faster than the second portion (backwards from A to F). When using safeguarding, they took an average of 4.3 times longer to complete the backwards segment compared to 2.1 times when using assisted control.

Assisted control did not have a measurable positive effect over safeguarding on the simpler forwards driving portion, but it reduced the time needed to complete the backwards driving segment by an average of 62.9 seconds. This difference is statistically significant ($p \approx 0.02$). When asked, seven of the eight children preferred assisted control. This choice is supported by the individual questionnaire results (Fig. H.3); the five of the eight participants found ARTY under assisted control to be easier to manoeuvre and behaved more as they expected.

However, it should be noted that not all the participants appreciated the extra assistance. Turning our attention to the one child who preferred safeguarding over assisted control, we postulate that the assisted control caused him confusion when the algorithm changed his control "too much". When the wheelchair swerved to avoid an obstacle, he stopped completely or issued fast "corrective" movements. One research topic of interest for us is detecting and providing the *appropriate* level of assisted control to accommodate such users [Demiris, 2009b].

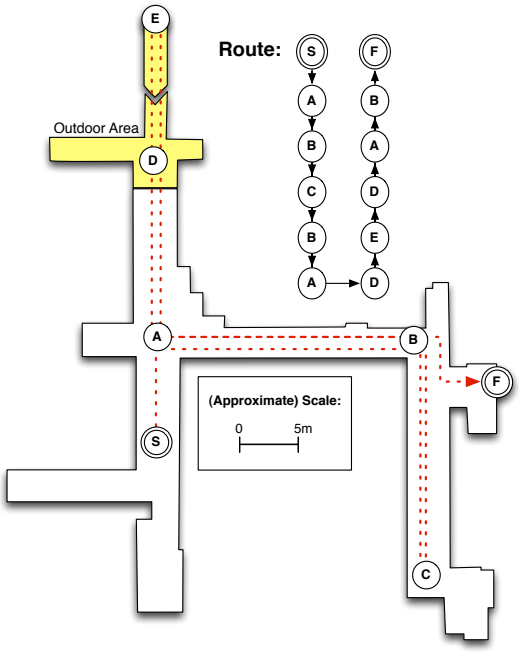


Figure H.4: Experimental Area and C's Driving Route.

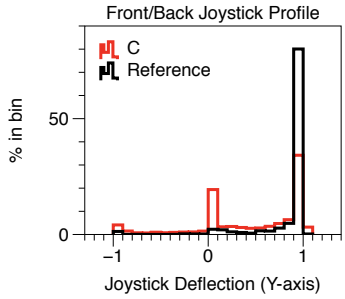
Notwithstanding the fact that more confirmatory studies are needed, these results suggest that assisted control is preferable to safeguarding for most children. Based on this conclusion, we used assisted control in the following case study involving a young child with special needs.

H.4 CASE STUDY: C, A 5-YEAR OLD BOY WITH SPECIAL NEEDS

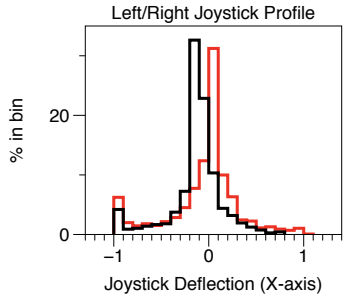
In this section, we report on a trial-run with C, a five-year old boy with both physical and cognitive disabilities [1]. Because of his age and condition (reduced inhibition and increased impulsiveness), C was considered by his occupational therapist (OT) to be *not yet ready* for a *regular* powered wheelchair. It was his OT's hope that ARTY would allow C to increase the amount of sensory feedback he would gain with movement while remaining safe and calm.

The experiment took place at the rehabilitation center where C is currently a patient (map shown in Fig. H.4). Before the start of the trial, C's OT gave him a short introduction to ARTY and told about how it worked.

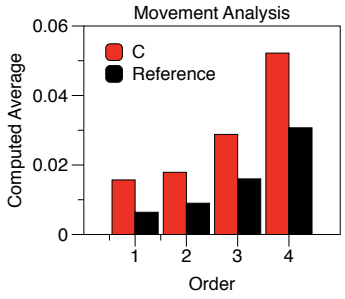
1 Informed consent was obtained from C's mother before the session.



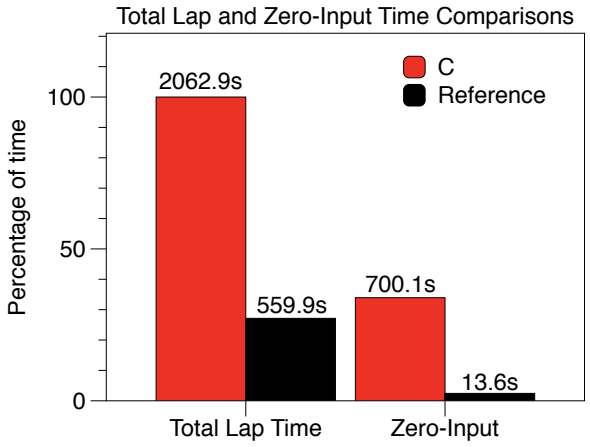
(a) Forwards/Backwards Profile



(b) Left/Right Profile



(c) Movement Analysis



(d) Task Time Analysis

Figure H.5: Analysis of the data collected during C's session.

Since this was his first try, the wheelchair was set to a relatively low maximum velocity (0.4m/s translational and 0.4rads/s rotational). Assisted control was used throughout (except in special cases detailed below). Throughout the session, C was allowed the opportunity to freely explore while his OTs provided directional cues and supervision. Data (including sensor readings, joystick movements and assisted controls) was logged at 20Hz. After C's session, an expert driver drove the wheelchair along a similar route to provide a *reference dataset* for comparison.

H.4.1 *Results and Discussion*

In general, we observed that C remained calm and interested while driving ARTY. The driving portion of the session lasted 33.4 minutes and the route taken consisted of both indoor and outdoor areas, as shown in Fig. H.4.

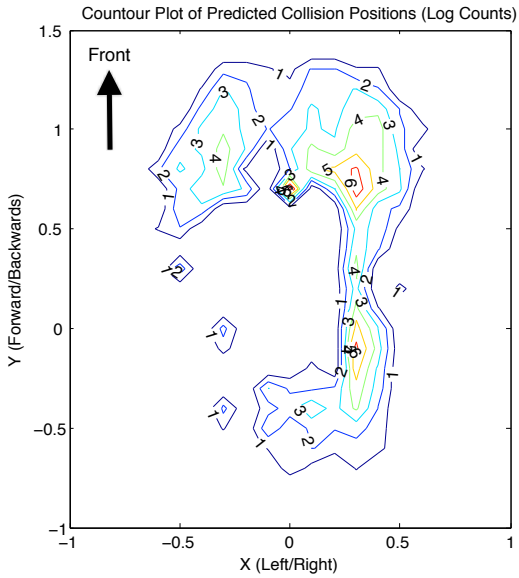
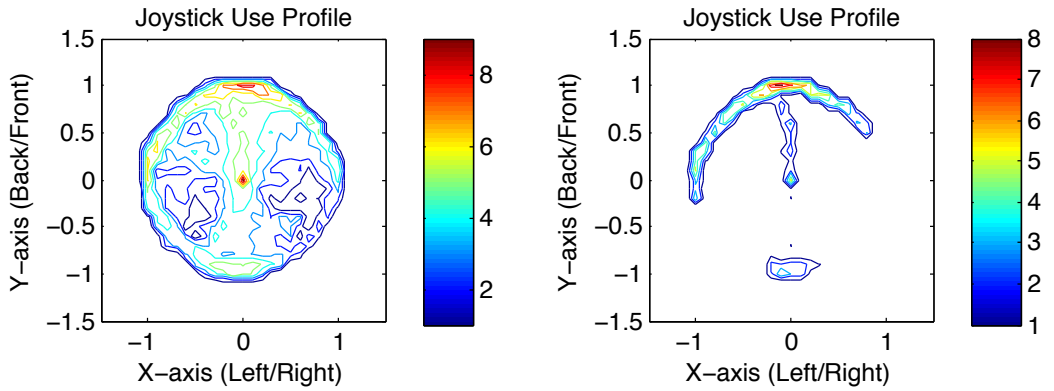


Figure H.6: Predicted Crash Positions. Note that the wheelchair is centred at (0,0) and most of the predicted crash positions occurred in the front area of the wheelchair and on the right-hand side. This was consistent with our observations during the session.



(a) C (b) Reference

Figure H.7: Joystick Profiles for C and the expert driver. The contours indicate log frequencies of the joystick positions during the driving sessions. Best viewed in colour.

To travel the same approximate route took the expert driver 7 minutes. This difference is large but not surprising; the goal of this session was to explore rather to race. Along the route, C interacted with objects and engaged with his friends. Comparing the actual distance travelled (reported by ARTY's odometry), C drove 213m; 44m (20%) more than the reference. In addition, his *non-active* driving time (defined as any contiguous time segment with no joystick input exceeding 5 seconds) was 20% of his total time (See Fig. H.5d), larger than the expert's 6% (25s)[2]. For the following analyses, the non-active segments were removed to avoid non-informative zero-centred peaks in the obtained distributions.

During the session, we observed C had a right-side bias when operating the joystick; this is clear when comparing C's x-axis joystick distribution against the reference (Fig. H.5b). Furthermore, a majority of the predicted (and avoided) collisions were on the right-side of the wheelchair (Fig. H.6).

This right-side bias would also often lead C very close to walls and into situations where he would get stuck in a corner. In these cases, after giving C an opportunity to manoeuvre his way out, his OT provided assistance. However, because of the position C would get stuck in, we noted that it was difficult for the OT to control the wheelchair (since she had to stand on the left side of the wheelchair while the wall and joystick were on the right) and we had to disable shared-control briefly using the tablet PC. To avoid similar difficulties in future sessions, we plan to provide the OT with a simple hand-held remote control to toggle shared-control.

Taking a closer look at C's joystick use, we observed his joystick position distribution (Fig. H.7), to be broader compared to the expert. A possible reason was that C, being unfamiliar with driving wheelchairs, more actively explored the joystick space. Another contributing factor was that whenever C got stuck, he would engage in random joystick motions in an attempt to "get free". We also noted that during 180 degree turns, he would always turn counter-clockwise, leading to the higher frequency on the far-left side of the distribution (Fig. H.5b).

2 The expert driver had to stop and wait at times for passer-bys to cross.

To better understand C's control capabilities, we computed the average first through fourth difference orders on the joystick movement data. This is akin to the velocity, acceleration, jerk and snap of the motion (but without the division by time). As Fig. H.5c shows, movements were more rapid and jerky compared to the expert; for adults, quick motions are an indication of inexperience with the joystick since rapid movements typically indicate quick corrective movements [Carlson and Demiris, 2008].

Overall, his OTs considered C's session with ARTY to be a success; C drove safely around his environment (both indoors and outdoors). Moreover, C's experience with ARTY was his *first* experience driving a powered wheelchair. Under normal circumstances, he would not have been allowed to drive a wheelchair until he was older. ARTY provided him with early access to the path towards mobility independence and it is expected that C will participate in future sessions.

H.5 CONCLUSIONS AND FUTURE WORK

In this chapter, we reviewed the current state of research on smart paediatric wheelchairs and provided two case studies using ARTY; the first involving eight able-bodied children who demonstrated the benefits of our shared-control method. The second involved C, a five-year old boy with both physical and cognitive disabilities. Although it was C's first time in a powered wheelchair, he drove ARTY for more than 30 minutes and engaged with objects and people in his environment. As we begin to conduct more studies with children with disabilities, we believe it is necessary to further develop smart wheelchair interaction/experimental methodologies as well as analytical tools. It is important to develop metrics to provide occupational therapists with quantitative measures of wheelchair driving performance. Finally, we consider our case-study with C to be an important milestone: a proof-of-concept demonstrating the feasibility of using smart wheelchairs in rehabilitation centres to provide young children with early access to independent mobility.

BIBLIOGRAPHY

M. A. Aizerman, E. A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control*, number 25 in Automation and Remote Control, pages 821–837, 1964.

S. I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

S. I. Amari, H. Park, and K. Fukumizu. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12(6):1399–1409, 2000.

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

A. F. Atiya and A. G. Parlos. New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, 11(3):697–709, 2000.

Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

J. Borenstein and U. Raschke. Real-time obstacle avoidance for non-point mobile robots. *SME Transactions on Robotics Research*, 2:2.1–2.10, 1992.

J. Borenstein, Y. Koren, S. Member, and A. Arbor. The Vector Field Histogram. *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

C. Butler. Augmentative mobility: why do it? *Physical Medicine Rehabilitation Clinics of North America*, 2(4):801–15, 1991.

C. Butler, G. A. Okamoto, and T. M. McKay. Powered mobility for very young disabled children. *Developmental Medicine and Child Neurology*, 25(4):472–474, 1983.

CALL Center. Learning through Smart Wheelchairs Final Report, 1994.

T. Carlson and Y. Demiris. Human-wheelchair collaboration through prediction of intention and adaptive assistance. In *IEEE International Conference on Robotics and Automation*, 2008 pages 3926 –3931.

T. Carlson and Y. Demiris. 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, 2012.

T. Carlson. *Collaborative Control Mechanisms for an Intelligent Robotic Wheelchair*. PhD thesis, Imperial College London, 2010.

T. Carlson and Y. Demiris. Robotic Wheelchairs: Scientific Experimentation or Social Intervention? *Euron GEM Sig Workshop on The Role of Experiments in Robotics Research at ICRA 2010*, 2010a.

T. Carlson and Y. Demiris. Increasing Robotic Wheelchair Safety With Collaborative Control : Evidence from Secondary Task Experiments. *IEEE International Conference on Robotics and Automation*, pages 5582–5587, 2010b.

M. Casdagli, S. Eubank, J. D. Farmer, and J. Gibson. State space reconstruction in the presence of noise. *Physica D: Nonlinear Phenomena*, 51(1):52–98, 1991.

R. Ceres, J. Pons, L. Calderon, A. Jimenez, and L. Azevedo. A Robotic Vehicle for Disabled Children. *IEEE Engineering in Medicine and Biology Magazine*, 24 (6):55–63, 2005.

K. M. A. Chai, C. K. I. Williams, S. Klanke, and S. Vijayakumar. Multi-task gaussian process learning of robot inverse dynamics. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 265–272, 2008.

S. Chatzis and Y. Demiris. Echo State Gaussian Process. *IEEE Transactions on Neural Networks*, 2011 22(9):1435 –1445.

S. Chitta, M. Piccoli, and J. Sturm. Tactile object class and internal state recognition for mobile manipulation. In *IEEE International Conference on Robotics and Automation*, 2010 pages 2342–2348.

S. Chitta, J. Sturm, M. Piccoli, and W. Burgard. Tactile Sensing for Mobile Manipulation. *IEEE Transactions on Robotics*, 2011 27(3):558 –568.

H. Chow and Y. Xu. Learning human navigational skill for smart wheelchair in a static cluttered route. *IEEE Transactions on Industrial Electronics*, 53(4): 1350–1361, June.

C. Cortes and V. Vapnik. Support-vector networks. 20(3):273–297, 1995.

D. L. Cox. Wheelchair Needs for Children and Young People : a Review. *British Journal of Occupational Therapy*, 66(May):219–223, 2003.

L. Csató. *Gaussian processes: iterative sparse approximations*. PhD thesis, Aston University, 2002.

L. Csató and M. Opper. Sparse On-Line Gaussian Processes. *Neural Computation*, 2002 14(3):641–668.

T. A. D. Schiebener, J. Morimoto and A. Ude. Integrating visual perception and manipulation for autonomous learning of object representations. *Adaptive Behavior*, pages 0–0, 2013.

E. Demeester, A. Hüntemann, D. Vanhooydonck, G. Vanacker, H. Van Brussel, and M. Nuttin. User-adapted plan recognition and user-adapted shared control: A Bayesian approach to semi-autonomous wheelchair driving. *Autonomous Robots*, 2007 24(2):193–211.

Y. Demiris. Knowing when to assist: developmental issues in lifelong assistive robotics. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3357–3360, 2009a.

Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition (hammer). *Robotics and Autonomous Systems*, 54(361-369), 2006.

Y. Demiris. Prediction of intent in robotics and multi-agent systems. *Cognitive Processing*, 8(3):151–158, 2007.

Y. Demiris. Knowing when to assist: developmental issues in lifelong assistive robotics. *Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, pages 3357–3360, 2009b.

Z. Deng and Y. Zhang. Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on Neural Networks*, 2007 18(5):1364 –1375.

R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47:109 – 116, 2004.

A. D. Dragan and S. S. Srinivasa. Assistive teleoperation for manipulation tasks. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 123–124, 2012.

A. Drimus, G. Kootstra, A. Bilberg, and D. Kragic. Classification of rigid and deformable objects using a novel tactile sensor. In *2011 15th International Conference on Advanced Robotics (ICAR)*, 2011 pages 427 –434.

J. Earwaker. *The tutorial relationship*, chapter 6. Taking Issue: Debates in Guidance and Conselling in Learning. Routledge, 1998.

Y. Engel, S. Mannor, and R. Meir. The Kernel Recursive Least-Squares Algorithm. *IEEE Transactions on Signal Processing*, 2004 52(8):2275–2285.

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997.

K.-i. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

J. C. C. Galloway, J.-C. Ryu, and S. K. Agrawal. Babies driving robots: self-generated mobility in very young infants. *Intelligent Service Robotics*, 1(2):123–134, 2008.

M. G. Genton. Classes of kernels for machine learning: a statistics perspective. *The Journal of Machine Learning Research*, 2:299–312, 2002.

B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

A. Girard. *Approximate Methods for Propagation of Uncertainty with Gaussian Process Model*. PhD thesis, University of Glasgow, Glasgow, UK, 2004.

A. Girard, C. E. Rasmussen, J. Quinonero-Candela, R. Murray-Smith, J. Quiñonero-Candela, O. Winther, J. Quiñonero-Candela, A. Girard, J. Larsen, C. Rasmussen, et al. Multiple-step ahead prediction for non linear dynamic systems–a gaussian process treatment with propagation of the uncertainty. *Advances in neural information processing systems*, 15:529–536, 2002.

P. W. Goldberg, C. K. Williams, and C. M. Bishop. Regression with input-dependent noise: A gaussian process treatment. *Advances in neural information processing systems*, 10:493–499, 1997.

T. Gomi and A. Griffith. Developing Intelligent Wheelchairs for the Handicapped. In *Assistive Technology and Artificial Intelligence*, volume 1458 of *Applications in Robotics, User Interfaces and Natural Language Processing*, pages 150 – 178, 1998.

N. Gorges, S. Navarro, D. Göger, and H. Worn. Haptic object recognition using passive joints and haptic key features. In *IEEE International Conference on Robotics and Automation*, pages 2349–2355, 2010.

D. Grollman and O. Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *IEEE International Conference on Robotics and Automation*, 2008 pages 3315 –3320.

S. Grossberg. Recurrent neural networks. 8(2):1888, 2013.

G. Guennebaud, B. Jacob, et al. Eigen v3, 2010. URL `http://eigen.tuxfamily.org`.

H. Hagras, V. Callaghan, and M. Collry. Outdoor mobile robot learning and adaptation. *Robotics & Automation Magazine, IEEE*, 8(3):53–69, 2001.

R. M. Hays. Childhood motor impairments: Clinical overview and scope of the problem. In K. M. Jaffe, editor, *Proceedings of the First Northwest Regional Conference on Childhood powered mobility: Developmental, technical and clinical perspectives*, pages 1–10, Washington, DC, 1987. RESNA.

J. Hensman and N. D. Lawrence. Gaussian Processes for Big Data through Stochastic Variational Inference. In *NIPS 2012 Workshop on Parallel and Large-scale Machine Learning*, 2012.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian Processes for Big Data. In *Uncertainty in Artificial Intelligence (UAI-13)*, 2013.

M. Hermans and B. Schrauwen. Recurrent Kernel Machines: Computing with Infinite Echo State Networks. *Neural Computation*, 24(1):104–133, 2011.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

P. J. and Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356, 1988.

H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report 148, German National Research Center for Information Technology, Bremen, 2001.

H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *Advances in Neural Information Processing Systems*, pages 593–600, 2003.

H. Jaeger. A tutorial on training recurrent neural networks, covering bppt , rtrl , ekf and the "echo state network" approach. Technical report, German National Research Center for Information Technology, 2005.

M. A. Jones, I. R. McEwen, and L. Hansen. Use of power mobility for a young child with spinal muscular atrophy. *Physical Therapy*, 83(3):253–262, 2003.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 1993 5(6):954–975.

H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*, volume 7. Cambridge University Press, 2003.

A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *IEEE 11th International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

F. Kianifard and W. H. Swallow. A review of the development and application of recursive residuals in linear models. *Journal of the American Statistical Association*, 91(433):pp. 391–400, 1996.

J. Kivinen, A. Smola, and R. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 2004 52(8):2165 – 2176.

S. Klanke and S. Vijayakumar. LWPR supplementary documentation. Technical report, University of Edinburgh, 2008.

P. Kountouriotis, D. Obradovic, S. Goh, and D. Mandic. Multi-step forecasting using echo state networks. In *The International Conference on Computer as a Tool (EUROCON 2005)*, volume 2, pages 1574–1577. IEEE, 2005.

S. P. Levine, D. A. Bell, L. A. Jaros, R. C. Simpson, Y. Koren, and J. Borenstein. The NavChair Assistive Wheelchair Navigation System. *IEEE Transactions on Rehabilitation Engineering*, 7(4):443–451, 1999.

W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 9–14, 2010.

W. Liu, P. Pokharel, and J. Principe. The Kernel Least-Mean-Square Algorithm. *IEEE Transactions on Signal Processing*, 56(2):543–554, 2008.

R. Lomasky, C. E. Brodley, M. Aernecke, D. Walt, and M. Friedl. Active class selection. In *Machine learning: ECML 2007*, pages 640–647. Springer, 2007.

M. Lukosevicius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

M. Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.

M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 2009 3(3):127–149.

A. K. Lynch. *Robot assisted mobility for very young infants.* PhD thesis, University of Delaware, 2009.

W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

L. Marchal-Crespo, J. Furumasu, and D. J. Reinkensmeyer. A robotic wheelchair trainer: design overview and a feasibility study. *Journal of NeuroEngineering and Rehabilitation*, 7:40, 2010.

A. McHutchon and C. E. Rasmussen. Gaussian process training with input noise. In *Advances in Neural Information Processing Systems*, pages 1341–1349, 2011.

M. Meier, M. Schopfer, R. Haschke, and H. Ritter. A probabilistic approach to tactile shape reconstruction. *IEEE Transactions on Robotics*, 27(3):630–635, 2011.

G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The iCub humanoid robot: an open platform for research in embodied cognition. In *8th Workshop on Performance Metrics for Intelligent Systems*, pages 50–56, New York, NY, USA, 2008.

D. P. Miller and M. G. Slack. Design and testing of a low-cost robotic wheelchair prototype. *Autonomous Robots*, 2(1):77–88, 1995.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA, USA, 2012.

R. Neal. *Bayesian learning for neural networks*. Number 118 in Lecture Notes in Statistics. Springer Verlag, 1996.

R. Neal. *Bayesian Statistics*, volume 6, chapter Regression and classification using Gaussian process priors, pages 475—501. Oxford University Press, 1997.

C. Nezhat, O. Lavie, M. Lemyre, E. Unal, C. H. Nezhat, and F. Nezhat. Robot-assisted laparoscopic surgery in gynecology: scientific dream or reality? *Fertility and Sterility*, 91(6):2620 – 2622, 2009.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, pages 841–848, 2001.

D. Nguyen-Tuong, M. Seeger, and J. Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

J. Nicholson and M. Bonsall. Powered Mobility for Children under Five Years of Age in England. *British Journal of Occupational Therapy*, 65(6):291–293, 2002.

S. Nikolaidis and J. Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*, pages 33–40, 2013.

P. Nisbet, J. Craig, P. Odor, and S. Aitken. "Smart" wheelchairs for mobility training. *Technology Disability*, 5(1):49–62, 1996.

Novint Technologies. Falcon Technical Specifications, 2013. URL http://www.novint.com/index.php/novintxio/41.

M. Opper. A Bayesian approach to on-line learning. *Online Learning in Neural Networks*, pages 363–378, 1998.

R. Orpwood, S. Halsey, N. Evans, and A. Harris. *Development of a Powered Mobility Vehicle for Nursery-aged Children*, pages 540–545. IOS Press, 2005.

Z. Pezzementi, E. Plaku, C. Reyda, and G. Hager. Tactile-object recognition from appearance information. *IEEE Transactions on Robotics*, 2011 27(3):473 –487.

J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 2005 6: 1939–1959.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 3, 2009.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems*, 2:881–888, 2002.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

C. Richard, J. Bermudez, and P. Honeine. Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing*, 2009 57(3):1058 –1067.

A. Rodan and P. Tino. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 2011 22(1):131 –144.

T. Rofer. Architecture and applications of the Bremen Autonomous Wheelchair. *Information Sciences*, 126(1-4):365–368, 1998.

D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

A. M. Schäfer and H. G. Zimmermann. Recurrent neural networks are universal approximators. In *Artificial Neural Networks–ICANN 2006*, pages 632–640. Springer, 2006.

A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard. Object identification with tactile sensors using bag-of-features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 243 –248, 2009.

M. Schneider and W. Ertel. Robot learning by demonstration with local gaussian process regression. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 255–260, 2010.

M. Schöpfer, H. Ritter, and G. Heidemann. Acquisition and application of a tactile database. In *IEEE International Conference on Robotics and Automation*, 2007 pages 1517 –1522.

M. Schöpfer, M. Pardowitz, and H. Ritter. Using entropy for dimension reduction of tactile data. In *International Conference on Advanced Robotics*, 2009 pages 1 –6.

M. Seeger. Bayesian methods for Support Vector machines and Gaussian processes. Technical report, Division of Informatics, University of Edinburgh, UK, 1999.

M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Ninth Intl. Workshop on Artificial Intelligence and Statistics*, volume 9, 2003.

M. E. Seligman. Learned helplessness. *Annual Review of Medicine*, 23(5):407, 1972.

B. Settles. Active learning literature survey. Technical report, University of Wisconsin, Madison, 2010.

R. Simpson, E. Lopresti, S. Hayashi, I. Nourbakhsh, and D. Miller. The smart wheelchair component system. *Journal Of Rehabilitation Research And Development*, 41(3B):429–442, 2004.

R. C. Simpson. Smart wheelchairs: A literature review. *Journal Of Rehabilitation Research And Development*, 42(4):423–436, 2005.

R. C. Simpson, D. Poirot, and F. Baxter. The Hephaestus Smart Wheelchair System. *IEEE Transactions on Neural and Rehabilitation Systems Engineering*, 10 (2):118–122, 2002.

R. C. Simpson, E. F. LoPresti, and R. A. Cooper. How many people would benefit from a smart wheelchair? *Journal Of Rehabilitation Research And Development*, 45(1):53–71, 2008.

E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264, 2006.

H. Soh and Y. Demiris. Involving young children in the development of a safe, smart paediatric wheelchair. *ACM/IEEE HRI-2011 Pioneers Workshop*, 2011.

H. Soh and Y. Demiris. When and How to Help: A Probabilistic Model for Learning Assistance by Demonstration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (Accepted)*, 2013.

H. Soh, Y. Su, and Y. Demiris. Online spatio-temporal Gaussian process experts with application to tactile classification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4489–4496, 2012.

H. Soh. iCub Grasp Dataset, 2012a. URL http://www.bitbucket.org/haroldsoh/icub_grasp_dataset.

H. Soh. Online Temporal Learning (OTL) C++ Library, 2012b. URL http://www.bitbucket.org/haroldsoh/otl.

H. Soh and Y. Demiris. Iterative Temporal Learning and Prediction with the Sparse Online Echo State Gaussian Process. In *IEEE International Joint Conference on Neural Networks*, pages 1–8, 2012a.

H. Soh and Y. Demiris. Towards early mobility independence: An intelligent paediatric wheelchair with case studies. In *IROS 2012 Workshop on Progress, Challenges and Future Perspectives in Navigation and Manipulation Assistance for Robotic Wheelchairs*, 2012b.

J. Stark, D. Broomhead, M. Davies, and J. Huke. Delay embeddings for forced systems. ii. stochastic forcing. *Journal of Nonlinear Science*, 13(6):519–577, 2003.

T. Strauss, W. Wustlich, and R. Labahn. Design strategies for weight matrices of echo state networks. *Neural Computation*, 2012 24(12):3246–3276.

F. Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence*, pages 366–381. Springer, 1981.

D. Tefft, P. Guerette, and J. Furumasu. Cognitive predictors of young children's readiness for powered mobility. *Developmental Medicine and Child Neurology*, 41(10):665–670, 1999.

J. Tegin and J. Wikander. Tactile sensing in intelligent robotic manipulation–a review. *Industrial Robot: An International Journal*, 32(1):64–70, 2005.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005. ISBN 0262201623.

M. K. Titsias and M. Lázaro-gredilla. Variational heteroscedastic gaussian process regression. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 841–848, 2011.

V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 2000 12(11):2719–2741.

V. Tresp. Mixtures of Gaussian processes. *Advances in Neural Information Processing Systems*, pages 654–660, 2001.

F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens. Phoneme recognition with large hierarchical reservoirs. In *Advances in Neural Information Processing Systems*, pages 2307–2315, 2010.

J. K. Tsotsos, G. Verghese, S. Dickinson, M. Jenkin, A. Jepson, E. Milios, F. Nuflo, S. Stevenson, M. Black, D. Metaxas, S. Culhane, Y. Ye, and R. Mann. PLAYBOT: A visually-guided robot for physically disabled children. *Image & Vision Computing, Special Issue on Vision for the Disabled*, 16(4):275–292, 1998.

S. Van Vaerenbergh. Kernel adaptive filtering toolbox KAFBOX: a Matlab benchmarking toolbox for kernel adaptive filtering. Grupo de Tratamiento Avanzado de Señal, Departamento de Ingeniería de Comunicaciones, Universidad de Cantabria, Spain, 2012. URL http://sourceforge.net/p/kafbox.

S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría. Kernel Recursive Least-Squares Tracker for Time-Varying Regression. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1313–1326, 2012.

V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, USA, 1995.

V. Vapnik, S. E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. *Advances in neural information processing systems*, pages 281–287, 1997.

S. Vijayakumar, A. D'souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005.

S. Vijayakumar, A. D'Souza, and S. Schaal. LWPR: Locally Weighted Projection Regression library, 2012. URL http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/.

A. S. Weigend. *Time Series Prediction: Forecasting The Future And Understanding The Past*. Sante Fe Institute Series. 1993 Westfield Press.

E. Wen. Time series data, 2011. URL http://web.cecs.pdx.edu/%7Eericwan/time-series-data.html.

R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989 1(2):270–280.

H. Xiao and C. Eckert. Lazy Gaussian Process Committee for Real-Time Online Regression. 2013 In *27th AAAI Conference on Artificial Intelligence*.

H. A. Yanco. A Robotic Wheelchair System: Indoor Navigation and User Interface. *LNCS Assistive Technology and Artificial Intelligence Applications in Robotics User Interfaces and Natural Language Processing*, 1458:256–268, 1998a.

H. A. Yanco. Integrating robotic research: a survey of robotic wheelchair development. In *AAAI Spring Symposium on Integrating Robotic Research*, pages 136–141, 1998b.

H. Yousef, M. Boukallel, and K. Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics—a review. *Sensors and Actuators A: Physical*, 167(2): 171–187, 2011.