

Q-Learning for Robot Control

A thesis submitted for the degree of
Doctor of Philosophy
of The Australian National University.

Chris Gaskett

Bachelor of Computer Systems Engineering H1 (RMIT University)

Bachelor of Computer Science (RMIT University)

Supervisor:

Professor Alexander Zelinsky

2002

Statement of Originality

These doctoral studies were conducted under the supervision of Professor Alexander Zelinsky. The work submitted in this thesis is a result of original research carried out by myself, in collaboration with others, while enrolled as a PhD student in the Department of Systems Engineering at the Australian National University. It has not been submitted for any other degree or award.

Chris Gaskett

Abstract

Q -Learning is a method for solving reinforcement learning problems. Reinforcement learning problems require improvement of behaviour based on received rewards. Q -Learning has the potential to reduce robot programming effort and increase the range of robot abilities. However, most current Q -learning systems are not suitable for robotics problems: they treat continuous variables, for example speeds or positions, as discretised values. Discretisation does not allow smooth control and does not fully exploit sensed information. A practical algorithm must also cope with real-time constraints, sensing and actuation delays, and incorrect sensor data.

This research describes an algorithm that deals with continuous state and action variables without discretising. The algorithm is evaluated with vision-based mobile robot and active head gaze control tasks. As well as learning the basic control tasks, the algorithm learns to compensate for delays in sensing and actuation by predicting the behaviour of its environment. Although the learned dynamic model is implicit in the controller, it is possible to extract some aspects of the model. The extracted models are compared to theoretically derived models of environment behaviour.

The difficulty of working with robots motivates development of methods that reduce experimentation time. This research exploits Q -learning's ability to learn by passively observing the robot's actions—rather than necessarily controlling the robot. This is a valuable tool for shortening the duration of learning experiments.

Acknowledgements

I thank my supervisor, Professor Alexander Zelinsky, and my previous supervisor Dr David Wettergreen, for guiding me through this research. Thank you to all the staff and students of Systems Engineering, past and present, for their constant support and friendship. Thanks also to all of the visitors to the Robotic Systems Laboratory for their valuable advice.

The mobile robot software was built upon the work of Dr Gordon Cheng (1996), and used correlation software developed by Dr Jochen Heinzmann (1997). Luke Fletcher ported Dr Cheng's software to the Nomad, integrated the learning system into the software, participated in the experiments, helped with video editing, and co-authored two papers. The active head software and experiments were the work of myself and Peter Brown. Vision libraries were provided by Dr Sebastien Rougeaux. Orson Sutherland, Leanne Matuszyk, and Harley Truong provided active head system software. Dr Thomas Brinsmead and Peter Brown assisted with the theoretical pendulum model. Dr Simon Thompson assisted with video editing. Karen Montefiore, Gareth Loy, Luke Fletcher, and Dr Simon Thompson assisted with preparation of diagrams. Dr Peter Bartlett and Dr Jonathan Baxter gave valuable advice about learning systems theory. Dr Baxter also supplied some example feedforward neural network source code. Professor Brian Anderson, Professor John Moore, Professor Robert Bitmead, Dr Thomas Brinsmead, and Dr Thomas Moor answered my questions about control systems. The display software for the cart-pole task was based on Professor Charles Anderson's (1998) pole balancing computer game. Dr Gordon Cheng reviewed an earlier draft. James Ashton kept the computers' wheels turning. Jenni Watkins, Marita Rendina, Karen Montefiore, and Rosemary Shepherd arranged everything and provided moral support. Dr Jochen Heinzmann, Dr David Jung, Dr Gordon Cheng, Dr Rochelle O'Hagan, and Andrew Brooks made me feel welcome from the start. My friends from RMIT University taught me the value of teamwork done well.

My family gave me unflagging support throughout the preparation of this thesis. Finally, I thank my wife Rika, for everything.

Chris Gaskett

List of Publications

Gaskett, C., Brown, P., Cheng, G., and Zelinsky, A. (2003), Learning implicit models during target pursuit, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2003)*, Taiwan.

Gaskett, C., Fletcher, L., and Zelinsky, A. (2000a), Reinforcement learning for a vision based mobile robot, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2000)*, Takamatsu, Japan.

Gaskett, C., Fletcher, L., and Zelinsky, A. (2000b), Reinforcement learning for visual servoing of a mobile robot, in *Proceedings of the Australian Conference on Robotics and Automation (ACRA2000)*, Melbourne, Australia.

Gaskett, C., Wettergreen, D., and Zelinsky, A. (1999a), Q-learning in continuous state and action spaces, in *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, Sydney, Australia.

Gaskett, C., Wettergreen, D., and Zelinsky, A. (1999b), Reinforcement learning applied to the control of an autonomous underwater vehicle, in *Proceedings of the Australian Conference on Robotics and Automation (AuCRA'99)*, Brisbane, Australia.

Wettergreen, D., Gaskett, C., and Zelinsky, A. (1998), Development of a visually-guided autonomous underwater vehicle, in *Proceedings of OCEANS98*, IEEE, Nice.

Wettergreen, D., Gaskett, C., and Zelinsky, A. (1999a), Autonomous control and guidance for an underwater robotic vehicle, in *Proceedings of the International Conference on Field and Service Robotics (FSR'99)*, Pittsburgh, USA.

Wettergreen, D., Gaskett, C., and Zelinsky, A. (1999b), Reinforcement learning for a visually-guided autonomous underwater vehicle, in *Proceedings of the International Symposium on Unmanned Untethered Submersibles Technology (UUST'99)*, Durham, New Hampshire, USA.

Contents

| | |
|--|------------|
| Abstract | iii |
| Acknowledgements | iv |
| List of Publications | v |
| 1 Introduction | 1 |
| 1.1 Reinforcement Learning for Robotics | 2 |
| 1.2 Research Objectives | 2 |
| 1.3 Contributions | 3 |
| 1.4 Thesis Organisation | 4 |
| 2 A Review of Reinforcement Learning | 5 |
| 2.1 Reinforcement Learning | 5 |
| 2.2 Reinforcement Learning for Robotics | 8 |
| 2.2.1 Simulation versus Reality | 9 |
| 2.2.2 Model-Based Learning systems | 11 |
| 2.3 Dynamic Programming and Q -Learning | 13 |
| 2.4 Off-Policy Learning | 15 |
| 2.4.1 Off-Policy Learning Techniques | 17 |
| 2.4.2 Practical Off-Policy Learning | 21 |
| 2.5 Q -Learning with Continuous States and Actions | 22 |
| 2.5.1 Continuous States | 22 |
| 2.5.2 Continuous Actions | 27 |
| 2.5.3 Necessary Properties | 27 |
| 2.5.4 Existing Approaches | 31 |
| 2.5.5 Evaluation of Existing Systems | 38 |
| 2.6 Summary | 39 |

| | | |
|----------|--|-----------|
| 3 | A Continuous State and Action Q-Learning Algorithm | 41 |
| 3.1 | Background | 41 |
| 3.2 | The Wire Fitted Neural Network | 43 |
| 3.3 | Training Algorithm | 47 |
| 3.4 | Illustrating Example: Navigating a Boat to a Target Position | 51 |
| 3.4.1 | Results | 52 |
| 3.4.2 | Application of Off-Policy Learning | 55 |
| 3.5 | Summary | 55 |
| 4 | Issues for Continuous Q-Learning | 59 |
| 4.1 | Convergence | 59 |
| 4.2 | The State-Action Deviation Problem | 61 |
| 4.3 | Rising Q | 66 |
| 4.3.1 | A Partial Solution through Advantage Learning | 67 |
| 4.4 | Parameter Sensitivity | 68 |
| 4.4.1 | The Advantage Learning Factor and Discount Factor | 69 |
| 4.4.2 | Neural Network Training Rule | 71 |
| 4.4.3 | Learning Rate | 72 |
| 4.4.4 | Number of Hidden Neurons | 73 |
| 4.4.5 | Number of Wires | 73 |
| 4.4.6 | Thinks per Action | 74 |
| 4.4.7 | Smoothing Factor | 76 |
| 4.4.8 | Derivative Error | 76 |
| 4.5 | Pure-Delayed Reward: The Cart-Pole Problem | 77 |
| 4.5.1 | The Learning Task | 77 |
| 4.5.2 | WFNN Results | 78 |
| 4.5.3 | Discussion | 79 |
| 4.6 | Summary | 82 |
| 5 | Implementation | 85 |
| 5.1 | Managing Unreliable Sensor Data | 85 |
| 5.2 | Managing Real-Time Constraints | 88 |
| 5.3 | Extending Processing Power | 92 |
| 5.4 | Safety Management | 94 |
| 5.5 | Artificial Neural Network Implementation | 96 |
| 5.6 | Wire Fitter Implementation | 97 |

| | | |
|----------|--|------------|
| 5.7 | Summary | 97 |
| 6 | Mobile Robot Experiments | 99 |
| 6.1 | Vision-Based Robot Behaviour | 99 |
| 6.1.1 | 3D Reconstruction and Planning | 100 |
| 6.1.2 | Purposive Vision and Behaviour | 102 |
| 6.2 | Robot System Architecture | 103 |
| 6.3 | Vision System | 107 |
| 6.4 | Learning to Wander | 109 |
| 6.4.1 | Task Representation | 110 |
| 6.4.2 | Results | 112 |
| 6.5 | Learned Visual Servoing | 112 |
| 6.5.1 | Task Representation | 114 |
| 6.5.2 | Results | 114 |
| 6.6 | Application of Off-Policy Learning | 117 |
| 6.7 | Discussion | 120 |
| 6.7.1 | Learning Difficulties | 120 |
| 6.7.2 | Vision Difficulties | 122 |
| 6.7.3 | Relation to Other Work | 124 |
| 6.8 | Summary | 125 |
| 7 | Active Head Experiments | 127 |
| 7.1 | Active Vision | 127 |
| 7.2 | Learned Gaze Control | 130 |
| 7.2.1 | Learned Smooth Pursuit | 130 |
| 7.2.2 | Learned Vestibulo-Ocular Reflex | 131 |
| 7.2.3 | Learned Saccadic Motion | 132 |
| 7.2.4 | Babybot: Learned Saccades, Smooth Pursuit, and VOR/OKR | 133 |
| 7.2.5 | Learned Vergence | 134 |
| 7.2.6 | Discussion | 134 |
| 7.3 | Experimental Platform: HyDrA | 135 |
| 7.4 | Experimental Task | 138 |
| 7.4.1 | Task Representation | 140 |
| 7.5 | Single Degree of Freedom Visual Servoing | 143 |
| 7.5.1 | Learning Process | 143 |
| 7.5.2 | The Training Apparatus | 144 |

| | | |
|----------|---|------------|
| 7.5.3 | Training with Static Targets—Evaluating with Static Targets | 145 |
| 7.5.4 | Training with Static Targets—Evaluating with Swinging Targets . . | 150 |
| 7.5.5 | Training with Swinging Targets—Evaluating with Swinging Targets | 153 |
| 7.5.6 | Training with Swinging Targets—Evaluating with Static Targets . . | 157 |
| 7.6 | Four Degrees of Freedom Visual Servoing | 162 |
| 7.6.1 | Coping with Redundancy | 164 |
| 7.6.2 | Coping with Instability | 166 |
| 7.7 | Discussion | 167 |
| 7.8 | Suggestions for Further Work | 169 |
| 7.9 | Summary | 170 |
| 8 | Conclusions | 173 |
| 8.1 | Summary | 173 |
| 8.2 | Future Directions | 176 |
| A | Example Interface for Dynamic-System Simulators | 179 |
| B | Wire Fitting C++ Source Code | 183 |
| | References | 187 |
| | Index | 201 |

List of Figures

| | | |
|------|--|----|
| 2.1 | A basic architecture for robot control through reinforcement learning | 6 |
| 2.2 | Robot navigation problems in a grid world and a real environment | 10 |
| 2.3 | Archery target | 22 |
| 2.4 | The archery target coarsely discretised by position | 23 |
| 2.5 | Finely discretised archery target | 24 |
| 2.6 | Finely discretised archery target without generalisation by position | 25 |
| 2.7 | Archery target represented through function approximation | 26 |
| 2.8 | Coarsely discretised actions (sightings) | 26 |
| 2.9 | Continuously variable actions (sightings) | 27 |
| 2.10 | Essential capabilities for a continuous state and action Q -learning system | 28 |
| 2.11 | Continuous vs. Piecewise constant | 30 |
| 2.12 | Piecewise constant actions (sightings) | 30 |
| 3.1 | Single network QCON architecture | 42 |
| 3.2 | 'Ideal' architecture | 42 |
| 3.3 | Single network Q -AHC architecture | 42 |
| 3.4 | WFNN architecture | 43 |
| 3.5 | The wire fitting process | 45 |
| 3.6 | The WFNN training algorithm, Parts 1,2, and 3 | 48 |
| 3.7 | Part 1 of the WFNN training algorithm in greater detail | 49 |
| 3.8 | Part 2 of the WFNN training algorithm in greater detail | 50 |
| 3.9 | Part 3 of the WFNN training algorithm in greater detail | 50 |
| 3.10 | Reinforcement learning specification for boat navigation | 51 |
| 3.11 | Boat simulation | 53 |
| 3.12 | Percentage of simulated boats reaching the target | 54 |
| 3.13 | Number of action steps for the simulated boat to reach target n | 54 |
| 3.14 | The action-values for a particular state | 56 |

| | | |
|------|--|-----|
| 4.1 | The action-values for a particular state | 62 |
| 4.2 | Q -learning vs. A -learning | 65 |
| 4.3 | Median performance of the WFNN algorithm for the boat navigation task | 69 |
| 4.4 | Lower quartile (near best case) performance | 70 |
| 4.5 | Interquartile range | 71 |
| 4.6 | Performance versus learning rate | 73 |
| 4.7 | Performance versus number of hidden neurons | 74 |
| 4.8 | Performance versus number of wires | 75 |
| 4.9 | Performance versus number of thinks | 75 |
| 4.10 | Performance versus smoothing factor | 76 |
| 4.11 | Pole balancing | 78 |
| 4.12 | Pole-balancing policy degradation | 80 |
| 5.1 | DFD and pseudocode example for the <i>learn</i> architecture | 86 |
| 5.2 | DFD and pseudocode example for the <i>act-learn</i> architecture | 87 |
| 5.3 | Real-time reinforcement learning timing diagram | 89 |
| 5.4 | DFD and pseudocode example for the <i>act-learn-think</i> architecture | 90 |
| 5.5 | DFD and pseudocode example for the <i>act-learn-think-MT</i> architecture | 91 |
| 5.6 | DFD and pseudocode example for the <i>act-learn-think-C/S</i> architecture | 93 |
| 6.1 | The Nomad 200 learning target pursuit | 104 |
| 6.2 | Uncalibrated camera mounted on the Nomad 200 | 104 |
| 6.3 | Behaviour-based model | 105 |
| 6.4 | The robot's view during the wandering behaviour | 106 |
| 6.5 | A ball has been identified as an interesting object | 107 |
| 6.6 | Carpet template acquisition during initialisation | 108 |
| 6.7 | Camera view illustrating Cheng and Zelinsky's wandering algorithm | 109 |
| 6.8 | Reinforcement learning specification for wandering | 111 |
| 6.9 | A path from a wandering experiment | 113 |
| 6.10 | Reinforcement learning specification for visual servoing | 114 |
| 6.11 | Learning to visually servo | 115 |
| 6.12 | Example target trajectories | 116 |
| 6.13 | Example Step response and motor commands | 116 |
| 6.14 | Misaligned camera | 117 |
| 6.15 | View while pursuing a person | 118 |
| 6.16 | Pursuing a person | 119 |

| | | |
|------|---|-----|
| 6.17 | The interest operator chooses an unexpected target | 123 |
| 7.1 | HyDrA | 136 |
| 7.2 | HyDrA's degrees of freedom | 136 |
| 7.3 | Binocular ophthalmotrope by Ruete | 137 |
| 7.4 | Monocular ophthalmotrope by Wundt | 137 |
| 7.5 | Experimental apparatus | 139 |
| 7.6 | Reinforcement learning specification for the active vision experiments . . | 141 |
| 7.7 | Information taken from the camera view for state and reward calculation . | 141 |
| 7.8 | Example joint and camera axis inclusion configurations | 142 |
| 7.9 | HyDrA and training apparatus | 144 |
| 7.10 | Transition between a static and a swinging target | 146 |
| 7.11 | Static-trained controller pursuing a static target | 147 |
| 7.12 | Static-trained controller pursuing a static target | 148 |
| 7.13 | Static-trained controller following a swinging target | 151 |
| 7.14 | Position data from the static-trained controller following a swinging target | 152 |
| 7.15 | Velocity data from the static-trained controller following a swinging target | 153 |
| 7.16 | Swing-trained controller following a swinging target | 154 |
| 7.17 | Position data from the swing-trained controller following a swinging target | 155 |
| 7.18 | Velocity data from the swing-trained controller following a swinging target | 156 |
| 7.19 | Actual output and model-predicted joint acceleration | 158 |
| 7.20 | Swing-trained controller failing to pursue a stationary target | 159 |
| 7.21 | Swing-trained controller failing to pursue a stationary target | 160 |
| 7.22 | Joint acceleration when exposed to a static change in target position | 162 |
| 7.23 | Visual servoing with four degrees of freedom and binocular vision | 163 |
| 7.24 | Examples of expected and learned motions | 165 |
| A.1 | Use of the WFNN algorithm within Simulink | 180 |
| A.2 | Graph from use of the WFNN algorithm within Simulink | 181 |

List of Tables

| | | |
|-----|--|-----|
| 2.1 | Categorisation of off-policy and on-policy learning algorithms | 16 |
| 7.1 | Summary of single degree of freedom results | 168 |