Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Barry Ridge

Učenje osnovnih funkcionalnih lastnosti predmetov v robotskem sistemu

DOKTORSKA DISERTACIJA

Mentor: prof. dr. Aleš Leonardis

Somentor: doc. dr. Danijel Skočaj

Ljubljana, 2014

University of Ljubljana

Faculty of Computer and Information Science

Barry Ridge

Learning Basic Object Affordances in a Robotic System

DOCTORAL DISSERTATION

Supervisor: prof. dr. Aleš Leonardis

Co-supervisor: assist. prof. dr. Danijel Skočaj

Ljubljana, 2014

Dedicated to those who come after me.

Acknowledgements

First and foremost I would like to thank my supervisor prof. dr. Aleš Leonardis and co-supervisor assist. prof. dr. Danijel Skočaj for giving me the opportunity to do this research, for making it possible to come and live in the beautiful country that is Slovenia, and for supporting me time and again when the going got tough. I am eternally grateful. I would like to thank Radu Hourad, who headed the VISIONTRAIN project which provided my original fellowship, for the opportunity to be part of an exceptionally stimulating early research training programme. I would also like to thank Jeremy Wyatt for having given me the opportunity to work on the CogX project- a fantastic example of what great people can do when they put their minds together, even while spread over a continent. A big thank you must go to Aleš Ude for offering me my current job at the Jožef Stefan Institute which has allowed me to both finish this Ph.D. and continue my research.

I would like to thank all of my former colleagues at the ViCoS lab in Ljubljana. Matej Kristan helped me more times that I can ever remember, often seeming like an oracle on all matters machine learning and otherwise. Roland Perko, the master of low-level vision, showed me plenty of tricks and above all, taught me how to be easy-going. A big thank you to Dušan Omerčević for kick-starting my YouTube career and all the positive energy and great discussions. Luka Fürst got me started with Matlab and reminded me that Vim is the one true text editor. I must thank Matej Artač for providing much of the software and groundwork that got me started working with stereo cameras and 3-D point clouds. Thank you to Aleš Štimec for pointing me in the direction of cross-modal learning, for all of the great help with various pieces of hardware and software, and for many stimulating discussions. I would like to thank Luka Čehovin, without whom I probably would not have had a webpage, SVN access, multi-core computing power, and who knows what else. Thank you to Ondrej Drbohlav for helping me figure out how to get curvature features from surface fitting. Special thanks must also go to the ViCoS coffee club for all the vibrant discussions lubricated by one of life's finer, darker little luxuries.

I would like to thank my current colleagues at the Humanoid and Cognitive Robotics Lab, and more generally, the Department for Automation, Biocybernetics and Robotics, at the Jožef Stefan Institute, who have bid me warm welcome into their midst and continue to provide both buoyant comradery and a vibrant research environment. Thank you to Bojan Nemec, Anton Ružić, Andrej Gams, Jan Babič, Igor Mekjavić, Leon Žlajpah and Igor Kovač for overseeing such a fantastic department. Thank you to my lab-mates Miha Deniša, Robert Bevec, Tadej Petrič, Rok Vuga, Aljaž Kramberger, Luka Peternel, Denis Forte, Nejc Likar and Jernej Čamernik, for their help, support, laughs, lunchtime conversations, and for being such great colleagues. And thank you to Adam McDonnell for being the very embodiment of home away from home. And finally, thank you to Tanja Dragojevič and Marija Kavčič for keeping the whole show running so well.

A very special thank you to my friend Andrej Schulz for the cartoon illustration in the introduction. Sometimes pictures really can paint a thousand words. Or perhaps a little more in this case.

I would like to thank all of the great friends I have made over the years during my time in Ljubljana who have made my passage through here all the lighter. Many have come and gone during that time, and whether the rest of us stay on or move on, I'm quite sure the memories will burn on.

I would like to thank my doctors, Ines Glavan Lenassi dr. med. and Draženka Miličević dr. med., without whose kind and professional care at crucial junctures, I would certainly not have been capable of completing this work.

Last, but of course, most importantly, I would like to thank my wonderful family without whose love and support I would never have come this far.

Barry Ridge
Ljubljana
October 2014

Abstract

One of the fundamental enabling mechanisms of human and animal intelligence, and equally, one of the great challenges of modern day autonomous robotics is the ability to perceive and exploit environmental affordances. To recognise how you can interact with objects in the world, that is to recognise what they afford you, is to speak the language of cause and effect, and as with most languages, practice is one of the most important paths to understanding. This is clear from early childhood development. Through countless hours of motor babbling, children gain a wealth of experience from basic interactions with the world around them, and from there they are able to learn basic affordances and gradually more complex ones. Implementing such affordance learning capabilities in a robot, however, is no trivial matter. This is an inherently multi-disciplinary challenge, drawing on such fields as autonomous robotics, computer vision, machine learning, artificial intelligence, psychology, neuroscience, and others.

In this thesis, we attempt to study the problem of affordance learning by embracing its multi-disciplinary nature. We use a real robotic system to perform experiments using household objects. Camera systems record images and video of these interactions from which computer vision algorithms extract interesting features. These features are used as data for a machine learning algorithm that was inspired in part by ideas from psychology and neuroscience. The learning algorithm is perhaps the main focal point of the work presented here. It is a self-supervised multi-view online learner that dynamically forms categories in one data view, or sensory modality, that are used to drive supervised learning in another. While useful in and of itself, the self-supervised learner can potentially benefit from certain augmentations, particularly over shorter training periods. To this end, we also propose two novel feature relevance determination methods that can be applied to the self-supervised learner.

With regard to robotic experiments, we make use of two different robotic setups, each of which involves a robot arm operating in an experimental environment with a flat table surface, with camera systems pointing at the scene.

Objects placed in the environment can be manipulated, generally pushed, by the arm, and the camera systems can record image and video data of the interaction. One of the camera systems in one of the setups is a stereo camera, and another in the other setup is an RGB-D sensor, thus allowing for the extraction of range data and 3-D point cloud data. In the thesis, we describe computer vision algorithms for extracting both salient object features from the static images and point cloud data, and effect features from the video data of the object in motion.

A series of experiments are described that evaluate the proposed feature relevance algorithms, the self-supervised multi-view learning algorithm, and the application of these to real-world object push affordance learning problems using the robotic setups. Some surprising results emerge from these experiments and as well as those, under the conditions we present, our framework is shown to be able to autonomously discover object affordance categories in data, predict the affordance categories of novel objects and determine the most relevant object properties for discriminating between those categories.

Key words:
affordances; affordance learning; self-supervised learning; multi-view learning; cross-modal learning; multi-modal learning; feature relevance determination; online learning; cognitive robotics; developmental robotics

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.1.1 What is an Affordance? . . . 3

1.1.2 Cognitive and Developmental Robotics . . . 5

1.2 An Object Affordance Learning Scenario . . . 6

1.3 Contributions . . . 8

1.4 Thesis Outline . . . 10

2 A History of Affordances **13**

2.1 Affordances in Ecological Psychology . . . 14

2.1.1 Formalising Affordances . . . 15

2.2 Affordances in Studies on Humans and Animals . . . 19

2.2.1 Studies on Humans . . . 19

2.2.2 Studies on Non-Human Animals . . . 22

2.3 Affordances in Artificial Intelligence . . . 25

2.3.1 The Symbol Grounding Problem . . . 25

2.3.2 Constructivist Learning . . . 26

2.4 Affordances in Vision . . . 29

2.4.1 Function-Based Object Recognition . . . 29

2.4.2 Object Categorisation . . . 30

2.5 Affordances in Robotics . . . 31

2.5.1 Pushing and Pulling Objects 31
2.5.2 Grasping Objects 34
2.5.3 Other Forms of Affordances in Robotics 35
2.6 Chapter Summary 37

3 Framing the Object Affordance Learning Problem 39
3.1 Formalising the Problem 39
3.2 Framing the Problem within Machine Learning 41
3.2.1 Supervised Learning or Unsupervised Learning? 42
3.2.2 Continuous/Online Learning 43
3.3 Requirements for Our Affordance Learner 45
3.4 Chapter Summary 47

4 Self-Supervised Multi-View Learning 49
4.1 Multi-View Learning 51
4.1.1 Representing a Data View or Sensory Modality 51
4.1.2 Connecting Data Views 56
4.1.3 Training an Unsupervised Multi-View Learner 60
4.2 Unsupervised Multi-View Discriminative Learning 63
4.2.1 Regression 64
4.2.2 Classification 65
4.3 Self-Supervised Multi-View Discriminative Learning 71
4.3.1 Learning Vector Quantization 72
4.3.2 Training LVQ Classifiers with Probabilities instead of Labels 75
4.3.3 Measuring Similarity Between Data Views 77
4.3.4 Self-Supervised Online Multi-View Training Algorithm . . 78
4.4 Chapter Summary 81

5 Feature Relevance Determination 83

5.1 Relevance Determination for LVQ . . . 84
5.1.1 RLVQ . . . 85
5.1.2 GRLVQ . . . 85
5.2 Relevance Determination for LVQ using Fisher Criterion Score . . 86
5.2.1 First Proposed Algorithm: FC1 . . . 87
5.2.2 Second Proposed Algorithm: FC2 . . . 88
5.3 Applying Relevance Determination to the Self-Supervised Multi-View Learner . . . 92
5.3.1 During Training . . . 92
5.3.2 At Classification Time . . . 92
5.4 Chapter Summary . . . 93

6 Robot and Vision Systems 95
6.1 Experimental Platforms . . . 96
6.1.1 Katana/Camera Setup . . . 96
6.1.2 KUKA-LWR/Kinect Setup . . . 99
6.2 Visual Feature Extraction . . . 100
6.2.1 Object Features . . . 101
6.2.2 Object Effect Features . . . 108
6.3 Chapter Summary . . . 115

7 Experimental Results 117
7.1 Experiments on Feature Relevance Determination Algorithms . . 117
7.1.1 Data . . . 118
7.1.2 Evaluation Procedure . . . 118
7.1.3 Algorithm Setup . . . 120
7.1.4 Results . . . 120
7.1.5 Summary . . . 124
7.2 Experiments on Self-Supervised Multi-View Learning Algorithms . 126

7.2.1 Data 126
7.2.2 Evaluation Procedure 128
7.2.3 Experiments 129
7.2.4 Results: Three-Class Synthetic Dataset 131
7.2.5 Results: Five-Class Synthetic Datasets Follow-up Study . . 141
7.2.6 Summary 144
7.3 Object Push Affordance Experiments with Katana/Camera Setup 145
7.3.1 Data 145
7.3.2 Evaluation Procedure 147
7.3.3 Experiments 149
7.3.4 Results: 3-D + 2-D Object Features Dataset 151
7.3.5 Results: 3-D + 2-D Object Features Dataset Follow-Up Study 157
7.3.6 Results: 3-D Object Features Dataset 160
7.3.7 Summary 163
7.4 Object Push Affordance Experiments with KUKA-LWR/Kinect Setup 163
7.4.1 Data 164
7.4.2 Staged Feature Relevance Determination 166
7.4.3 Results 166
7.4.4 Summary 170
7.5 Chapter Summary 170

8 Conclusion **173**
8.1 Thesis Summary 173
8.2 Summary of Contributions 176
8.3 Future Work 176

References **177**

Appendix A **197**

Appendix B **199**

B.1 Uvod . . . 199

B.1.1 Motivacija . . . 199

B.1.2 Scenarij za učenje funkcionalnih lastnosti predmetov . . . 203

B.1.3 Prispevki . . . 205

B.2 Samo-nadzorovano več-modalno učenje . . . 206

B.3 Določanje ustreznosti značilnic . . . 209

B.4 Robotski sistem za učenje funkcionalnih lastnosti predmetov . . . 210

B.5 Rezultati poizkusov . . . 212

B.6 Zaključek . . . 214

Published Work **217**

Declaration **221**

List of Tables

3.1 Weng's comparison of various learning approaches. 44

3.2 Hard online learning requirements. 45

3.3 Self-supervised learning requirements. 46

7.1 Attribute list for datasets in Section 7.1.1. 118

7.2 Evaluation of various supervised feature relevance determination algorithms. 122

7.3 FC1GLVQ & FC2GLVQ versus SVM. 123

7.4 Summary of dataset from KUKA-LWR/Kinect experiment. 165

List of Figures

1.1 Cartoon illustration of the affordance learning problem. 2
1.2 Object affordance learning experimental setup. 7
1.3 The main idea behind our affordance learning framework. 8

2.1 Chimpanzees using wooden crates to retreive bananas in the experimental studies of Wolfgang Köhler [14]. 23
2.2 A small octopus using a nut shell and a clam shell for protection [17]. 25

3.1 Potential affordance learning models. 40
3.2 Expanding on the main object affordance learning idea. 41

4.1 Vector quantization Voronoi diagram. 52
4.2 Multi-view vector quantization. 57
4.3 Cross-view Hebbian projection. 59
4.4 Unsupervised multi-view training. 61
4.5 Cross-view projection of meta-clusters. 68
4.6 Cross-view classification. 69
4.7 Self-supervised multi-view training. 76

5.1 First proposed feature relevance determination technique. 87
5.2 Fisher criterion score failure over multi-modal distributions. . . . 89
5.3 Second proposed feature relevance determination technique. . . . 90

6.1 Katana/Camera setup system architecture. 96

6.2 Learning environment for the Katana/Camera setup. 98

6.3 Sample rolling versus non-rolling objects. 99

6.4 Learning environment for the KUKA-LWR/Kinect setup. 100

6.5 Object feature extraction pipeline in the Katana/Camera setup. . 101

6.6 Multi-modal object segmentation pipeline in Katana/Camera setup.102

6.7 Object segmentation and surface fitting in Katana/Camera setup. 105

6.8 Dividing 3-D object point cloud into parts in KUKA-LWR/Kinect setup. 107

6.9 Object tracking in Katana/Camera setup. 110

6.10 Object tracking in KUKA-LWR/Kinect setup. 111

6.11 Object tracking refinement: rolling object. 112

6.12 Object tracking refinement: non-rolling object. 113

7.1 Feature relevance bar plots for fully-supervised learners on various datasets. 125

7.2 Projections from three-class dual-view synthetic dataset. 127

7.3 Projections from five-class dual-view synthetic dataset. 128

7.4 Online class discovery results for various unsupervised SOM-based multi-view learners. 132

7.5 Online classification results for various unsupervised multi-view SOM-based learners. 133

7.6 Fully-supervised versus self-supervised LVQ1-based and GLVQ-based learners. 135

7.7 Fully-supervised versus self-supervised RLVQ1-based and GRLVQ-based learners. 136

7.8 Fully-supervised versus self-supervised FC2LVQ1-based and FC2GLVQ-based learners. 137

7.9 Feature relevance bar plots for learning on a synthetic multi-view dataset. 138

7.10 Comparison of multiple self-supervised learners for two epochs of training. 139

7.11 Comparison of multiple self-supervised learners for 100 epochs of training. 140

7.12 Class discovery results for 10-folds cross validation on the 500 sample 5-class synthetic dataset (left graph) and for 5-folds cross validation on the 50000 sample 5-class synthetic dataset (right graph). 142

7.13 Class prediction results for 10-folds cross validation on the 500 sample 5-class synthetic dataset (left graph) and for 5-folds cross validation on the 50000 sample 5-class synthetic dataset (right graph). 143

7.14 Feature relevance determination results for 10-folds cross validation on the 500 sample 5-class synthetic dataset (left graph) and for 5-folds cross validation on the 50000 sample 5-class synthetic dataset (right graph). 143

7.15 Sample rolling versus non-rolling objects. 146

7.16 Projection of Katana/Camera experimental dataset onto two feature dimensions. 147

7.17 Sample object interactions from Katana/Camera experimental dataset. 148

7.18 Online affordance class discovery results for unsupervised multi-view SOM-based learners. 151

7.19 Online object affordance prediction of unsupervised multi-view SOM-based learners over two training epochs. 152

7.20 Online object affordance prediction of fully-supervised and self-supervised learners over two training epochs. 153

7.21 The effect of prototype culling on a supervised learner. 154

7.22 The effect of prototype culling on a self-supervised learner. 155

7.23 Comparing unsupervised and self-supervised learners, with and without feature relevance determination at classification time. 155

7.24 Comparing unsupervised and self-supervised learners with prototype culling, with and without feature relevance determination at classification time. 156
7.25 Feature relevance bar plots for fully-supervised learners. 157
7.26 Feature relevance bar plots for learning on the object affordance dataset. 158
7.27 Class prediction results for Katana/Camera 3-D + 2-D object features follow-up study. 159
7.28 Feature relevance results for Katana/Camera 3-D + 2-D object features follow-up study. 160
7.29 Class prediction results for Katana/Camera 3-D object features experiment. 162
7.30 Input view feature relevance results for Katana/Camera 3-D object features experiment. 162
7.31 Segmented 3-D object point clouds from KUKA-LWR/Kinect experiment. 164
7.32 Object trajectories and trajectory convex hulls from KUKA-LWR/Kinect experiment. 165
7.33 Staged feature relevance determination results from KUKA-LWR/Kinect experiment. 167
7.34 Class discovery and prediction results for KUKA-LWR/Kinect experiment. 167
7.35 Selected feature relevance determination results from KUKA-LWR/Kinect experiment. 168
7.36 Specific class discovery and prediction results for KUKA-LWR/Kinect experiment. 169

B.1 Postavitev sistema za učenje funkcionalnih lastnosti predmetov. . 204
B.2 Glavna ideja našega sistema za učenje funkcionalnih lastnosti predmetov. 204
B.3 Več-modalna projekcija meta-gruč 209

B.4 Proces segmentacije predmeta. 211

B.5 Sledenje predmetu 212

B.6 Sprotno napovedovanje funkcionalnih lastnosti predmetov. Primerjava med različnimi tipi samo-nadzorovanega učenja za dve učni epizodi. 213

List of Algorithms

1 Unsupervised Multi-View Learner Training 61
2 Unsupervised Multi-View Learner Online Update 63
3 Cross-View Regression 64
4 Cross-View Classification 70
5 Self-Supervised Multi-View Learner Online Update 80
6 Multi-Modal Object Segmentation 103
7 Object Tracking Refinement 109

Chapter 1

Introduction

1.1 Motivation

If robots are ever going to live up to their potential and permeate our daily lives in any meaningful sense, they are going to need to be able to learn and to adapt to their environments autonomously. The reason for this is deceptively simple in that it boils down to the fact that the world around us is deceptively complex. The world around us is so complex, in fact, that almost paradoxically, we can end up in situations where tasks that would be challenging or even impossible for humans to perform end up being trivial for robots to perform; whereas tasks that are comparatively simple for humans to perform, turn out to be exceedingly complex or impossible for robots to replicate. By way of explanation, while at the present time it is routine to employ robots to perform involved, but well-defined, tasks in controlled environments such as car factories, it is far less feasible to manually program robots to account for all of the possible situations they might encounter in more uncontrolled environments such as modern family homes. Where the fortunate engineer of a robot tasked with repeatedly soldering the same complex circuit design into the satellite navigation system of a car might have little to worry about, since the circuit design does not change very often, the hapless designer of a multi-functional home robot will more than likely despair when it fails to find the white television remote control in its new home, having only ever been programmed to locate black ones.

The answer to this lies in learning, but endowing robots with the ability to learn is a deep, multi-faceted problem that spans many diverse research fields such as artificial intelligence, machine learning, computer vision, psychology, neuroscience and others. One of those facets, that of object affordance learning, forms the subject of this thesis. As human beings, the ability to learn how objects in the world around us behave when we manipulate them, that is, the ability to learn what objects in the world *afford* us, is fundamental to our broader ability to learn more complex ideas; it is, in other words, fundamental to the development of our intelligence. If robots had a similar ability, then it stands to reason that it would be fundamental to their development of more complex abilities also. However, just as it is non-obvious precisely how best to approach the general autonomous robotic learning problem, so too is it unclear how the more specific object affordance learning problem should be approached. As humans, we are so accustomed to sub-consciously learning about and exploiting affordances from an early age with relative ease, that we tend to have little appreciation of the underlying complexity of the problem, as well as of what a robotic system would have to be capable of in order to solve it, an idea that is illustrated in cartoon form in Figure 1.1. Fortunately, there are a number of possible angles of attack, some of which we explore in this thesis. We begin with an exploration of the meaning of the word "affordance" itself.

Figure **1.1:** How should robots go about learning the basic rules for manipulating objects in their environment?[1]

[1]With thanks to Andrej Schultz for this original illustration.

1.1.1 What is an Affordance?

The term *affordance* was coined by the psychologist J.J. Gibson [65, 1979 1st edition] to describe the interactive possibilities offered to an agent by its environment, e.g., "a ball affords being rolled on the ground" or "a light switch affords the illumination of a light bulb". To quote Gibson himself regarding his inception of the term,

> "The *affordances* of the environment are what it offers the animal, what it *provides* or *furnishes*, either for good or ill. The verb to *afford* is found in the dictionary, but the noun *affordance* is not. I have made it up. I mean by it something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment." [66, p. 127, 1986 2nd edition].

This "complementarity of the animal and the environment" turns out to be quite important. When discussing how a cognitive agent like a robot might learn affordances, a number of important considerations present themselves, namely, the morphology, or shape, of the agent, the morphology of the environment and the objects that inhabit it, the environmental context and object contexts, e.g., how an object is positioned in the environment, as well as the agent context, and the possible actions that are available to the agent in the present context. Again quoting Gibson,

> "Different layouts afford different behaviours for different animals, and different mechanical encounters. ... Knee-high for a child is not the same as knee-high for an adult, so the affordance is relative to the size of the individual." [66, p. 128].

This is why the ability to *learn* affordances dynamically and developmentally is fundamental. In human beings, or indeed in any other type of organism, no two bodies are exactly the same and, moreover, bodies change dramatically during the lifetime of an individual, thus the necessity to be able to adapt to and learn affordances continuously. Each and every cognitive organism in the natural world has co-evolved with its environment over the course of evolutionary history and co-developed with its environment over the course of its own lifespan such that its perceptual systems and actuators are uniquely built for it, and it alone, to recognise and use the environmental affordances it will need to exploit in order to survive and replicate. To paraphrase a common proverb, what's good for Peter may *not* be good for Paul: what one type of organism perceives as an affordance it can exploit, another type of organism may not be able to exploit at all, or even perceive at all. This is because their particular blends of sensors and actuators may differ substantially from one another.

The natural world abounds with examples of this. A dog will never be able to learn grasping affordances of objects when using one of its paws, because its paws do not have opposable thumbs. A cow might see a patch of grass and recognize that it affords being eaten, whereas an ant crawling around near the cow's feet might see a single blade of grass in the same patch and recognize that it affords being crawled upon. The cow will never see a blade of grass as an object that can support its weight and the ant will never consider digesting a whole patch of grass. A recent study of the drosophila fly [16] provided a striking illustration of how the lack of ability to perceive the current environmental context and what it affords, can mean the difference between life and death. The visually mediated motor planning mechanisms of the drosophila fly usually afford it escape from an incoming threat. When a person attempts to swat a fly with a newspaper, the visual mechanisms of the fly sense the newspaper approaching at speed and before the fly leaps to safety, it first calculates the location of the newspaper, then adjusts the position of its body and its wings to a "take-off" position, before flying in the opposite direction. However, if the person were to sneak up on the fly by moving the newspaper slowly towards it, the fly would not take these pre-flight measures because its visual mechanisms are not nearly as adept at perceiving slow incoming movements than they are at perceiving fast movements, and so the person would stand a much better chance of swatting the fly successfully.

Why is this important when attempting to design a robot that is capable of affordance learning? The point being made here is that there is a strong dependency between what a system is capable of either perceiving, using or learning in terms of affordances, and the sensors, actuators and cognitive mechanisms available to it. Moreover, even very small changes in setup or differences between separate systems can have a dramatic impact. For example, obtaining good 3-D point cloud data of small objects using a stereo camera depends on a number of variables including the baseline of the camera, object texture, lighting conditions, the camera's distance from the objects, etc. This can mean the difference between perceiving the object as being curvy or flat, bumpy or smooth, or in extreme circumstances, present or non-present. The lesson here is that we should be mindful of these factors and others when constructing such a system. Moreover, although it is essential to have the right sensor and actuation systems in place, we would probably also be well advised to try to make the learning mechanism as general and plastic as possible so that it can adapt itself to changes in the underlying sensor and actuation systems or other unforeseen circumstances without requiring re-programming.

1.1.2 Cognitive and Developmental Robotics

As indicated in the title of this thesis, we are chiefly concerned with how a robot might discover and learn to exploit basic affordances of objects in its environment. By using its sensors to observe objects, both statically and while in motion, and by using its actuators to interact with them, a robot may infer relationships between the morphologies of objects, the different actions used to interact with them and the way they behave during interaction. If the information that is used to infer such relationships is rich enough and these relationships are learned effectively, then it should be possible to generalize over the relationships and aid the robot in making predictions in novel situations by applying such generalized knowledge. In other words, the robot would perform an action, e.g. "push the centre of the object", on similar objects, e.g. a football and a tennis ball, and observe what happens afterwards- in the case of our example, the objects would roll away from the effector because they are round. When the system subsequently encounters an orange, it should be able to infer that it will also roll away when pushed in its centre, based on what it observed in the previous situations pushing similar round objects. Equally, if the system were to perform the same centre-pushing action on cereal boxes and books and saw that they slide forward slightly, it should be able to predict, perhaps, that packs of playing cards will move similarly when pushed the same way because they are shaped similarly.

Recent years have seen a surge of activity in the area of developmental robotics [105], a trend that can be seen to underscore the desire to move away from task-specific robotic systems and towards more robust, adaptable platforms and architectures. Desirable traits of such systems include the capacity to construct new concepts from previously learned or known ones, the ability to actively learn via interaction with a tutor or another knowledgeable entity, and indeed via interaction with the environment and objects in the environment, etc. Naturally, these are difficult problems that are unlikely to be amenable to wholesale solutions, but many interesting, more tractable sub-problems can be identified, one of which is the issue of affordance learning.

Another important issue for developmental robotics is that of *continuous* or *on-line* learning. A continuous learning system, rather than deriving its world knowledge from a single batch training procedure, iteratively updates its knowledge representations throughout the course of its lifespan. Given the nature of a developmental robotic system operating in a real world environment, where new situations are being encountered perpetually, it stands to reason that the system will not necessarily have access to data from past situations at any given moment. This scenario would seem to preclude the use of batch learning, which assumes that all training data are available at the same time, and favour the use of a continuous learning mechanism.

It would therefore appear to make sense, when designing a developmental robotic system that is capable of learning object affordances, to try to design it so that it can learn in a continuous manner. Later in Chapter 3 we provide a discussion of how we developed our research on affordance learning with this consideration in mind. In the following section, in order to ground the concepts discussed above with a practical example, we provide a brief overview of our particular object affordance learning scenario before expanding on the core research themes in detail later in the thesis.

1.2 An Object Affordance Learning Scenario

Naturally, different researchers place the emphasis on different aspects of the affordance learning problem depending on the particular problems they are trying to solve and the equipment available to them. In our particular case, we have primarily made use of the setup depicted in Figure 1.2. It consists of a robotic arm attached to a table surface with various types of cameras observing the scene from fixed positions. This setup is described in more detail in Chapter 6. Objects may be placed on the work surface, the robotic arm can interact with them, and the cameras can record visual data of the interaction.

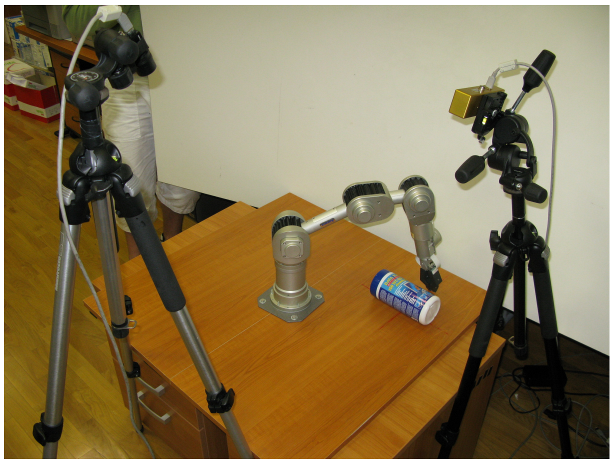


Figure **1.2:** Object affordance learning experimental setup.

Our priority was to focus on the core problem of object affordance learning while attempting to avoid some of the more purist related issues stemming from computer vision and autonomous robotics that might serve to complicate matters. Thus, using such a setup, as opposed to a mobile robot for instance, allowed us to place certain constraints on the nature of the interactions that aided the data acquisition process. One of our main interests was in tracking objects on the work surface when they are pushed, grasped, or otherwise interacted with by the robotic arm. Therefore, from a computer vision perspective, having statically positioned cameras reduced the number of potential issues that could interfere with tracking the objects such as motion blur, occlusion and scale invariance. Having the arm statically positioned in the workspace also helped circumvent camera and environment localization issues. Static, controlled light sources were a further consideration that aided in both object tracking and segmentation. Even the work surface itself, it being a flat wood-textured surface, was helpful in this regard. Having a flat table table surface present in the 3-D point cloud data acquired from a stereo camera actually facilitated the background subtraction process for object segmentation, since it is easy to isolate using surface fitting. Again, the above matters are explored in more detail in Chapter 6.

This setup allowed us to implement and explore the main idea behind our approach to object affordance learning, which is visualized in Figure 1.3. When an object is placed in the workspace we may gather visual data, such as an image of the object and its 3-D point cloud from stereo, and extract salient object features from the data. The arm may then interact with the object and video

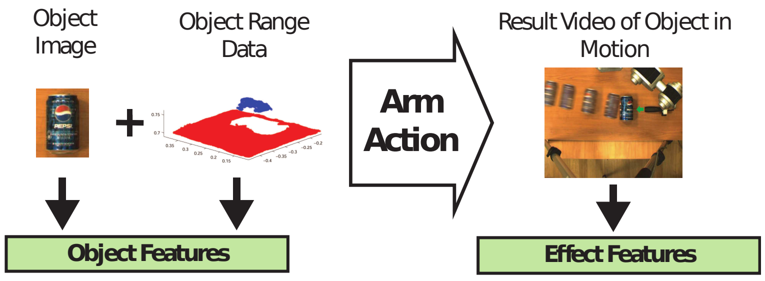


Figure **1.3:** The main idea behind our affordance learning framework.

is recorded of the interaction while the object is in motion. Effect features may be extracted from tracking the object in the video. Our main objective was to form some type of model and learning algorithm for this scenario such that the robotic system, when presented with objects, would be able to gradually learn how they behave from experience interacting with them. When presented with a new object, it should be able to predict how that object will behave in terms of possible effects of actions grounded in effects features, either as a class prediction or as effect feature predictions or otherwise, based on its model and object feature observations. Ideally, given feature data from such interactions, we wanted the system to be able to form its own affordance models from a naive starting point developmentally, at least to some extent. The next section provides an overview of the main contributions that emerged from our research based on these ideas.

1.3 Contributions

The primary academic contributions of this work are as follows:

- **A self-supervised multi-view learning algorithm is proposed that dynamically identifies classes in one data view while using them to drive online supervised learning in another data view.**[2] The approach is based on vector quantization using codebooks of prototypes to represent different sensory modalities (or data views) cross-view connected via a Hebbian mapping. The algorithm performs online clustering in each of the separate modalities while the Hebbian mapping records data co-occurrences between them, allowing for cross-view projections that form the driving force behind the self-supervisory aspect of the algorithm. The self-supervision is based on the learning vector quantization (LVQ) paradigm and training can proceed without the need for class labels, relying instead on cross-view class probabilities.

- **Two novel feature relevance determination algorithms for learning vector quantization are proposed to augment the self-supervised learning algorithm.**[2] Each of these is based on the application of the Fisher criterion score in individual feature dimensions, with one of them taking estimates of the score based on the global positioning of codebook prototypes and the other taking estimates based on the local positioning of nearest-neighbour prototypes. Both of these may be applied to a broad range of LVQ-based algorithms. We also demonstrate how the local method may be applied during online training of the self-supervised learner when only class probabilities as opposed to class labels are available.

- **A robotic system with appropriate actuation and visual feature extraction mechanisms has been developed, used to perform real-world experiments on object affordance learning and to test our learning algorithms on the resulting data.** Building on existing computer vision techniques, we developed algorithms to aid both object shape feature extraction from static images and object effect feature extraction from video. In the former case, this involved the development of a multi-modal feature segmentation technique that segmented objects both from image data and 3-D point cloud data. In the latter case, objects were tracked in video data using a particle filter, and the results were refined and stabilized using colour histogram back-projection such that local object appearance changes could be analysed.

- **Experimental results demonstrating how, when comparing fully-supervised and self-supervised LVQ-based learning under certain conditions, the additional information provided by self-supervision from a separate data view or sensory modality can provide better performance than fully-supervised learning with ground truth labels.** The conditions referred to here are hard online learning constraints discussed later in Chapter 3 of this thesis. The results

[2]Source code has been released under the GNU/GPL license at `https://github.com/barryridge/SSLVQ`.

manifest themselves over short-term training periods where, under the hard online learning constraints, the proposed multi-view self-supervised learning algorithm holds the advantage of dynamic prototype labelling over fully-supervised learning which, lacking prior information, must label prototypes arbitrarily. This is discussed further in the experimental results of Chapter 7.

1.4 Thesis Outline

Chapter 2 discusses the history and related work on affordances from the perspectives of various research fields such as ecological psychology, autonomous robotics, artificial intelligence, machine learning and computer vision.

Chapter 3 describes our particular basis for approaching the affordance learning problem including how we chose to formalize an affordance, what our functional requirements for an affordance learning algorithm were, and how we set about fulfilling them from a machine learning perspective. Key to our exposition are the concepts of multi-view and self-supervised learning. Thus, this chapter includes a high-level discussion on self-supervised learning and how we distinguish it from both unsupervised and supervised forms of learning, as well as how a multi-view or multi-modal perspective on the problem can help achieve such self-supervision.

Chapter 4 focuses on the development of our proposed self-supervised multi-view learning algorithm that we employ for online learning of object affordances. It is broken into three main sections. In the first section, we discuss how sensory modalities may be represented via vector quantization using codebooks of prototypes, we show how these codebooks may be cross-view connected with a Hebbian co-occurrence mapping, and we describe an general algorithm for training such a structure online. A key concept that the multi-view learning hinges upon, known as Hebbian projection, is also described here. In the second section, we show how Hebbian projection may be used to implement both unsupervised regression and unsupervised classification within the general framework. We focus more attention on unsupervised classification, which involves meta-clustering prototypes in an output data view into class clusters, which are mapped back to prototypes in an input data view to be used as class labels. In the third section, we describe how the learning vector quantization paradigm can be used to enact self-supervision in the input data view and show how update rules may be derived for that purpose using class probabilities as opposed to actual class labels, which facilitates efficiency during training.

Chapter 5 addresses our proposed feature relevance determination techniques and how we employed them for use in the self-supervised learning algorithm from Chapter 4. In the first part of this chapter, we review two popular feature relevance determination methods for learning vector quantization. In the second, we describe why the Fisher criterion score is a useful metric for feature relevance determination, and go on to introduce our two proposed algorithms on that basis.

Chapter 6 describes the robotic and vision systems used as a basis for our affordance learning experiments, as well as the various algorithms that were used in their implementation. We made use of two different setups, the first of which consisted of a 5-DOF robotic arm, an RGB monocular camera, and a greyscale stereo camera, and the second of which consisted of a 7-DOF robotic arm and an RGB-D depth sensor. Much of Chapter 6 is devoted to outlining the visual feature extraction techniques that provided both object features and effect features in each of these setups. Generating object features involved segmenting objects from both image data and 3-D point cloud data and extracting shape features from the segmentations, whereas, when generating effects features, particle filter trackers were employed to track object motion in video, and features describing both global object workspace motion and local object appearance changes were extracted from the resulting data.

Chapter 7 details experimental results divided into four sections. In the first section we describe experiments on the feature relevance determination algorithms from Chapter 5 in a fully-supervised setting, using both synthetic data and well-known datasets, such that they can be evaluated independently of the self-supervised algorithm. In the second section we evaluate the self-supervised algorithm from Chapter 4 using synthetic data under various different conditions and using various different combinations of feature relevance determination and base learning rules. The third and fourth sections describe object push affordance learning experiments that were performed using each of the robotic setups described in Chapter 6 respectively, and describe the results of applying our self-supervised learning framework in that context.

Chapter 8 provides concluding thoughts and details potential future research directions.

Chapter 2

A History of Affordances

In this chapter we set about reviewing the history of affordances broadly from the perspectives of ecological psychology, artificial intelligence, vision, and robotics. Ecological psychology, the field from which the term emerged, is important to us not simply out of historical interest, but because much work has been done therein on formalising affordances and providing suitable definitions on what affordances actually are. Studying this will help us to settle on a suitable formalisation for affordances that will both expand our understanding of the nature of the affordance learning problem and subsequently aid us in attempting to solve it, or partially solve it, within the contexts of more applicative domains.

Artificial intelligence is a field in which affordance ideas are often used implicitly as atomic concepts in higher-level reasoning where action-based ideas like "pick up the ball" are dealt with symbolically. Naturally, the question of how these symbols are ascertained in the first place presents itself. This question, immortalized as *the symbol-grounding problem*, may find its solution in autonomous robotics where robotic systems can go out into the real world and learn such basic affordance symbols through experience interacting in the environment. Other fields such as machine learning, computer vision and neuroscience, for example, are also highly important in attempting to realise such goals and solve the associated problems, and we will refer to them as necessary in the discussion that follows in this chapter before returning to some of them in more detail later in the thesis.

2.1 Affordances in Ecological Psychology

Ecological psychology, broadly speaking, attempts to deal with the relationship between the knower and the known, or more practically speaking, the organism and the environment. The term, though associated with a number of different branches of the broader field of psychology, derives primarily from two main strands therein, those are the works of James J. Gibson [65, 66] and Roger G. Barker [4]. We focus here on the Gibsonian interpretation and the studies which stemmed from there, as that is the primary point of inception of the affordance concept. The affordance concept itself indeed became a topic of central importance to ecological psychology, an idea that had the potential to tie together otherwise disparate fields. Reed [142], for example, took the primary point of ecological psychology to be the linkage between affordances and natural selection:

> "The fundamental hypothesis of ecological psychology . . . is that affordances and only the relative availability (or non-availability) of affordances create selection pressure on animals; hence behaviour is regulated with respect to the affordances of the environment for a given animal" [142, p. 18]

J.J. Gibson, for his part, opines here on the link between affordances and niches in the natural world:

> "Ecologists have the concept of a niche. A species of animal is said to utilize or occupy a certain niche in the environment. This is not quite the same as the habitat of the species; a niche refers more to how an animal lives than to where it lives. I suggest that a niche is a set of affordances." [66, p. 128]

and on the relationship between the physical and mental worlds:

> "...an affordance is neither an objective property nor a subjective property; or both if you like. An affordance cuts across the dichotomy of subjective-objective and helps us to understand its inadequacy. It is equally a fact of the environment and a fact of behaviour. It is both physical and psychical, yet neither. An affordance points both ways, to the environment and to the observer." [66, p. 129]

Gibson, however, rejected ideas of indirect perception, information processing and cognitivist views of human behaviour in favour of what he believed to be direct perception envisaged by his conception of affordances:

> "The perceiving of an affordance is not a process of perceiving a value-free physical object to which meaning is somehow added in a way that no one has been able to agree upon; it is a process of perceiving a value-rich ecological object." [66, p. 140]

Given what we know today, this is a difficult position to defend. Having said that, as we shall see from the various formalisations of affordances that came to follow later, Gibson's original interpretation is not entirely irreconcilable with more practical modern approaches. We begin here by taking a look at the main theoretical formalisations of affordances before moving on to reviewing the experimental studies in ecological psychology and related disciplines.

2.1.1 Formalising Affordances

If we are to have any hope of designing and realizing a robotic system that can autonomously learn about affordances, we need to be able to define what affordances actually are and to model them well enough such that we can work with them algorithmically. There have been a number of attempts made in the literature at arriving at a suitable formalisation of affordances [26, 68, 117, 156, 157, 167, 170, 182, 195]. Chief amongst them are those of Turvey [182], Stoffregen [170], Chemero [26], and more recently, Şahin *et al.* [156].

2.1.1.1 As Dispositions of Things

In his 1992 paper, Turvey [182] set out to seek an understanding of affordances given their importance as a fundamental aspect of prospective control in animal activity. He proposed an interpretation of affordances as *dispositions*, or properties, of things that are potentials or possibilities. Moreover, he considered such dispositions as having natural complements such that the dispositions become actualized when combined with their complements. Thus, Turvey's informal definition of an affordance was

> "An affordance is a particular kind of disposition, one whose complement is a dispositional property." [182]

an example of which might be

> "The disposition p of salt to be soluble rests with the fact that it is a lattice of electrically charged ions bound by an electrical attraction between opposite charges that can be eliminated by a liquid with a high dielectric constant." [182]

Here, the high dielectric constant of the liquid acts as a complement to the disposition of the salt to be soluble when combined with the liquid.

He also provided a more formalised definition as follows:

Definition. Let W_{pq} (e.g. a person-climbing-stairs system) $= j(X_p, Z_q)$ be composed of different things Z (person) and X (stairs). Let p be a property of X and let q be a property of Z. Then p is said to be an affordance of X and q the effectivity of Z (i.e., the complement of p), if and only if there is a third property r such that

(i) $W_{pq} = j(X_p, Z_q)$ possesses r

(ii) $W_{pq} = j(X_p, Z_q)$ possesses neither p nor q

(iii) Neither Z nor X possesses r.

Note here that p is defined to be a property of the stairs that assumes the form of an affordance under certain conditions.

2.1.1.2 As Properties of the Animal-Environment System

A common criticism of Turvey's formalisation is that it places too much emphasis on the environment in its definition of an affordance. Stoffregen [170] sought to address this by re-framing affordances as *properties of the animal-environment system*:

> "Affordances are properties of the animal-environment system, that is, that they are emergent properties that do not inhere in either the environment or the animal." [170]

Stoffregen's formal definition for this was as follows:

Definition. Let W_{pq} (e.g. a person-climbing-stairs system) $= (X_p, Z_q)$ be composed of different things Z (e.g. person) and X (e.g. stairs). Let p be a property of X and q be a property of Z. The relation between p and q, p/q, defines a higher order property (i.e., a property of the animal-environment system), h. Then h is said to be an affordance of W_{pq} if and only if

(i) $W_{pq} = (X_p, Z_q)$ possesses h

(ii) Neither Z nor X possesses h.

Here, Stoffregen explicitly defines an affordance as a relation h between a property of the stairs and a property of the person. The idea is that the dynamics of animal locomotion are influenced by the dynamics of the environment and vice versa:

> "The result is that there are dynamics that are unique to 'this animal if it climbed these stairs' or 'this animal when climbing these stairs,' but that do not inhere in either the animal or the stair." [170]

2.1.1.3 As Relations between Organism and Environment

In Chemero's [26] critique of previous attempts at formalisation of affordances, including those of Turvey [182] and Stoffregen [170], he cited that where those authors agreed that affordances are animal-relative properties of the environment, their main disagreements were over two issues: what kind of animal-relative properties of the environment affordances are, and what it is about animals that affordances are relative to.

Chemero argued three main points:

1. Affordances are not properties (or at least not always).
2. Affordances are not in the environment.
3. Affordances are in fact relations.

To this end, he provided the following initial definition:

Definition.

$$\text{Affords-}\phi(\text{environment, organism}), \text{where } \phi \text{ is a behaviour.}$$

before arguing that this interpretation was incomplete due to confusion over which aspects of the environment related to which aspects of the organism. He points out that taking the obvious interpretation of the experimental work of Warren [194] to be expressions of affordances as ratios between body scale and some bit of the environment measured in the same units, for example:

$$\text{affords-climbing(my leg length, riser height)},$$

carries with it the flaw that body scale is just an easily quantifiable stand-in for ability. Thereafter he referred to subsequent experimental work by Cesari *et al.* [18] as evidence that

> "...people perceive stair climbing and descending affordances not as the ratio between leg length and riser height (as Warren 1984 [194] held), but rather as a relation between stepping ability and riser height." [26]

and thus later refined the definition to:

Definition.

$$\text{Affords-}\phi(\text{feature, ability}),$$

thereby explicitly defining affordances to be relations between abilities of organisms and features of the environment.

2.1.1.4 As a Three-Way Relationship

A more contemporary formalisation, borne out of autonomous robotics but grounded in the language of ecological psychology, is that of Şahin *et al.* [156]. Basing their definition on that of Chemero, which they argued was too generic to be useful in an autonomous robotics context, they extended it to explicitly account for the environmental effect generated by an agent with a given ability acting on a given feature of the environment [45, 76, 156]. Their formal definition was therefore as follows:

Definition. An affordance is an acquired relation between a certain effect and an (entity, behaviour) tuple such that when the agent applies the behaviour on the entity, the effect is generated. This may be formalised as

$$\text{(effect, (entity, behaviour))}.$$

They used the term *entity* to denote the environmental relata of the affordance rather than the term *features* as used by Chemero or *object* as used elsewhere since, in their view, it is a less restrictive term. They also argued that the agent's relata should represent the part of the agent that is generating the interaction with the environment that is producing the affordance and is best encapsulated by the term *behaviour*. They emphasize that all three components in the definition are assumed to be sensed through the proprioception of the agent, i.e. through perception-action routines for behaviour, through the perceptual representation of the entity and through the perceptual detection of a change in the environment of the effect. Taking all of this together, an example of an affordance defined under the interpretation of Şahin *et al.* might be

$$\text{(lifted, (black-can, lift-with-right-hand))}$$

where the term *black-can* is short-hand for the perceptual representation of the black can by the agent, like, for instance, a feature vector derived from cameras observing the can. The labels *lifted* and *lift-with-right-hand* would also be short hand for similar perceptual and proprioceptive representations.

2.2 Affordances in Studies on Humans and Animals

2.2.1 Studies on Humans

Extensive experiments have been reported in the ecological psychology literature and elsewhere aiming to analyse human perception of affordances. Most of these experiments involve analysing ratios between bodily measurements of humans and environmental measurements and finding correlations between such ratios and the ability of the human subjects to perceive certain affordances. Various such experiments have been trialled on both human adults [25, 193, 194] and human infants [64, 113, 129], as well as on different types of affordances, including environmental 'traversability' [25, 64, 81, 128, 194, 194], object 'graspability' [71, 113, 124, 129, 180, 181], and even 'sit-ability' [110].

It was in Warren's classic study [194] of stair climbing affordances that the paradigm of using ratios between both bodily measurements and environmental measurements as a basis for measuring affordances was introduced. By analysing the variations in such measurement ratios, specifically the ratio between leg length and stair riser height, Warren was able to demonstrate that there are critical points, corresponding to phase transitions in behaviour, and optimal points, corresponding to stable, preferred regions of minimum energy expenditure. Two groups of short and tall men respectively were studied in three experiments examining visual perception of critical riser height, energetics of optimal riser height, and visual perception of optimal riser height, respectively. The perceptual category boundary between "climbable" and "unclimbable" stairs was predicted by a biomechanical model, and visually preferred riser height was predicted from measurements of minimum energy expenditure during climbing. Through these experiments, Warren was able to show that perception for the control of action reflects the underlying dynamics of the animal-environment system, at least in the context of human stair-climbing. Subsequently, Warren and Whang [193] extended Warren's original analysis to the visual guidance of walking through apertures. Groups of large and small human subjects were videotaped walking through apertures of different widths to determine the critical ratio between the aperture width and shoulder width that would marks the transition from frontal walking to body rotation. By using an Ames room setup, they were also able to study the relationship between the perception of the "passability" affordance and perceived eye height of the human subjects by adjusting the floor height of the aperture setup. They found that raising the floor height resulted in a corresponding change in affordance perception such that narrower aperture widths were judged to be passable.

Further to the original experimental paradigm developed by Warren *et al.*, E.J. Gibson *et al.* [64] expanded beyond visual perception of affordances by including a haptic aspect to their experiments with infants traversing surfaces. The infants in their study were presented with two surfaces in succession: a standard surface that both looked and felt rigid and a deforming surface that both looked and felt nonrigid. By studying latency in locomotion of the infants when presented with various different configurations of the surfaces, they were able to conclude that the infants perceived the traversability of surfaces in relation to the mode of locomotion, i.e. either crawling or walking, characteristic of their developmental stage and that they did this by actively exploring the properties of the surfaces both visually and haptically. For example, when confronted with a nonrigid surface, walkers tended to either hesitate for longer than crawlers, choose a rigid, patterned route as an alternative when a choice was offered, engage in more exploratory activity or divert their attention away from the nonrigid surface more. Kinsella-Shaw *et al.* [81] also brought a multi-modal approach to bear on a study of the traversability of sloped surfaces, using two experiments: one designed to test the optical perception of a slope in adult humans and the other designed to test the geographical perception of a slope by measuring the foot inclination of participants using suitable apparatus. In other studies of traversability affordances involving more dynamic contexts, Oudejans *et al.* [128] looked at how people cross streets safely in the presence of traffic, while Chemero *et al.* [25] used a moving platform to examine how changes in the layout of affordances affect perception of them.

With regard to the study of "graspability" affordances, one early example is that of Newell *et al.* [124] where both preschool children and adults participated in experiments grasping cubes of various sizes. Their study indicated a relationship between task constraints and the bodily constraints of the participants when it came to selecting the grasp pattern to be utilized. However, when the objects were scaled to the hand size of the participants, they found common dimensionless ratios between body and object measurements across all age groups, suggesting that body scale plays a significant role in the development of coordination. Tucker and Ellis [180] utilized a stimulus-response compatibility (SRC) behavioural paradigm to study the response times of adult human subjects who, when presented with images of common graspable household items, were asked to judge whether they were upright or inverted by pressing buttons with either their left or right hands. The objects were oriented horizontally such that grasping them would be more naturally suited either to a left-handed grasp or a right-handed grasp depending on the particular view in a given image. Amongst other results, they demonstrated that the left-right orientation of the objects had a significant effect on the speed with which a particular hand made a simple push-button response, thus suggesting that seen objects automatically potentiate aspects of the actions they afford without there necessarily being an intention to act on the part of the subject. McCarty *et al.* [113] looked at the role of vision in grasping experiments involving infants reaching for horizontally and vertically oriented rods under different lighting conditions, They demonstrated that hand orientations of the infants remained consistent regardless of lighting conditions or whether or not they could observe the objects throughout the reach, indicating that the infants may use the initial sight of object orientation, or the memory of it, to plan a grasp.

2.2.2 Studies on Non-Human Animals

Experimental studies on human affordances go part of the way toward elucidating the biological basis of such phenomena, but humans are not the only organisms that make use of environmental affordances, and thus experimental studies on animals are likely to prove to be just as important. A variety of different species have been observed exploiting affordances, to the extent that some species have been known to go beyond what might be accounted for purely by genetic encoding as exemplified by their engagement in certain interesting forms of tool-use. Such species include primates [12, 85, 139, 191], cetaceans [96], birds [21, 80, 177, 178], cephalopods [52] and rodents [114, 115], and we endeavour to briefly review some of those examples in the following.

A book by Wolfgang Köhler entitled *"The Mentality of Apes"* [85], first published in 1925, was one of the earliest published studies detailing the use of tool affordances by animals, specifically chimpanzees. Köhler observed chimpanzees in captivity on the island of Tenerife and performed experiments to study their behaviour and problem-solving skills. The animals were tested in a number of ways. In one experiment, they were placed in a cage where food was positioned just out of reach outside the cage. The chimps used sticks as reaching tools and were able to obtain the food. In another experiment, illustrated in Figure 2.1, food was placed in an area that was too high for the chimps to reach and they solved the problem by stacking wooden boxes and climbing up to reach the food. As a result of such experiments, Köhler concluded that the chimps exhibited some level of insight over and above what might be attributed to simple trial-and-error experiments: they appeared to realise the solutions to the problems in advance and set about implementing them through deliberate planning. A more recent study [139] has shown that chimpanzees are capable of hunting with tools. Other recent studies have exposed various forms of tool use in other primates. Breuer *et al.* [12] have reported the first two known observations of tool use in wild gorillas where one female gorilla was sighted using a stick to test water depth prior

Figure **2.1:** Chimpanzees using wooden crates to retreive bananas in the experimental studies of Wolfgang Köhler [14].

to crossing a pool of water, while a second was spotted manipulating a shrub trunk to be used as both a stabilizer during food and as an improvised bridge for crossing a deep patch of swamp. Meanwhile, wild bearded capuchin monkeys have been observed carefully selecting appropriate hammer-like stones and using them to crack open nuts [191].

Experiments performed by Mechner *et al.* [114, 115] in the 1950's and 60's demonstrated that caged rats could learn to press various levers in a certain sequence in order to obtain a food reward. In Mechner's original work [115], the rats were deprived of food for a certain amount of time before being placed in their cages which contained two different levers. Pressing one of the levers would deliver the food reward, but only after the other lever had been pressed a certain number of times (usually a small number, e.g. four times). If the rat were to press the second lever the wrong number of times or not at all, or press the levers in the wrong sequence, it would be punished. According to the results, the rats were able to learn to press the levers in the correct sequence in order to obtain the reward. In their subsequent work [114], Mechner *et al.* demonstrated that the rats were learning to associate the truly relevant parameter, i.e. the number of lever presses, to the reward delivery, as opposed to the duration of the lever pressing sequence. They achieved this by showing that their results were invariant to the degree of deprivation of the rats. Thus, even though some of the rats were extremely hungry and pressed the levers much more quickly to get the reward, they still learned to press the first lever the correct number of times.

Studies on tool use in various species of bird have garnered much attention in recent times. Egyptian vultures have been observed throwing stones to break into ostrich eggs both in the wild and in captivity where they were reared from birth under circumstances where the possibility of cultural transmission for stone-throwing by copying experienced birds was ruled out [178]. New Caledonian Crows (Corvus moneduloides) have been shown to exhibit extraordinary tool manipulation abilities. Studies by Kenward *et al.* [79, 80] have shown how such crows not only make use of tools to retrieve food, but are capable of spontaneously manufacturing and utilizing their own tools from twigs and other raw materials without any contact with adults of their species or any prior demonstration by a human. Taylor *et al.* [177] showed that the abilities of such crows also extend to metatool use such that they may, for example, use a short tool from one location to extract a longer tool from another location which is in turn used to extract a piece of meat from a third location that would be otherwise unreachable using the shorter tool.

Tool use is not solely limited to land-based organisms. A variety of ocean-dwelling creatures have also been observed exploiting tool affordances on the seabed. In an interesting example of how a tool affordance can be discovered and subsequently communicated amongst a population for widespread benefit, bottlenose dolphins have been observed using pieces of sponge to protect their noses while foraging for fish, a skill that is believed to have been discovered by one dolphin before being passed along to her offspring [96]. Octopuses, long considered to be one of the more intelligent underwater species, have recently been observed carrying around coconut shells before climbing into them and employing them as a protective exoskeleton when necessary [52], similarly to the octupus depicted in Figure [17].

Figure **2.2:** A small octopus using a nut shell and a clam shell for protection [17].

2.3 Affordances in Artificial Intelligence

2.3.1 The Symbol Grounding Problem

The ideas discussed above naturally confer a particular importance on linking action and perception, which can be regarded as an example of the *symbol grounding problem (SGP)* as introduced by Harnad in 1990 [72]. The symbol grounding problem can perhaps best be characterised as addressing the issue of how symbols get their meaning and what that meaning really is, or what that meaning relates to. It seems clear that this problem is highly related to the concept of affordances if we consider that linking different modalities in an embodied cognitive system implies that internal representations are grounded in world experience and that these representations are in turn, linked together in some meaningful way. Not all would agree with the purist interpretation of symbol grounding [13] where the symbols employed are pre-specified or indivisible, and it seems that some sort of crossover between symbol grounding and connectionist or more generalised learning mechanisms could be the appropriate way forward. Since the introduction of the symbol grounding problem, [72], a plethora of papers have been published aiming to address it [2, 24, 35, 62, 152, 153, 169, 192] Harnad proposed a hybrid model [72] as a means of solving the SGP that would mix the useful elements of both symbolic and connectionist systems by connecting the symbols manipulated by an autonomous agent to the perceptual data they denote. This formed the basis of further analyses by Mayo [112] and Sun [173] in a similar vein. Subsequently, a number of authors re-analysed the problem [147, 192] and attempted to extend the hybrid model in various directions. In particular, Davidson's 1993 study of symbol grounding [35] emphasises the importance of incremental learning for concept formation and grounding of concepts, a point we shall return to when discussing our requirements for our affordance learner later in Chapter 3.

Symbol grounding problems have often been addressed by researchers from various fields, from psychology, to computational linguistics, artificial intelligence, and computer as well as cognitive vision. Some of our previous work [164, 165] on developing a cognitive vision framework focused on learning qualitative linguistic descriptions of visual object properties and scene descriptions. This was closely related to the work of Roy [150–155] where, for instance in [150], his system was designed to learn word forms and visual attributes from speech and video recordings, work that was subsequently extended for generating spoken descriptions of scenes [152]. The work of Chella *et al.* [2, 24] contained further attempts at developing cognitive learning frameworks involving symbol grounding. Their work was based on Gärdenfors' paradigm of three levels of inductive inference [62] and their implementation of this paradigm in [2] involved grounding linguistic symbols in superquadric representations of scenes using neural networks.

2.3.2 Constructivist Learning

Constructivist learning is a theory of learning that was championed by Swiss developmental psychologist Jean Piaget [136] during the mid-twentieth century, the main tenet of which being that, during development, children form new ideas or representations by constructing them from known or previously learned ones. Affordance learning is closely related to constructivist learning for two main reasons. Firstly, before a cognitive agent can learn any complex, constructivist concepts in its environment, it must be aware of what the environment and the contents of the environment affords it. In most existing constructivist learning systems, these basic affordances are atomic and assumed to be known in advance. Secondly, although basic object affordance ideas such as "I can afford to push the ball to the left" can be learned using traditional learning algorithms, it takes something more akin to hierarchical, constructivist learning to acquire an equally valid, but *recursive* affordance concept such as "I can afford to push the ball around the obstacle by pushing it right, then pushing it up, then pulling it right". Developing this idea is crucial to realizing the full potential of affordance learning. With our work, we aim chiefly to contribute to the former aspect, whereby our proposed work would aid in deriving basic affordance concepts which could then be used to form more high-level constructivist concepts.

There have been a number of attempts to develop constructivist learning systems [20, 23, 46, 122, 123, 135], notably Gary Drescher's work, as presented in his 1991 book "Made-Up Minds: A Constructivist Approach to Artificial Intelligence" [46], which discussed the design of a cognitive-developmental learning system called the *schema mechanism* in an artificial learning environment. The schema mechanism started with a set of primitive *items* and primitive *actions*. Items were binary variables that represented environmental states, e.g. `+GripperOpen` or `-GripperOpen`, while actions allowed the environment to be manipulated, e.g. `CloseGripper`. Associations were formed between items and actions through the use of *schemas*, expressions that took the form of `<Context/Action/Result>` where `Context` and `Result` represented sets of known item states. `Context` item states refer to the initial state of the environment before the `Action` is taken, whereas `Result` item states refer to the predicted state of the environment as a result of taking that action, given that particular context. An example schema might be `<+GripperOpen/CloseGripper/-GripperOpen>`. It is worth noting here the similarity between such schemas and the tuples in the affordance formalisms discussed previously. Without going into detail, during an exploratory learning process that started out with bare schemas for every action, the schema mechanism used a mechanism called *marginal attribution* to record statistics on item state changes when actions are performed, monitor them over time, and "spin off" new schemas when certain thresholds were breached. This, along with the help of some other machinery, implemented a form of constructivist learning.

Although Drescher's work was highly influential, it was also highly computationally inefficient. Moreover, the items and actions of the schema mechanism were discretised, making it ideal for deterministic AI applications, but less obviously suitable for scenarios involving real data, e.g. developmental robotics, computer vision. If, however, a system could be devised that took the overall constructivist flavour of the schema mechanism, applied it to real-world data and made it efficient enough to function on-line, it would become a powerful tool for learning in developmental cognitive systems. A number of authors have attempted to tackle the inefficiency issue [22, 23, 30, 141], notably Chaput [22, 23]. Chaput's system, dubbed the *Constructivist Learning Architecture Schema Mechanism (CLASM)*, stemmed the inefficiency issue by using self-organizing maps

(SOMs) [86] to create a hierarchical structure by using the output of low-level SOMs as input to higher-level SOMs.

In his Ph.D. thesis [23], Chaput described an experiment where the CLASM system was used in conjunction with reinforcement learning to enable a simulated Pioneer 2-DX mobile robot to learn how to navigate in a simulated environment and acquire specimen objects with an associated reward. A gripper on the front of the robot housed a laser trigger that indicated when an object is within the grippers. A camera system on the robot provided a 60° viewing angle that was split into five segments: `s1`, `s2`, `s3`, `s4`, `s5`. In the experiment, the robot started off with six possible primitive actions: `forward`, `backward`, `turn left`, `turn right`, `turnaround` and `grip`, as well as seven primitive items: `s1`, `s2`, `s3`, `s4`, `s5` that indicated when an object was present in the respective view segment, `ingrip` that indicated when an object was in between the grippers and `bump` that indicated when bump sensors on the robot were activated by the robot hitting a wall in the environment. By offering a reward for activating the `ingrip` item and diminished rewards for any schemas that bring the robot closer to activating the `ingrip` item, CLASM was able to develop a schema policy to target and acquire an object specimen. Examples of learned schemas included `<+s1/turn left/+s2>` which describes how turning left might move a specimen from one part of the visual field to another, and `<+s3/forward/+s3>` which indicates that moving forward while a specimen is in front of the robot will leave the specimen in front. This, in effect, is an implementation of a form of constructivist affordance learning. Particular actions afford particular results given particular contexts; these relationships are learned as schemas and these schemas may be combined to form more complex affordance schemas.

There is ample scope to develop this idea in a real robotic system, and although a fully fledged implementation will be beyond the scope of this work, we aim to contribute to an important aspect of it that is of fundamental importance in systems that have to operate in the real world. In the above simulated system by Chaput, although the robot operated in continuous sensor and actuator spaces, the spaces were discretised and the discretisation, e.g. `s1`, `s2`, `s3`, `s4`, `s5`, is specified in advance. Mugan *et al.* [122, 123] provided an example of a system with a robot arm operating in simulation that automatically discretised its feature spaces and learned the affordances of its environment and subsequent planning strategies through exploration and observation of contingencies. We aim to create a *real-world* system that can automatically discretise its feature space(s) in an unsupervised manner through interactions in its environment and that can thereby learn basic object affordances by observing similar contingencies.

2.4 Affordances in Vision

In the history of the study of vision, which itself arguably dates back as far as Aristotle, there have been many paradigm shifts in thinking over the years and sometimes significant ideological differences. One such troublesome divide of particular relevance to the affordance learning problem, which indeed this thesis attempts to address in part, stems from the dichotomy between the perspectives of Gibson, who saw vision as a "bottom-up" direct process, in which visual percepts already exist out in the world and are in some sense parsed from it, and that of neuroscientist David Marr [111] and the earlier Gestalt psychologists such as Max Wertheimer [201], who saw vision as a "top-down" indirect process, in which hypotheses of visual percepts are inferred from sensory information. Outside of that broader ongoing debate, while much of the recent work done in the area of computer vision has tended to focus on specific sub-topics in sub-domains such as recognition, categorisation, tracking and so on, even if these efforts have not always been directly targeted at solving more general or multi-disciplinary problems like affordance learning, they can certainly be said to have helped to establish a basis for doing so.

2.4.1 Function-Based Object Recognition

A closely related area of computer vision to affordance learning is that of function-based object recognition [202, 204]. Works in this area tend to be similar to affordance-learning, in that they attempt to recognize and learn about the functionality afforded by various objects, but they tend to assume that the object models are known and tend not involve actual physical interaction with the objects. For example, in [202], Woods *et al.* present a system that uses labelled 3-D training models of chairs and cups to learn fuzzy membership functions. These fuzzy membership functions are used to provide evaluation measures which determine how well a shape model fits the functional description of an object category. This functional description is a category definition tree built from functional properties such as “Provides sittable surface” and “Provides arm support”, which in turn are built from measurable primitives such as “relative orientation” (angle between normals for two surfaces) and “stability” (checks that a given shape is stable when placed on a flat supporting plane). Of course, one of the main differences between this and the above works on affordance learning in robotics, is that the 3D object models are provided a priori. As the authors note, it is conceivable that such a system could be used in conjunction with range imaging to identify functionally important parts of real objects, but this was not implemented in their work. This was subsequently addressed in [204] where Wünstel *et al.* used $2\frac{1}{2}$D laser range data to recognize basic objects like chairs or tables within an office environment. Their approach split the range data into three separate layers by training a Bayes classifier on sample data. Morphological operations were used on each of the layers to extract object segments and from there, object classification could take place using a similar classification tree to [202] and relations such as “isLower”, “isHigher”, “minDistCenter” and “maxDistCenter” between the object segments in the various layers. More recently, Grabner *et al.* [67] proposed a method for learning an affordance detector that identifies locations in a 3-D space that support particular functions. They assumed a 3-D model of a scene as well as a 3-D model of an actor, then tried to match the actor model in appropriate poses to parts of the scene in order to find areas where the “sittability” affordance was realised.

2.4.2 Object Categorisation

Much has been accomplished in recent times in the area of object categorisation, a sub-topic that aids in the affordance learning process a great deal since objects are often categorised on the basis of common features, shape, or compositions of parts, all of which are useful ingredients when approaching object affordance learning. A number of authors have used various combinations of low-level features and compositions of parts in combination with probabilistic generative modelling for object categorisation [49–51, 99]. Such generative approaches can incorporate prior knowledge into their representations as a probability density function on the parameters of their models, using information from previously learnt object categories, before generating posterior models for subsequent categories using just a small number of training samples and Bayes’ theorem [49, 50]. A parallel can be drawn between such approaches, where competing model hypotheses are matched to low-level sensory information, and Marr's [111] top-down approach to vision. Others have taken the approach of discriminative modelling, that is, rather than modelling joint probability distributions over training samples and categories as in the generative case, modelling the mapping from training samples to categories directly, either estimating the posterior class probabilities or a decision function [83, 125, 190]. While these types of approaches provide much in the way of visual feature representation and machine learning techniques, that by themselves can indeed be used to address the affordance learning problem in some sense, e.g. learning from affordance-labelled images, they are found lacking in an aspect most crucial to the problem — action. Here, we turn to the robotics literature, where action is, naturally, a more central concern.

2.5 Affordances in Robotics

2.5.1 Pushing and Pulling Objects

Perhaps the most closely related work in the literature on affordance learning to our proposed work is by Fitzpatrick *et al.* [55–57, 116]. The authors trained the Cog robot to recognize "rolling" affordances of four household objects (a plastic bottle, a toy car, a toy cube and a toy ball) using a fixed set of four actions to poke the objects in different directions as well as simple visual descriptors for object recognition. During training, the robot would localise objects viewed by its camera system using colour histogram back-projection [175], gather shape features, poke the object from a certain direction using one of four pre-defined actions, and track the object for a dozen frames afterwards. Over many trials, histograms were created for each of the four actions and each of the four objects. For the action histograms, statistics were gathered over all the object types to measure the likelihood of each action causing the objects to move in a particular direction. From this, the robot learned that objects, generally, are likely to move to the right when pushed using an action that pokes from left to right. Similar lessons were learned for the other three poking actions. For the object histograms, statistics were gathered during trials of the likelihood of each of the different objects moving in a particular direction with respect to their principle axis, which was found using the shape features gathered after localisation. From this, the robot learned that the bottle is likely to roll at right angles to its principle axis, the car is likely to roll along its principle axis, and both the cube and the ball have no particular direction of movement. While the learning scenario described is quite similar to our proposal, there are a number of key differences in how we intend to have our system learn object affordances. Firstly, the feature associated with the rolling direction affordance is pre-determined, both in case of the action histograms, where it is the angle of movement in degrees, and in the case of the object histograms, where it is the difference between the angle of movement and the principle axis of the object. What we intend to do differently, is provide our system with a number of different result features and have it determine for itself which ones are important for identifying different affordances based on variance between trials with different objects. Secondly, in the case of the action histograms the system of Fitzpatrick *et al.* learned rolling direction affordances for *all* types of objects presented, whereas in the case of the object histograms, it learned a rolling direction affordance for *each individual* object (in the testing phase, the particular object was recognised using histogram back-projection, and its rolling direction was determined based on its individual identity). We intend to have our system determine the affordance class of objects (grounded in result features) based, not on their individual identity, but on their *class* with respect to a broad set of object property (e.g. shape) features.

Stoytchev examined an interesting extension of the general affordances idea in [171, 172] by enabling a robotic arm to learn tool affordances, i.e. action possibilities that different tools afford the system when used to interact with an object. He distinguished between two sub-classes of tool affordances: *binding affordances*, that describe how tools afford being grasped by the arm, and *output affordances* that describe the effect that the various tools afford during interaction with the environment. Like Fitzpatrick *et al.* in [171], he used a small set of predefined exploratory actions and, by allowing the arm to explore these actions with different T-hook tools acting on a hockey puck, enabled the robot to build a tool output affordance table. The T-hook tools were colour-coded to facilitate recognition and affordances were learned with respect to individual tools rather than classes of tools. It is also worth noting that the features used to learn these tool output affordances were pre-selected and explicitly associated with these particular affordances based on prior knowledge of the experimenters. In [172], Stoytchev examined tool binding affordances, though in a simulated robotic system.

An architecture for action (mimicking) and program (gesture) level visual imitation in a robotic platform is presented in [103], where object affordance contexts are used to focus the attention of a gesture recognition system and reduce ambiguities. Gesture recognition in the motor space is performed using a Bayesian framework that relies on prior knowledge from object affordance contexts, which are provided. In [121], the authors used a humanoid robot to push objects on a table and used a Bayesian network to form associations between actions, objects and effects. Though quite similar to our approach, their learning method may not be as amenable to full online learning, as they have to gather a certain amount of data initially to form categories within the various modalities before the network can be trained. In the work of Omrčen *et al.* [127], the robot first observes how an object moves when pushed in a certain direction. The collected data are used as input to a neural network which learns to predict the motion of pushed objects. Kopicki *et al.* [88–91] used a probabilistic framework to address prediction of rigid body transformations in an object pushing scenario both in simulation and using a real robotic system. Their work was similar to ours here in that it explicitly addressed the representation of object parts as well as the combination of knowledge from multiple models. However, their visual system relied on the use of prior object models for object localisation, something we explicitly avoid in this work.

There have been some recent examples of works in robotics where pushing objects has been used to achieve some goal other than direct affordance learning, while still being strongly linked to affordance learning. Katz *et al.* [77, 78] used pushes to enable a robot to autonomously acquire manipulation expertise with articulated objects, e.g. scissors and shears, by modelling their kinematic structure with a relational state representation and applying reinforcement learning. Ude *et al.* used pushing as a means of developing early concepts of objects that a robot sees for the first time [184]. By probing hypothetical objects in a scene with pushes from a humanoid robot arm, visual features on the objects were caused to move, something that enabled the robot to either confirm or reject the initial object hypothesis. Thus, the robot would consider as objects those physical entities that are comprised of features which move consistently when acted upon. This also enabled figure-ground segmentation of the object hypotheses, something that is, in turn, very useful for later potential affordance learning.

2.5.2 Grasping Objects

One approach to the grasping problem is to compute grasps directly from visual observations. This may be done by, for example, identifying graspable parts of objects, then fitting a grasp to one of those parts. Miller *et al.* [119] accomplished this by modelling objects as sets of shape primitives such as spheres, cylinders, cones and boxes, and using rules to generate a set of grasp starting positions and shapes. However, there approach required that the objects be decomposed to primitives by hand. This was improved upon in a subsequent paper [75] where the object parts decomposition was achieved by fitting the shape primitives to object point clouds. Each of the above methods have shortcomings in that they require pre-definition of many primitive shapes and grasps. More recently, Popović *et al.* [137], rather than making use of any specific prior knowledge of objects or object models, used an early cognitive vision system to extract co-planar, co-colour pairs of edges of objects for grasping.

Rather than engineering a mapping from visual information to grasp parameters as in the above approaches, another approach is to learn to compute grasps from visual observations, that is to learn such a mapping from the data. Saxena *et al.* [159–162] provided an interesting blend of ideas from function-based object recognition and affordance learning when constructing a robotic system for grasping novel objects. Their robot uses the same robotic arm used in our work (cf. Chapter 6) that has a webcam mounted on the end-effector. In [159] they trained a probabilistic model on features extracted from ray-traced images of synthetically generated household objects with labelled grasp points. After training, the system was tested on attempts to grasp real objects that it had not encountered in the training set by finding a potential grasping point and choosing between one of two possible types of grasping actions. Although synthetic object models are used for training, their work differs from function-based object recognition in that the synthetic models are purely used to generate images for view-based learning. From an affordance learning perspective, the fact that this system recognizes parts of novel objects that are graspable is impressive, however, similarly to some of the other approaches mentioned thus far, the actual affordance concept that is learned is binary and specified in advance: graspable or non-graspable.

Later, Detry *et al.* [44] tackled a similar problem and, rather than training their learning algorithm on synthetically generated objects, they enabled an autonomous robotic arm to grasp objects in an exploratory manner, trained on these interactions with real objects. They described a method for learning object grasp affordance densities, i.e., continuous probabilistic models of object grasp success over object-relative grasp poses. In both of these works the two possible affordances were specified in advance: graspable or non-graspable. Ugur *et al.* [187], on the other hand, developed a system generated its own affordance classes through interaction with objects. They worked with a robotic system consisting of a range scanner and a robotic arm that learned affordances of objects in a table-top setting using an unsupervised two-step approach of effect class discovery and discriminative learning for class prediction. They also applied similar techniques to a scenario involving self-discovery of motor primitives and learning grasp affordances [188]. The system presented in this dissertation, by comparison, also attempts to learn via class discovery and prediction, but in a self-supervised multi-view online setting. More recently, Ugur *et al.* [189] have expanded upon this work to develop a system whereby complex affordance learning is bootstrapped via pre-learned basic affordances, an idea that is commensurate with the constructivist learning approach discussed previously.

Montesano *et al.* [120], expanding upon their prior work in [121], proposed an algorithm for learning grasp affordances by mapping local features to the probability of success for specific grasping actions. Their method was able to select the most relevant features amongst a set of general low level visual descriptors for successful grasping by automatically adjusting kernel bandwidths in a probabilistic model based on robot experience. Sweeney and Grupen [176] recalled the idea of decomposing objects into parts, but with a visual learning approach.

2.5.3 Other Forms of Affordances in Robotics

MacDorman [107] used the concept of affordances to develop a mobile robotic system named Ψro that must survive in a maze-like environment by intercepting ‘tasty’ robots and avoiding ‘poisonous’ robots. The architecture used to capture the affordance concept in this work was essentially three-tiered. The body tier provided sensor feedback and actuator possibilities, the motivation system tier tracked the internal state and well-being of the robot (e.g. battery level), and the internal model tier developed sensorimotor mappings between the body tier and the motivational tier based on the robot's experiences and its learning mechanisms. Partition nets [106], an on-line learning algorithm developed by the author, were used to learn the sensorimotor mappings. Affordance categories were formed as Ψro tracked invariant features over image signatures, however, these categories were not related to the actions that exploit the affordance, but to the internal indicators present in the motivational system.

Cos-Aguilera *et al.* [33] employed a similar idea to MacDorman's motivational system in their 2004 paper "Using a SOFM to Learn Object Affordances". They used a *self-organizing map (SOM)* [86] with a Hebbian learning mechanism called a *Growing When Required (GWR)* network to aid a simulated Khepera robot in learning affordances of objects with survival values such as nutrition and stamina so that it could prosper over time in its environment. The GWR network was used to cluster sensor data in the input space. Whenever the robot encountered a particular situation or object, the closest matching node in the network was found and weights were assigned to the node for each action the robot attempted to use in that situation based on the success of the action. Their prior work in [32] used a more simplified model to estimate the likelihood that a particular object, perceived as a set of features, affords the robot a particular action from its behavioural repertoire. While we also propose to make use of SOMs in our learning algorithm, we will be using them in a different way to Cos-Aguilera *et al.* Rather than using individual SOM nodes as exemplars, we intend to use SOMs to vector quantize feature space and cluster over the SOM nodes to form affordance classes.

Paletta *et al.* [60, 130–132] developed a mobile robotic platform called Kurt2 that is equipped with a crane featuring a magnetic end-effector which it uses to pick up metallic objects in its surroundings. Their affordance learning system uses reinforcement learning of predictive features (SIFT descriptors) to distinguish between two affordance classes of liftable and non-liftable metallic objects.

In some of their earlier work [185, 186], Ugur *et al.* developed a mobile robotic system equipped with a 3-D laser scanner that learned to perceive the so-called "traversability affordance" properties of various objects, such as spheres, cylinders and boxes in a room. Objects that the robot can easily move, e.g., spheres and cylinders that are lying on their curved edge, are considered traversable, whereas boxes and upright cylinders are considered non-traversable. In [186], the robot was provided with a set of seven possible actions (one to move forward and six possible rotations) and used its range scanner to gather over 30,000 angle and distance features aggregated over a grid-division of the range image. It then learned mappings between environmental situations and the results of its actions by first selecting relevant features from the full set, then using support vector machines to classify the relevant features into affordance categories. Though good results were achieved, the affordance categories were pre-defined and binary: traversable or non-traversable.

2.6 Chapter Summary

In this chapter we reviewed the literature and related works that have informed our research on object affordance learning. This review traversed a number of different fields including ecological psychology, experimental psychology, artificial intelligence, computer vision and autonomous robotics. Ecological psychology, being the area where the term 'affordance' originated from, provides a core insight into what affordances actually are, how they have been approached historically, and how they might be formally defined. Experimental studies on both humans and animals are also quite valuable, as they provide clues as to how a robot might approach the affordance learning problem. Affordances appear in the field of artificial intelligence in a number of different contexts, but two problems therein that were of particular interest with respect to our research were the symbol grounding problem and constructivist learning, and we discussed their relationship to the affordance learning problem in this chapter. Computer vision is an important area when addressing the affordance learning problem in robotics, thus we reviewed some of the work done there in this chapter, particularly with regard to functional object recognition. We discuss computer vision methods as applied to affordance learning further in Chapter 6. Finally, we provided a broad-ranging review of how the affordance learning problem has been tackled in the autonomous robotics domain, looking at various different approaches to the problem, including pushing and pulling objects, grasping, and others. In the following chapter, we discuss our own approach to the affordance learning problem and how we set about enabling our robotic system to learn basic object affordances in a self-supervised manner.

Chapter 3

Framing the Object Affordance Learning Problem

In order to approach the problem of learning object affordances in a robotic system we need a number of things. First of all, we, of course, need a robotic system or platform of some kind, such that the objects may be perceived and manipulated, and such that data may be acquired. We will also need to settle on a formalism of how the affordances should be represented. Given the diversity of affordance learning approaches in autonomous robotics and elsewhere, as highlighted in the literature review of the previous chapter, it would also be prudent to outline precisely what the requirements are for our affordance learner. The formalism and requirements will help to guide us in designing an appropriate affordance model and, in turn, an appropriate machine learning approach. The goal of this chapter is to address these issues at a high level and thus lay the foundations for the remainder of the thesis where concrete solutions will be proposed.

3.1 Formalising the Problem

As a starting point to modelling the scenario detailed above, we first present a modified form of the affordance formalisation of Şahin *et al.* in Section 2.1.1.4.

Definition. An object affordance is an acquired relation between a certain object-related effect and an (object, action) tuple such that when the agent applies the action on the object, the effect is generated. This may be formalised as

$$(\text{effect}, (\text{object}, \text{action})).$$

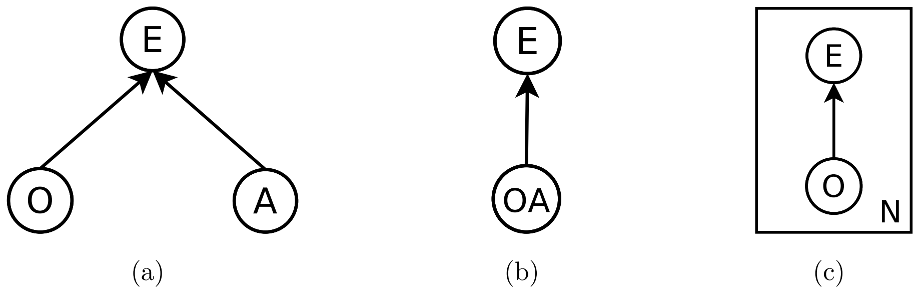


Figure **3.1:** Three potential models for representing affordances as defined by the formalism in Section 3.1. In (a) effects are conditional on objects and actions, which are independent. In (b) effects are conditional on objects and actions, which are concatenated together. In (c) effects are conditional on objects alone and N such models are considered for N-many discretised action types.

While the above definition is quite abstract, as is the case with many of the formal affordance definitions presented in the literature review of the previous chapter, it serves as a good basis for developing a more complex model. For simplicity in describing our particular affordance learning scenario, we have replaced the terms *entity* and *behaviour* in the original definition of Şahin *et al.* with *object* and *action* respectively. This ties in closely with the idea of *object-action complexes*, or *OACs* [95, 203], which have been proposed in the literature as basic building blocks for cognitive system architectures with a similar formalisation, where an action applied to an object forms a state transition function.

Without necessarily committing to a particular type of machine learning solution, it is convenient to interpret the above definition in the cause-and-effect diagrammatic language of directed graphical models [133] as in Figure 3.1. Figure 3.1(a) casts the affordance relation as an effect that is caused by (or conditional on) both an object and an action. Such an interpretation would involve parametrising both objects and actions and modelling them independently. A simpler alternative might be to concatenate objects and actions and set the effects to be conditional on such a joint model as in Figure 3.1(b). A further potentially even simpler alternative in the case where it is convenient to discretize the action space into a finite number of N well-defined actions is that shown in Figure 3.1(c), where N-many models are created where effects are conditional upon objects for a given action. This is the type of model we shall adopt for much of the remainder of this thesis. We will assume that the actions of the robotic arm are well-defined and consistently repeatable, allowing us to concentrate specifically on modelling the relationships between objects and the effects they undergo when those actions are performed.

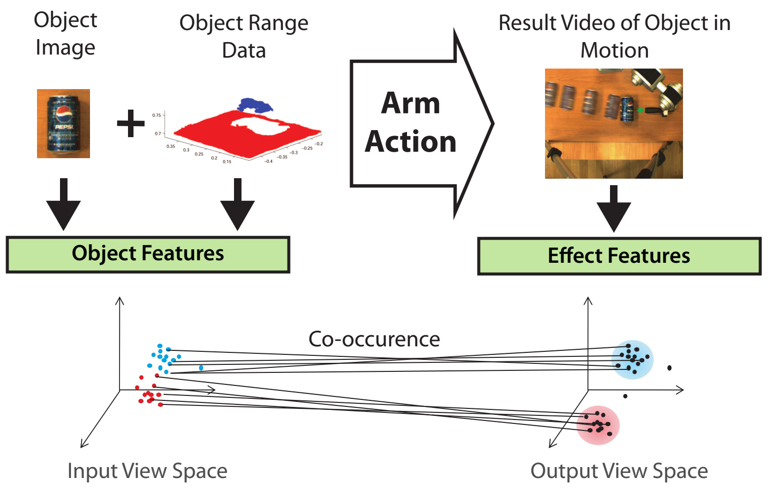


Figure **3.2:** Expanding on the main object affordance learning idea.

Two distinct sets of features are therefore being considered here, object features and effect features, and as data are gathered over the course of many interactions, they naturally form two feature spaces which we variously term the *object feature space* or *input view space* and *effect feature space* or *output view space* respectively. We shall return to the idea of separate data views, or sensory modalities, a little later in Chapter 4. For now, it is worth noting that given that data co-occurs across these different feature spaces, as illustrated in Figure 3.2, it should be possible to find interesting percepts within one feature space and map them to corresponding percepts in the other feature space using the co-occurrence information. For example, significant clusters of data might be identified in the object effect space, forming affordance classes, which might then be mapped back to data in the object property space, thereby making it possible to predict affordance classes in the effect space using object property feature vectors.

3.2 Framing the Problem within Machine Learning

So what kind of learning problem is this? A regression problem or a classification problem? Does it call for supervised learning or unsupervised learning? There are no empirical categories or classes in the world and nature does not label its data, at least not without giving rise to intelligent agents acting within the world. Yet, the ability to classify data is clearly an important facet of cognition [72, 98]. So if the data is not accompanied by class labels, it begs the question, how are classes formed in the first place? In this section, we elaborate on some of the ideas posed such questions and how they might be addressed.

3.2.1 Supervised Learning or Unsupervised Learning?

What is affordance learning really about from a machine learning perspective? Is it a supervised learning problem? Not quite it would seem. Certainly when children are at the early stages of their development, when they are motor babbling to ascertain basic affordances in their environments by flailing their arms at objects arbitrarily and subsequently grabbing objects and placing them in their mouths, they are unlikely to be making use of supervisory signals, at least not in the traditional sense. Lacking the language skills to communicate with their care-givers in those early stages, they are not yet interpreting complex linguistic notions such as "push the ball" or "pick up the doll". Nevertheless, there do appear to be qualitative differences inherent in many common affordances in and of themselves. Objects are either graspable or non-graspable, light switches can be either on or off, balloons can be bounced or burst or inflated, sand can be shovelled into a bucket or spilt out of it, etc. Is it perhaps then better formulated as an unsupervised learning problem? Clustering or density estimation of some kind might be used to find such qualitative structure in the affordance data. For example, visual data could be gathered on what bouncing a ball looks like versus what throwing a ball looks like, interesting features could be extracted from the visual data, and then the data could be clustered within this feature space to find a "bouncing" cluster and a "throwing" cluster. Yet this is not the whole story either. Such clusters would clearly be formed within the effect domain of the $(\mathrm{effect}, (\mathrm{object}, \mathrm{action}))$ tuple from our affordance formalism. Yet, when these clusters, and indeed the qualitative categorical labels they come to represent, do not exist at the outset of learning, what is there to be predicted from performing an action on an object? Regression could be performed, e.g. from the object space to the effect space, but when or where do the classes come into play? It would seem that there is an inherently developmental nature to the learning problem in this context. Perhaps in the early stages of learning, the classes do not exist but are gradually formed over time. This implies a form of *continuous* developmental learning, and we explore that idea further next.

3.2.2 Continuous/Online Learning

Continuous or online learning implies a life-long adaptive capacity characteristic of the type of learning exhibited in humans and many animals. Most machine learning algorithms tend to involve a one-off training phase where learning is performed by cross-comparing many training samples. This is referred to as batch learning since such algorithms are trained using a single batch of training samples. While such algorithms can be quite robust at recognising or categorising data samples, they fail to accommodate the novel information contained in data samples that were not encountered during the training phase, which could potentially be advantageous when recognising or classifying future samples. To this end, researchers have strived to design algorithms that continuously accommodate the salient aspects of novel data into their representations incrementally as new data is presented to the algorithms over time. This is especially important in the case of autonomous robotic systems, as they are often involved in learning scenarios where they may not have access to data that they previously encountered at any given moment, so it is important that they can accommodate the important aspects of novel data into their representations as and when they have access to the data.

A fair amount of work exists on the subject of online learning in developmental systems, e.g., [3, 82, 83, 168], particularly with regard to object recognition, e.g. [82] where the authors have developed a biologically motivated feature hierarchy-based method that combines notions of short-term and long-term memory to achieve online learning of objects. The authors of [168] developed an online learning system for the AIBO robot that uses a similar dialogue setting for tutor interaction to one that was developed during the course of our own research [163–165].

Some work of particular note comes from Weng *et al.* [74, 196–200, 206, 207] in the area of developmental robotics. Arguing that robots should develop their minds automatically and learn to deal with novel situations and environments without having to be re-programmed, Weng's team developed the SAIL and Dav developmental robots, as well as some interesting developmental and incremental learning algorithms to try to achieve these goals. In [198] Weng *et al.* offered the comparison between automatic development and other approaches shown in Table 3.1.

Approach	Species architecture	World knowledge	System behaviour	Task specific
Knowledge-based	programming	manual modelling	manual modelling	Yes
Behaviour-based	programming	avoid modelling	manual modelling	Yes
Learning-based	programming	treatment varies	special-purpose learning	Yes
Evolutionary	genetic search	treatment varies	genetic search	Yes
Developmental	programming	avoid modelling	general-purpose learning	No

Table **3.1:** Weng's comparison of various learning approaches.

Their suggestion that developmental learning should avoid pre-modelled world knowledge and system behaviour appears to make sense and is commensurate with the developmental principles we sought to address when initially broaching the research for this thesis, but precisely how to go about achieving that is another question. In addressing this issue, the authors define AA-learning (automated, animal-like learning) as a type of state machine where a robot's brain state is updated at every time step based on the previous brain state and sensor readings. A new action is also generated at each time step based on the current brain state and an action generation function. A mapping $h : \mathcal{X} \mapsto \mathcal{Y}$ must be formed from input space $\mathcal{X}$ to output space $\mathcal{Y}$ using training samples $\{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, 2, \ldots, n\}$ that arrive one at a time, where y_i may or may not be provided by tutor intervention (a tutor may choose to teach the robot to do something by directly driving its motors). The robotic systems of Weng *et al.* require real-valued outputs in the $\mathcal{Y}$ space rather than class labels, as these outputs are distributed directly to the motor systems of the robots. This would amount to a regression problem, were it not for the fact that Weng *et al.* also require superior generalization in the $\mathcal{Y}$ space. To this end, they developed an incremental learning mechanism called *Incremental Hierarchical Discriminant Regression (IHDR)* [196] that is capable of forming clusters in both the $\mathcal{X}$ and $\mathcal{Y}$ spaces.

We place particular emphasis on mentioning this here since a similar issue exists in affordance learning as we have framed it here. In our scenario, we have an object property space as well as an object effect feature space. Clusters must be formed in these spaces for generalization purposes and relationship mappings must be formed between the clusters, while regression is also important so that our agent may make feature predictions with real feature values as output. This idea of clustering in separate feature spaces will be developed further in the following chapter when we describe our proposed self-supervised multi-view learning algorithm. For now however, we turn to what our requirements for this self-supervised learner were as we were developing it, as informed by some of the ideas in the above discussion.

3.3 Requirements for Our Affordance Learner

In our proposed research, given the setup previously described, we sought to create an affordance learning methodology that matched the ideals of developmental robotic systems as closely as possible. To that end, our main requirements were the hard online learning requirements detailed in Table 3.2.

1.	The learning algorithm should be capable of *continuous or online learning.*
2.	We should avoid the explicit design of task-specific object models in favour of providing the system with a sufficiently rich sensory feature set from which it might *derive its own models.*
3.	We should ensure that the learning method employed to derive relationships between object types and the effects they undergo through robot interaction is *general-purpose* and *unsupervised* or *self-supervised.*
4.	The algorithm should be able to commence *learning from scratch* without access to an initial batch of training data, previously observed or otherwise.

Table **3.2:** Hard online learning requirements.

The motivation for the first requirement, which was summarized previously in Section 1.1.2 of the introduction, is primarily that developmental robotic systems should be capable of continuous learning because the learning scenarios they are involved in often preclude the possibility for batch learning by their very nature: such systems may not have access to previously encountered data at any given moment. The second requirement is important because the system designer may not know in advance what types of objects the system will encounter, how they will behave when the robot interacts with them, or how best to model these things so that the robot can learn effectively. Far better to provide the robot with sensory systems and mechanisms that derive features from those sensory systems from which it may derive its own models. This brings us to the third requirement. As the robotic system gains experience by encountering different objects, acting upon them, and observing the effects, its learning mechanism should do the three things listed in Table 3.3.

1.	Automatically derive affordance classes grounded in effect features, e.g. motion features observed both during and after object interaction, based on qualitative differences between data clusters.
2.	Automatically identify the object features, e.g. shape features observed prior to interaction, that are most relevant for affordance class prediction.
3.	Automatically model cross-view relationships between affordance classes and the predictive features.

Table **3.3:** Self-supervised learning requirements.

A learning algorithm that is designed in this way would be much more robust to changes in the underlying sensor and actuator systems, or changes in the learning scenario. For example, the robotic system might go from one learning scenario where it is using poking or pushing actions to move objects, to another scenario where it is attempting to pick the objects up using grasping actions.

The end goal of such a learning procedure will be to produce an object affordance classifier, so that when the robotic system encounters a novel object, it may observe its object property features, input them into the classifier, and derive its affordance class grounded in the result feature set.

The actions that are used to interact with the objects will be pre-defined and be treated separately from each other within the learning framework. That is, independent classifiers will be formed for each individual action. This is a sensible approach because since the robotic system controls its own actions, it will always have prior knowledge of which action is in use during a given interaction and will thus be able to invoke or update the appropriate classifier.

Ideally, a developmental robotic system should also be capable of constructivist learning. By this we mean that the system should not only progressively refine its recognition or classification abilities as it encounters new data, but also that it should have the ability to construct novel concepts by building on previously learned and/or innately specified ones. While we will not be implementing a full-fledged constructivist learning robotic system in this work, we do aim to provide enabling technology for an important aspect of such a system.

3.4 Chapter Summary

In this chapter we explored some the ideas that lead to the research that is developed in the remainder of this thesis. We formalised the affordance learning problem from the perspective of our approach to it, given our robotic setup, our particular object affordance learning problem, and the learning issues we sought to address. We elaborated on those learning issues from a machine learning and developmental robotics perspective with the goal of addressing continuous learning, at least in part. From there we outlined a set of requirements for our affordance learning algorithm that were broken into a set of hard online learning requirements and self-supervised learning requirements. In the following chapter, we describe the self-supervised multi-view learning framework we developed to address those requirements.

Chapter 4

Self-Supervised Multi-View Learning

This chapter is devoted to the development of a self-supervised multi-view learner that we will use to enable autonomous online learning of basic object affordance concepts. The learner will be multi-view in the sense that it will form representations of data views, or sensory modalities, by clustering data in each of their respective feature spaces and will connect them together via data co-occurrence mappings. It will be self-supervised in the sense that the data distribution in one data view coupled with the co-occurrence information, will be used to form an error signal for supervised learning in another data view. This will work on the basis that, as online training progresses, assuming an appropriate underlying data distribution, natural clusters should emerge in the driving view that can be treated as classes for discriminative learning.

Multi-view learning [174], sometimes also referred to using the somewhat less general terms cross-modal learning, multi-modal learning or co-clustering[1] [5, 7, 27, 39, 40] is an area of machine learning where, rather than having learning algorithms operate on data in a single feature space, learning is performed over multiple separate feature spaces, otherwise known as data views or modalities, in which data co-occur. Given that common theme, the learning goal may otherwise differ depending on the particular context [174]. A popular current application involves using web page text as one view and links to the web page as another

[1]The terms "cross-modal" and "multi-modal" are perhaps the most well-known of the ones listed here, but since the "modal" in their construction refers to "sensory modality", we prefer the terms "cross-view" or "multi-view" and the corresponding "data view" here, which can be applied somewhat more generally. A sensory modality could be considered to be a specific type of data view, whereas the reverse ought not be true. That said, we tend to use the terms interchangeably throughout the thesis and any ambiguities should be resolved via context.

view [5, 43, 102, 183]. Another common application in the literature is to attempt to relate co-occurring visual and audio data of utterances from human speakers [38, 126, 149]. Our particular interest stems from the domain of autonomous robotics and the problem of object affordance learning in particular [144].

In our affordance learning scenario, object properties such as shape can be considered as the basis for one data view, whereas object effects under interaction can be considered as the basis for another. The effects view ought to drive learning in the shape view, e.g. through the formation of effect classes which are mapped back to the shape view such that discriminative learning might be performed and object shape information used to predict object effects. This leads us to the idea of self-supervised multi-view learning which is a difficult problem to address, particularly if it is being considered as an online learning problem, which is often the case in the autonomous robotics domain and is how we will be considering it here.

Much of what is accomplished in this chapter could be regarded as being inspired by, as well as being a combination of, different ideas proposed primarily by three main authors. Miikkulainen [118] provided a model that cross-modally linked codebooks of labelled prototype vectors via Hebbian co-occurrence associative mappings in order to simulate lexical development. Each codebook represented different feature maps for three separate lexical modalities, orthographic, phonological and semantic respectively, and the codebooks were trained in an unsupervised manner using the *self-organizing map (SOM)* [86] algorithm while the Hebbian mappings interconnecting the codebooks were trained via Hebbian learning. By labelling the prototype vectors in each modality, it was possible to, for example, map associations between phonological concepts and semantic concepts. The structure of the learning network developed in this chapter is more-or-less identical to that of Miikulainen, but the manner in which it is trained is different. We forgo explicit supervised labelling in favour of deriving class labels in an unsupervised manner.

de Sa [37, 38, 40–42] developed an unsupervised neural network algorithm that learned separate visual and audio speech classifiers using co-occurring patterns of lip motion and sound signals from a human speaker. The basis for training this network was to minimize disagreement between modalities using unsupervised prototype labelling and learning vector quantization to optimize the positions of prototypes around decision borders. We also employ a modified form of learning vector quantization in this chapter, though the topological structure of our network and the nature of the training differ from that of de Sa *et al.*

More recently, Coen [27–29] proposed an algorithm for cross-modal clustering with a network structured similarly to that of Miikkulainen, though without the SOM training mechanism. Coen instead focused on finding meta-clusters of the prototype vectors in each modality by using the co-occurrence mappings to project probabilistic weights from one modality's prototypes onto another's in order to form a cross-modal distance metric to be used for the meta-clustering. In contrast with Coen, we are less interested in unsupervised co-clustering between modalities and more interested in finding class clusters in an unsupervised way in one driving modality that would be mapped back to one or more other modalities to be used for supervised learning.

4.1 Multi-View Learning

4.1.1 Representing a Data View or Sensory Modality

Given that data views as we present them here are essentially feature spaces containing data distributions in which we wish to perform unsupervised learning, there are a number of different ways in which they could be represented. There are two main strategies for unsupervised learning popular in the machine learning literature. *Clustering* involves grouping together similar samples within the data, usually based on some distance criteria. *Generative modelling*, on the other hand, usually involves attempting to form either parametric or non-parametric probabilistic models of the underlying data distribution from which the samples emerged. We will, for now however, find it instructive to develop a simpler model based on the concepts of clustering, vector quantization and competitive learning.

4.1.1.1 Clustering, Vector Quantization and Competitive Learning

Vector quantization can be viewed as a form of clustering, and indeed, that is one of our primary motivations for employing it in the following. Similarly to Coen [27–29], given a feature space, we seek to hyper-cluster the data distribution in the feature space such that we have a *codebook* of clusters which can subsequently be meta-clustered to find significant modes in the data distribution, which would represent significant sensory percepts. Here, hyper-clustering refers to the idea of forming an abundance of clusters, perhaps more clusters than are necessary to encode the significant modes of the data distribution, whereas meta-clustering refers to the idea of forming clusters of those clusters. In the specific case of multi-view affordance learning, these meta-clusters might correspond to significant perceptual events in the various data views/sensory modalities. In Coen's work, codebooks of hyper-clusters were used to facilitate co-clustering between separate sensory modalities via cross-modal Hebbian co-occurrence mappings. In our work, while we also make use of this cross-modal structure in a similar way to Coen, we are, by contrast, also concerned with employing the codebooks of hyper-clusters to facilitate online learning. Vector quantization lends itself to online learning owing to a number of well-established algorithms, both supervised and unsupervised, that use it as a basis. In the following, we provide a brief overview of vector quantization and show how it can be used for online competitive learning, before moving on to develop the multi-view structure of our algorithm.

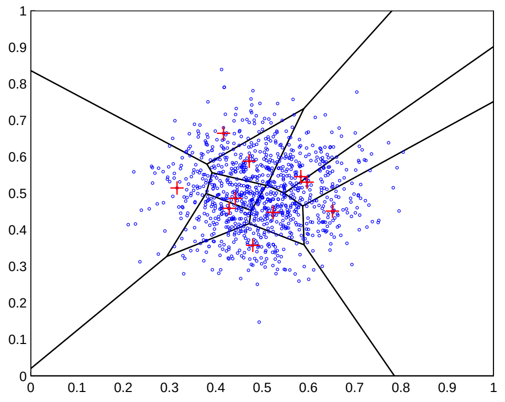


Figure **4.1:** A Voronoi diagram showing a simple 2-D Gaussian data distribution quantized by 10 prototype vectors randomly selected from the distribution.

Given a dataset

$$\mathcal{X} = \left\{ \mathbf{x}^i \in \mathbb{R}^n \mid i = 1, \ldots, N \right\}, \tag{4.1}$$

vector quantization [61, 86, 100] involves finding a set of *prototype vectors* collectively referred to as a codebook,

$$\mathcal{W} = \left\{ \mathbf{w}^i \in \mathbb{R}^n \mid i = 1, \ldots, M \right\}, \tag{4.2}$$

that forms a coarse, compressed representation of the probability distribution that the dataset is sampled from. The prototype vectors partition the space into *Voronoi regions* or *receptive fields* such that each prototype vector represents those data vectors for which it is the nearest neighbour and that thus fall within its receptive field as dictated by some distance metric, usually the Euclidean distance or squared Euclidean distance,

$$d^2(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n} \lambda_i (x_i - w_i)^2, \tag{4.3}$$

in which case the definition of the receptive field for prototype $\mathbf{w}^i$ would be

$$R^{\mathbf{w}^i} = \left\{ \mathbf{x} \in \mathcal{X} \,|\, \forall \mathbf{w}^j \in \mathcal{W} : \, d^2(\mathbf{x}, \mathbf{w}^i) \leq d^2(\mathbf{x}, \mathbf{w}^j) \right\}. \tag{4.4}$$

In Equation (4.3) above, the λ_i are weighting factors for each dimension which allows for the possibility of an adaptive metric for the purposes of feature relevance determination as discussed later in Chapter 5.

Usually, the goal of vector quantization is to minimize an error function such that the prototypes are positioned within the feature space in a way that optimizes their representation of the data distribution. This error function is often the *expected quantization (or distortion) error*, the continuous version of which is defined as

$$E = \sum_{i \in \mathcal{W}} \int_{R^c} d^2(\mathbf{x}, \mathbf{w}^i) p(\mathbf{x}) d\mathbf{x} \tag{4.5}$$

where $p(\mathbf{x})$ is the probability of the density function of $\mathbf{x}$. Minimization of the error function is usually achieved via *competitive learning*, where the prototype vectors $\mathbf{w}_i \in \mathcal{W}$ compete in a winner-take-all manner to represent the data points $\mathbf{x}_i \in X$ based on their proximity to each other. In particular, given a data sample x, the prototype $\mathbf{w}^c$ with the minimal distance to the sample,

$$\mathbf{w}^c = \underset{\mathbf{w}^i \in \mathcal{W}}{\arg\min}\, d^2(\mathbf{x}, \mathbf{w}^i), \tag{4.6}$$

may be variously termed as the *winning prototype*, *best-matching unit (BMU)* [86] or *nearest neighbour* to the sample, and we will use such terminology interchangeably in the remainder.

The classical minimization method is the *Linde-Buzo-Gray (or generalized Lloyd) (LBG)* algorithm [100, 101], which is guaranteed to decrease the distortion error, or at least leave it unchanged. It is, however, a batch method that requires access to the entire dataset at once, and as noted previously, we are interested in an online method that can update sample-by-sample. Another well-known method for data clustering, which is quite similar to the LBG algorithm, is the *k-means algorithm* [108]. k-means functions by planting k prototype vectors in the input space, associating each of the data vectors with a prototype vector, adjusting the prototype vectors to the mean of its associated data vectors, and then repeating this process until convergence. Though the above methods are amenable to online updating, we do not employ them for that purpose here, though we shall return to k-means later in this chapter in a different context. For now we turn our attention to self-organizing maps, which also allow for online vector quantization, as well as providing some other interesting properties that will prove useful for the development of our algorithm.

4.1.1.2 Self-Organizing Maps

The *self-organizing map (SOM)* (or self-organizing feature map) algorithm, developed by Kohonen [86], extends the idea of vector quantization by imposing a neighbourhood function on the prototype vectors in order to preserve the topological properties of the input space. Moreover, they can be trained in an online manner, such that given a data sample, the closest prototype as well as its topological neighbours get updated. This can be a useful property if we are hyper-clustering the data space, i.e. we have a large number of prototypes, and we wish to update many of them at once at each training step.

Given dataset $\mathcal{X}$ and codebook $\mathcal{W}$, the update rule for prototypes at each training step of the SOM algorithm is

$$\mathbf{w}^j_{t+1} = \mathbf{w}^j_t + \alpha^{\mathcal{W}}_t \, h^{\mathbf{w}^c}_t(\mathbf{w}^j) \, (\mathbf{x} - \mathbf{w}^j_t), \tag{4.7}$$

where h_t is the neighbourhood function for the winning prototype c, that is, the closest prototype to data sample $\mathbf{x}$ at time t, and $\alpha^{\mathcal{W}}_t$ is the learning rate at time t. The topology and neighbourhood function may in principle be defined almost arbitrarily, but often follow de facto definitions in the literature [86]. The topology, for instance, might be defined such that the prototypes are arranged in a rectangular sheet-shaped lattice such that, given prototype $\mathbf{w}^j$, it would have an associated location vector $\mathbf{r}^j \in \mathbb{R}^2$ governing its position within the lattice.

The neighbourhood kernel function is often a Gaussian function, such as

$$h_t^{\mathbf{w}^c}(\mathbf{w}^j) = \exp\left(-\frac{d(\mathbf{r}^c, \mathbf{r}^j)}{2\sigma_t^2}\right) \tag{4.8}$$

where the vectors $\mathbf{r}^c \in \mathbb{R}^2$ and $\mathbf{r}^j \in \mathbb{R}^2$ define 2-D lattice locations of the winning prototype $\mathbf{w}^c$ and prototype $\mathbf{w}^j$ respectively, and the parameter σ_t specifies the width of the kernel at time t. An alternative, simpler formulation would be

$$h_t^{\mathbf{w}^c}(\mathbf{w}^j) = \begin{cases} 1 & \text{if } \mathbf{r}^j \in B_{\sigma_t}(\mathbf{r}^c) \\ 0 & \text{otherwise,} \end{cases} \tag{4.9}$$

where $B_{\sigma_t}(\mathbf{r}^c)$ defines a ball with a centre at $\mathbf{r}^c$ and radius σ_t. This yields a neighbourhood set of prototypes

$$\mathcal{N}_t^{\mathbf{w}^c} = \left\{\mathbf{w}^i \in \mathcal{W} \,|\, h_t^{\mathbf{w}^c}(\mathbf{w}^i) \neq 0\right\} \tag{4.10}$$

around the winning prototype $\mathbf{w}^c$ at time t.

The SOM algorithm has its drawbacks. It does not have a well-defined error function and is not guaranteed to converge. In its original form, it also has a fixed network topology with a pre-specified, finite number of prototypes, which . Nevertheless, it serves as a useful basis for developing the self-supervised multi-view learner that can be trained online that we aim to develop in this chapter. In particular, we can use the fixed network topology to our advantage, in that it simplifies the mechanics of connecting two separate data view codebook layers together, as we shall see in the following subsection. In recent years, the general SOM approach has been placed on more principled footing where, for example, the prototypes are altered to become Gaussian kernels and the overall map represents a mixture of Gaussians [205]. A review of a number of such approaches can be found in [104].

4.1.1.3 Codebook Activations

An important consideration in the following is how the codebook relates to data samples in terms of an *activation function*. Such an activation function should show how closely each of the prototypes in the codebook match the data sample at a given time-step, thus allowing for an activation distribution over the entire codebook. Given that we will be using SOM training, we can exploit the SOM neighbourhood function and topological properties for this purpose. There are a few different potential approaches to this, one of which was provided by Miikkulainen [118] where, using the neighbourhood definitions in Equations (4.9) and (4.9), the activation function is defined as

$$a_t(\mathbf{w}^j) = \begin{cases} 1 - \frac{d(\mathbf{x},\mathbf{w}^j) - d_{\min}}{d_{\max} - d_{\min}} & \text{if } \mathbf{w}^j \in \mathcal{N}_t, \\ 0 & \text{otherwise,} \end{cases} \tag{4.11}$$

which gives the activation of prototype $\mathbf{w}^j$ at time t, where $d_{\min}$ is the smallest and $d_{\max}$ is the largest distance of data sample $\mathbf{x}$ to a prototype in the neighbourhood set $\mathcal{N}_t$. In the case of the Gaussian neighbourhood function from Equation (4.8), one option is to measure the distances between the distances between the data sample and the prototypes directly [1] with the following function

$$a_t(\mathbf{w}^j) = \frac{1}{1 + d(\mathbf{x}, \mathbf{w}^j)}, \tag{4.12}$$

but this can produce a rather coarse activation response from the codebook depending on far training has progressed. Another option is to use the Gaussian neighbourhood function itself as the activation function:

$$a_t(\mathbf{w}^j) = \exp\left(-\frac{d(\mathbf{r}^c, \mathbf{r}^j)}{2\sigma_t^2}\right). \tag{4.13}$$

For any of the above activation function definitions, we can take a probabilistic interpretation of the activation distribution over the entire codebook. We may define

$$P(\mathbf{w}^j|\mathbf{x}) = \frac{a_t(\mathbf{w}^j)}{\sum_i^{M_{\mathcal{W}}} a_t(\mathbf{w}^i)} \tag{4.14}$$

to be the conditional probability of the activation of prototype $\mathbf{w}^j \in \mathcal{W}$ given data sample $\mathbf{x}$, and

$$A_{\mathcal{W}}(\mathbf{x}) = \left\{P(\mathbf{w}^1|\mathbf{x}), P(\mathbf{w}^2|\mathbf{x}), \ldots, P(\mathbf{w}^{M_{\mathcal{W}}}|\mathbf{x})\right\} \tag{4.15}$$

to be the discrete spatial probability distribution function of the activation of codebook $\mathcal{W}$ given $\mathbf{x}$.

4.1.2 Connecting Data Views

In this subsection, we discuss how to connect separate data views, as represented by codebooks of prototypes, together in such a way that the connections represent

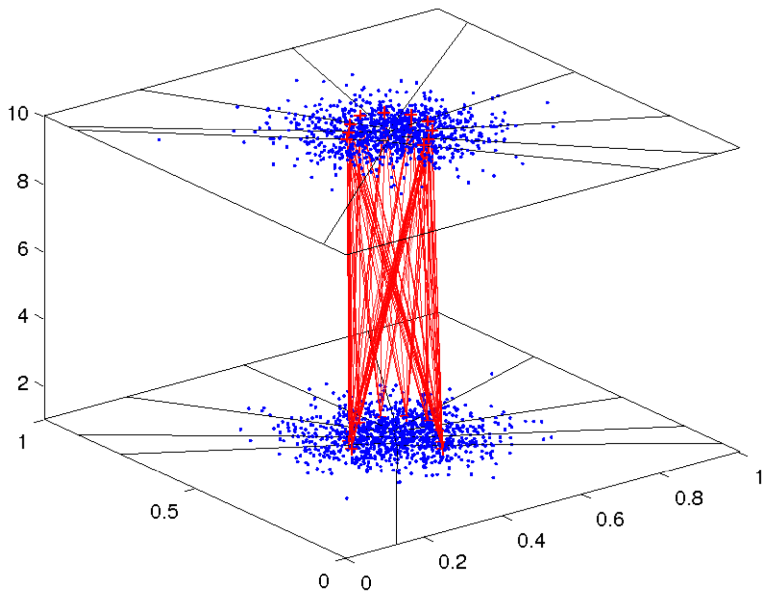


Figure **4.2:** This figure shows two Gaussian data distributions similar to that in Figure 4.1 divided into two separate data views which are hyper-clustered by self-organizing maps (SOMs) with full cross-view connections between the SOM prototypes. Voronoi tessellation is also illustrated for the prototypes in each view.

the co-occurrences between data in each view. Why would we want to segregate data views and connect them together in this way, when it seems more obvious to collate the data from all views and form a unified representation? Firstly, if we wish to use the naturally occurring classes, i.e. prominent clusters, in one view, to drive supervised learning in another view, then it makes sense to separate them. It is not so obvious how to achieve this over such a joint data distribution, particularly if online learning is a requirement. Secondly, as was demonstrated by de Sa [42], collation of data from separate data views can serve to hinder self-supervised cross-modal learning.

4.1.2.1 Hebbian Co-Occurrence Mapping

Assuming we have two different data views, each of them might yield two datasets of co-occurring data, $\mathcal{X}$ and $\mathcal{Y}$ respectively, defined as

$$\mathcal{X} = \left\{ \mathbf{x}^i \in \mathbb{R}^{n_x} \mid i = 1, \ldots, N \right\}, \tag{4.16}$$

$$\mathcal{Y} = \left\{ \mathbf{y}^i \in \mathbb{R}^{n_y} \mid i = 1, \ldots, N \right\}, \tag{4.17}$$

and we could represent each of the views with codebooks $\mathcal{W}$ and $\mathcal{V}$ as follows:

$$\mathcal{W} = \left\{ \mathbf{w}^i \in \mathbb{R}^{n_w} \mid i = 1, \ldots, M_{\mathcal{W}} \right\}, \tag{4.18}$$

$$\mathcal{V} = \left\{ \mathbf{v}^i \in \mathbb{R}^{n_v} \mid i = 1, \ldots, M_{\mathcal{V}} \right\}. \tag{4.19}$$

In the following, we may sometimes refer to $\mathcal{W}$ as the input view codebook and to $\mathcal{V}$ as the output view codebook without loss of generality. We adopt the same structural approach taken by both Coen [27–29] and Miikulainen [118] in that we define full network connectivity between the two codebooks with what we will term as a *Hebbian co-occurrence mapping* as follows:

$$\mathcal{H}(\mathcal{W}, \mathcal{V}) = \left\{ \gamma^{\mathbf{w}^i \mathbf{v}^j} \in \mathbb{R} \;\middle|\; \mathbf{w}^i \in \mathcal{W}, \mathbf{v}^j \in \mathcal{V}, \forall i, j \right\}, \tag{4.20}$$

where the $\gamma^{\mathbf{w}^i \mathbf{v}^j}$ are weights that are used to record the co-occurrence of winning prototypes between each of the data view codebooks. An example of two codebooks fully connected in such a manner is shown in Figure 4.2. For explanatory purposes, in Figure 4.2 and the remaining figures in this chapter, we will assume that the lower codebook is the input view codebook $\mathcal{W}$ and the upper codebook is output view codebook $\mathcal{V}$.

Working under the assumption that each of the codebooks are trained simultaneously in an online manner, using the SOM algorithm for example, then $\gamma^{\mathbf{w}^i \mathbf{v}^j}$ can simply be the number of times $\mathbf{w}^i$ was selected as the winning prototype in input codebook $\mathcal{W}$ at the same time-step that $\mathbf{v}^j$ was selected as the winning prototype in output codebook $\mathcal{V}$. This is essentially classical Hebbian learning [73] in that the weights between active prototypes increase proportional to their level of activity over time. Coen [27–29] uses this approach, though without the online training.

However, given a small training set, the above approach may result in a sparse weight matrix. Miikkulainen [118] provides a method of exploiting the topological structure of the SOM such that, not only are the Hebbian weights between the winning prototypes in each view updated, but also those weights between the neighbours of the winning prototypes. Such an update would make use of the neighbourhood activations from Equations (4.11), (4.12) or (4.13) by applying the following update [73, 118] to all weights in $\mathcal{H}(\mathcal{W}, \mathcal{V})$ at each training step:

$$\gamma_{t+1}^{\mathbf{w}^i \mathbf{v}^j} = \gamma_t^{\mathbf{w}^i \mathbf{v}^j} + \alpha_t^{\mathcal{H}} \, a_t(\mathbf{w}^i) \, a_t(\mathbf{v}^j), \tag{4.21}$$

where $\alpha_t^{\mathcal{H}}$ refers to a learning rate for the co-occurrence mapping which may be specified differently from the learning rates for the individual codebooks.

4.1.2.2 Hebbian Projections

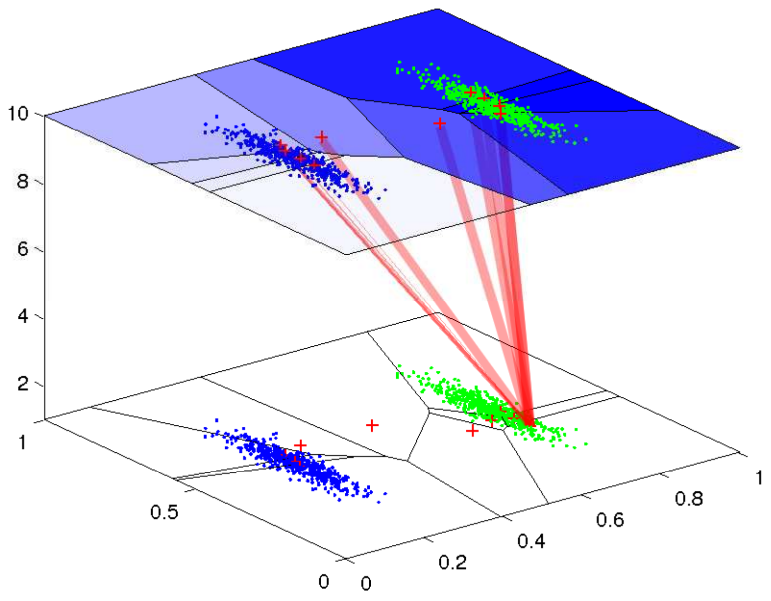


Figure **4.3:** An example of Hebbian projection from a prototype in a codebook for one data view onto the codebook for another view. The data consists of two Gaussian clusters mirrored sequentially in each view. After the codebooks are trained, along with the Hebbian co-occurrence mapping between the codebooks (cf. Section 4.1.3), the weights of the mapping from a prototype in the lower codebook are projected onto the prototypes of the upper codebook, as illustrated by the blue shading in the figure. The resulting distribution in the upper codebook is most strongly concentrated around the cluster corresponding via co-occurrence to the cluster in the lower data view that the projected prototype partially quantizes.

The Hebbian co-occurrence mapping can be used to map the relationship between the prototypes in the different data views and one way to achieve this is through the use of *Hebbian projections* [27–29]. A Hebbian projection is a spatial probability distribution over a codebook and is the result of selecting a prototype in one codebook and normalising the Hebbian co-occurrence weights that map from it onto another codebook. This is a useful, intuitive tool that allows us to measure how one data view looks from the perspective of another in terms of past co-occurrences of data.

Taking our two data view codebooks $\mathcal{W}$ and $\mathcal{V}$ as before, given prototype $\mathbf{w}^i \in \mathcal{W}$ we define

$$P(\mathbf{v}^j|\mathbf{w}^i) = \frac{\gamma^{\mathbf{w}^i\,\mathbf{v}^j}}{\sum_j^{M_\mathcal{V}} \gamma^{\mathbf{w}^i\,\mathbf{v}^j}} \tag{4.22}$$

to be the conditional probability of data co-occurring in the receptive field of prototype $\mathbf{v}^j \in \mathcal{V}$ given data occurring in the receptive field of prototype $\mathbf{w}^i \in \mathcal{W}$. We may now define a Hebbian projection from prototype $\mathbf{w}^i$ in codebook $\mathcal{W}$ to codebook $\mathcal{V}$ as

$$\vec{H}_\mathcal{W}^\mathcal{V}(\mathbf{w}^i) = \left\{P(\mathbf{v}^1|\mathbf{w}^i), P(\mathbf{v}^2|\mathbf{w}^i), \ldots, P(\mathbf{v}^{M_\mathcal{V}}|\mathbf{w}^i)\right\}. \tag{4.23}$$

An example of Hebbian projection is visualized in Figure 4.3. It is worth noting the similarity between this definition and the activation distribution function of Equation (4.15). in Section 4.1.1.3, and indeed, the relationship between activation distribution functions and Hebbian projections forms a core part of the proposed self-supervised multi-view learning algorithm discussed further below in Section 4.3.3.

4.1.3 Training an Unsupervised Multi-View Learner

In the previous sub-sections we established a way of representing the separate data views using codebooks of prototype vectors, as well as the co-occurrence of data between the data views using the Hebbian co-occurrence mapping. We now put those ideas together to describe the main training algorithm for our unsupervised multi-view learner, as well as a means performing online updates. The main idea that forms the basis of the unsupervised training is illustrated in Figure 4.4.

4.1.3.1 Main Training Algorithm

The main unsupervised training algorithm is detailed in Algorithm 1, which in turn makes use of Algorithm 2. Prior to the main loop, the training data for both the input view and output view are normalized in some way, usually such that each dimension is bound in the range $[0, 1]$. The respective codebooks in

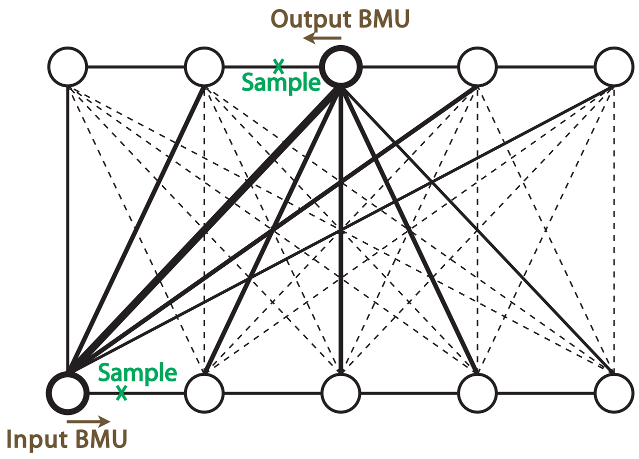


Figure **4.4:** Unsupervised multi-view training (cf. Algorithms 1 and 2): given co-occurring training samples for both an input view codebook (bottom) and an output view codebook (top), regular SOM training is performed in each view to the effect that the BMU prototypes as well as their neighbours are moved closer to the training samples in each view. The connections in the Hebbian co-occurrence mapping are also updated such that the weights between the BMUs and their neighbours are strengthened.

Algorithm 1 Unsupervised Multi-View Learner Training

Input: Input view codebook $\mathcal{W}$, output view codebook $\mathcal{V}$, Hebbian co-occurrence mapping $\mathcal{H}(\mathcal{W},\mathcal{V})$, training sets $\mathcal{X}_{\text{train}}$ and $\mathcal{Y}_{\text{train}}$, the number of training epochs E.

Output: Trained input view codebook $\mathcal{W}'$, trained output view codebook $\mathcal{V}'$, trained Hebbian co-occurrence mapping $\mathcal{H}'(\mathcal{W},\mathcal{V})$.

```
1: Normalize $\mathcal{X}_{\text{train}}$ and $\mathcal{Y}_{\text{train}}$.
2: Initialize $\mathcal{W}$ and $\mathcal{V}$.
3: for $e = 1, \ldots, E$ do
4:     for $t = 1, \ldots, N$ do
5:         Call Algorithm 2 or Algorithm 5 with inputs $\mathcal{W}$, $\mathcal{V}$, $\mathcal{H}(\mathcal{W},\mathcal{V})$, $\mathbf{x}^t \in \mathcal{X}_{\text{train}}$
           and $\mathbf{y}^t \in \mathcal{Y}_{\text{train}}$.
6:     end for
7: end for
```

each view are then initialized. This can be done in a number of different ways that can potentially affect the performance of the algorithms detailed here, particularly in the early stages of training. Each data dimension of each prototype might be randomly initialized within the normalization range, for instance. Alternatively, prototypes could be randomly sampled from the training data, or if information is available in advance about the ground truth classes in the data, the prototypes could be divided into class groups and initialized either by randomly sampling from the class training data, or taking the mean class values. Some of the experimental results of Section 7.1 in Chapter 7 demonstrate the effects of some of these different prototype initialization techniques. After normalization of the training data and initialization of the codebooks, learning commences for a certain number of training epochs by iteratively updating the output view codebook, the input view codebook and the Hebbian co-occurrence mapping. The number of training epochs can be an important consideration for many competitive learning algorithms such as SOM and others listed later in this chapter, depending on the size and dimensionality of the dataset. Training over an epoch implies training sample-by-sample for one run through the dataset, a process that may be repeated over several epochs. This type of training is used in order to simulate the existence of large datasets in the absence of them, or to simulate extended training durations, or both. This is discussed further in Section 7.1.2 of the experimental results chapter.

4.1.3.2 Online Updating

The unsupervised training procedure in Algorithm 1 makes iterative calls to an online update procedure described in Algorithm 2 Given training samples, this procedure updates both the input view codebook and the output view codebook, as well as the Hebbian co-occurrence mapping that connects them using the update equations from Sections 4.1.1 and 4.1.2. Prior to the updates the training samples must, of course, be normalized. Given novel data samples, this can be performed by scaling the novel samples within the range of previously observed and normalized training data, e.g. the training set referred to in Algorithm 1. Alternatively, though this has not been implemented in the research presented in this thesis, if fully-fledged online learning is required, online estimates of the scaling of the data in each dimension could be be obtained by maintaining multivariate Gaussian estimates of the incoming training data [6, pp. 94-97] [94].

Algorithm 2 Unsupervised Multi-View Learner Online Update

```
Input: Input view codebook $\mathcal{W}$, output view codebook $\mathcal{V}$, Hebbian co-occurrence
    mapping $\mathcal{H}(\mathcal{W}, \mathcal{V})$, training samples $\mathbf{x}$ and $\mathbf{y}$.
Output: Updated input view codebook $\mathcal{W}'$, updated output view codebook $\mathcal{V}'$,
    updated Hebbian co-occurrence mapping $\mathcal{H}'(\mathcal{W}, \mathcal{V})$.
 1: Given $\mathbf{y}$, find output view nearest-neighbour (BMU) $\mathbf{v}_t^c$ and its neighbours
    $\mathcal{N}_t^{\mathbf{v}^c}$ using Equations (4.6) and (4.10).
 2: for $i = 1, \ldots, M_{\mathcal{V}}$ do
 3:     Update $\mathbf{v}_{t+1}^i$ using Equation (4.7).
 4: end for
 5: Given $\mathbf{x}$, find input view BMU $\mathbf{w}_t^c$ and its neighbours $\mathcal{N}_t^{\mathbf{w}^c}$ using Equations
    (4.6) and (4.10).
 6: for $i = 1, \ldots, M_{\mathcal{W}}$ do
 7:     Update $\mathbf{w}_{t+1}^i$ using Equation (4.7).
 8: end for
 9: for $i = 1, \ldots, M_{\mathcal{W}}$ do
10:     for $j = 1, \ldots, M_{\mathcal{V}}$ do
11:         Get prototype activations $a_t(\mathbf{w}^i)$ and $a_t(\mathbf{v}^j)$ using Equation (4.11),
            (4.12), or (4.13).
12:         Update $\gamma_{t+1}^{\mathbf{w}^i\mathbf{v}^j} \in \mathcal{H}(\mathcal{W}, \mathcal{V})$ using Equation (4.21).
13:     end for
14: end for
```

4.2 Unsupervised Multi-View Discriminative Learning

Using the concepts developed above, we are now in a position to develop some basic multi-view predictive capabilities. We can simultaneously train the codebooks in each view and the multi-view co-occurrence mapping online, sample-by-sample using Algorithms 1 and 2. The structure of the network also means that we can project co-occurrence information from the codebook in one view onto the codebook in the other as demonstrated in Section 4.1.2.2. This gives us a stepping stone to performing both unsupervised regression and classification from one data view to another and that is the subject of this section. In the case of regression, we would like to form a mapping $f : \mathbb{R}^{n_x} \to \mathcal{V}$ that maps input view samples to output view prototypes, whereas in the case of classification, we would require a mapping $g : \mathbb{R}^{n_x} \to L(\mathcal{V})$ that maps input view samples to class labels, where

$L()$ is some labelling function. The general approach in both cases is similar. Given an input view sample $\mathbf{x}$, the nearest neighbour $\mathbf{w}^c$ is found in the input view codebook and is mapped to either an output view prototype or cluster of prototypes via the Hebbian co-occurrence mapping. These ideas are explained in more detail in the following.

4.2.1 Regression

Algorithm 3 Cross-View Regression

```
Input: Input view Codebook $\mathcal{W}$, output view codebook $\mathcal{V}$, Hebbian co-
    occurrence mapping $\mathcal{H}(\mathcal{W},\mathcal{V})$, test set $\mathcal{X}_{\text{test}} = \left\{\mathbf{x}^1,\ldots,\mathbf{x}^{N_t}\right\}$.
Output: Predictions $\mathcal{Y}' = \left\{\mathbf{y}'^1,\ldots,\mathbf{y}'^{N_t}\right\}$.
 1: if $\mathcal{X}_{\text{test}}$ is not normalized then
 2:     Normalize $\mathcal{X}_{\text{test}}$.
 3: end if
 4: for $i = 1,\ldots,N_t$ do
 5:     Given test sample $\mathbf{x}^i \in \mathcal{X}_{\text{test}}$, find input view nearest neighbour (BMU) $\mathbf{w}^c$
        using Equation (4.6).
 6:     Perform Hebbian projection $\vec{H}^{\mathcal{V}}_{\mathcal{W}}(\mathbf{w}^c)$ to find posterior probabilities
        $\{P(\mathbf{v}^j|\mathbf{w}^c) \mid \mathbf{v}^j \in \mathcal{V}\}$ using Equation (4.23).
 7:     Assign $\mathbf{y}'^i := \arg\max_{\mathbf{v}^j\in\mathcal{V}} P(\mathbf{v}^j|\mathbf{w}^c)$.
 8: end for
```

Given the structure we already have in place, it is a relatively small matter to apply a crude form of regression. An algorithm for this purpose is detailed in Algorithm 3. Given an input view test sample $\mathbf{x}$, after finding its nearest neighbour $\mathbf{w}^c$ in the input view codebook via Equation (4.6), a Hebbian projection may be performed from $\mathbf{w}^c$ to the output view codebook to find the posterior probabilities of the output view prototypes given $\mathbf{w}^c$ and a target prediction $\mathbf{y}'$ may be assigned to the output view prototype with the maximum posterior probability. While this is potentially quite useful, our primary interest as outlined in Chapter 3 is in dynamically finding class clusters in the output view and associating them with percepts in the input view, a topic we turn to in the following sub-section.

4.2.2 Classification

As was motivated at the beginning of this chapter, after hyper-clustering the data in the data views with codebook prototypes we seek to form meta-clusters over the prototypes themselves. These meta-clusters would represent the naturally occurring classes in the data as quantized by the hyper-clustering of the prototype vectors. Our interest is in finding these classes in one driving data view which would be used to train a classifier in a separate data view. The means of mapping the class clusters from the driving view codebook to the classifying view codebook is provided by Hebbian projection, and the class cluster projection process is detailed in Section 4.2.2.2 below. Prior to that however, we must decide how to perform the class meta-clustering in the first place, and that is the subject of the next sub-section. In the specific case of object affordance learning as formulated in Chapter 3, the driving data view would be the effects view, representing the post-action motion of objects, whereas the classifying data view would be the object features view, representing the shapes of objects significant to their action-induced behaviour. The idea in that particular setting would be to form class clusters in the driving effects view and to map them back to the classifying object features view, which would then be able to form effect-class predictions from object feature vectors.

4.2.2.1 Meta-Clustering within a Data view

In order to find the meta-clusters of prototypes, we treat the prototype vectors as data points and employ traditional unsupervised clustering, specifically, the k-means algorithm discussed previously in Section 4.1.1.1. One issue with the regular k-means clustering algorithm is that k must be selected in advance. It is possible, however, to augment the algorithm such that k is selected automatically. We achieve this by running k-means for multiple different values of k, then evaluating each of the clusterings using multiple different cluster validity indices, and finally selecting the k-value which is most consistently selected as the best value by these.

Thus, given some set $\mathcal{X}$ we define a collection $\mathcal{C}^{\mathcal{X}}$ of K possible k-clusterings or k-partitions of the elements of $\mathcal{X}$ as follows

$$\mathcal{C}^{\mathcal{X}} = \left\{ C_k^{\mathcal{X}} = \bigcup_{i=1}^{k} \{X_i\} \;\middle|\; X_i \subseteq \mathcal{X}, k = 1, \ldots, K \right\} \tag{4.24}$$

where the X_i sets are the individual clusters of elements in each $C_k^{\mathcal{X}}$ clustering. The validity indices we use to evaluate a given k-clustering $C_k^{\mathcal{X}}$ are the following:

- Davies-Bouldin [9, 36], which aims at identifying sets of clusters that are compact and well separated. It is defined as

$$\mathrm{DB}(C_k^{\mathcal{X}}) = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} \left\{ \frac{\Delta(X_i) + \Delta(X_j)}{\delta(X_i, X_j)} \right\} \tag{4.25}$$

  where $\delta(X_i, X_j)$ defines inter-cluster distance between clusters X_i and X_j and $\Delta(X_i)$ represents the intra-cluster distance of cluster X_i.

- Dunn [9, 48], which has a similar aim to the Davies-Bouldin index and is defined as

$$\mathrm{DU}(C_k^{\mathcal{X}}) = \min_{1 \leq i \leq k} \left\{ \min_{\substack{1 \leq j \leq k \\ j \neq i}} \left\{ \frac{\delta(X_i, X_j)}{\max_{1 \leq l \leq k} \{\Delta(X_l)\}} \right\} \right\} \tag{4.26}$$

  where δ, Δ, X_i and X_j are defined as before.

- Calinski-Harabasz [15, 47], where for each number of clusters $k \geq 2$, the index is defined as

$$\mathrm{CH}(C_k^{\mathcal{X}}) = \frac{\mathrm{tr}\mathbf{B}_k/(k-1)}{\mathrm{tr}\mathbf{W}_k/(k-1)} \tag{4.27}$$

  where $\mathbf{B}_k$ and $\mathbf{W}_k$ are the matrices of between and within k-clusters sums of squares and cross-products and tr denotes the trace of a matrix.

- Krzanowski-Lai [47, 97], defined as

$$\mathrm{KL}(C_k^{\mathcal{X}}) = \frac{|\mathrm{diff}(k)|}{|\mathrm{diff}(k+1)|} \tag{4.28}$$

  where

$$\mathrm{diff}(k) = (k-1)^{\frac{2}{p}}\mathrm{tr}\mathbf{W}_{k-1} - k^{\frac{2}{p}}\mathrm{tr}\mathbf{W}_k. \tag{4.29}$$

- The silhouette index [9, 148], which assigns a quality measure

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} \tag{4.30}$$

  called the silhouette width to the ith sample of a given cluster $X_j = \{x_1, \ldots, x_m\}$ where $a(x_i)$ is the average distance between the ith sample and all of the samples in X_j and $b(x_i)$ is the minimum average distance between the ith sample and all of the samples in $\{X_l \,|\, \forall l \neq j\}$. The silhouette index may then be defined as

$$\mathrm{SI}(C_k^{\mathcal{X}}) = \frac{1}{k}\sum_{j=1}^{k}\left\{\frac{1}{m}\sum_{i=1}^{m}\{s(x_i)\,|\,x_i \in X_j\}\right\}. \tag{4.31}$$

Taking the output view codebook $\mathcal{V}$, we may then select a k^*-clustering from a given set of clusterings $\mathcal{C}^{\mathcal{V}}$ that is optimal under the above validity indices as follows

$$k^* = \arg\max_k \sum_{i=1}^{5} \mathbf{1}_{f_i(\mathcal{C}^{\mathcal{V}})}(k) \tag{4.32}$$

where $\mathbf{1}_A(x)$ is the indicator function, and the f_i are functions such that $f_i : \mathbb{P}(\mathcal{V}) \to \mathbb{N}$, with $f_1(\mathcal{C}^{\mathcal{V}}) = \arg\min_k \mathrm{DB}(C_k^{\mathcal{V}})$, $f_2(\mathcal{C}^{\mathcal{V}}) = \arg\max_k \mathrm{DU}(C_k^{\mathcal{V}})$, $f_3(\mathcal{C}^{\mathcal{V}}) = \arg\max_k \mathrm{CH}(C_k^{\mathcal{V}})$, $f_4(\mathcal{C}^{\mathcal{V}}) = \arg\max_k \mathrm{KL}(C_k^{\mathcal{V}})$, and $f_5(\mathcal{C}^{\mathcal{V}}) = \arg\max_k \mathrm{SI}(C_k^{\mathcal{V}})$.

4.2.2.2 Projecting Class Clusters Between Data views

The $C_{k^*}^{\mathcal{V}}$ clusters found by meta-clustering the output view codebook $\mathcal{V}$ define the classes to be used for classifying data within input view codebook $\mathcal{W}$. Given an input view test sample $\mathbf{x}$ to be classified, the winning prototype (cf. Equation (4.6) $\mathbf{w}^i$ in the input view layer is found and its Hebbian weight links are mapped to the output view layer meta-clusters via Hebbian projection as follows where, given cluster $V^l \in C_{k^*}^{\mathcal{V}}$, the projection is

$$\vec{H}_{\mathcal{W}}^{V^l}(\mathbf{w}^i) = \left\{ P(\mathbf{v}^j|\mathbf{w}^i) \in \vec{H}_{\mathcal{W}}^{\mathcal{V}}(\mathbf{w}^i) \,\middle|\, \forall j : \mathbf{v}^j \in V^l \right\}. \tag{4.33}$$

4.2.2.3 Assigning Class Cluster Labels and Probabilities to Prototypes

By summing the posterior probabilities $P(\mathbf{v}^j|\mathbf{w}^i)$ provided by such a projection, we can determine the likelihood of class cluster V^l in output view codebook $\mathcal{V}$ given prototype $\mathbf{w}^i$ in input view codebook $\mathcal{W}$ as follows

$$P(V^l|\mathbf{w}^i) = \int_{R^{V^l}} P(V^l|\mathbf{w}^i)P(\mathbf{w}^i)d\mathbf{v} \tag{4.34}$$

$$= \sum_{\mathbf{v}^j \in V^l} P(\mathbf{v}^j|\mathbf{w}^i)P(\mathbf{w}^i) \tag{4.35}$$

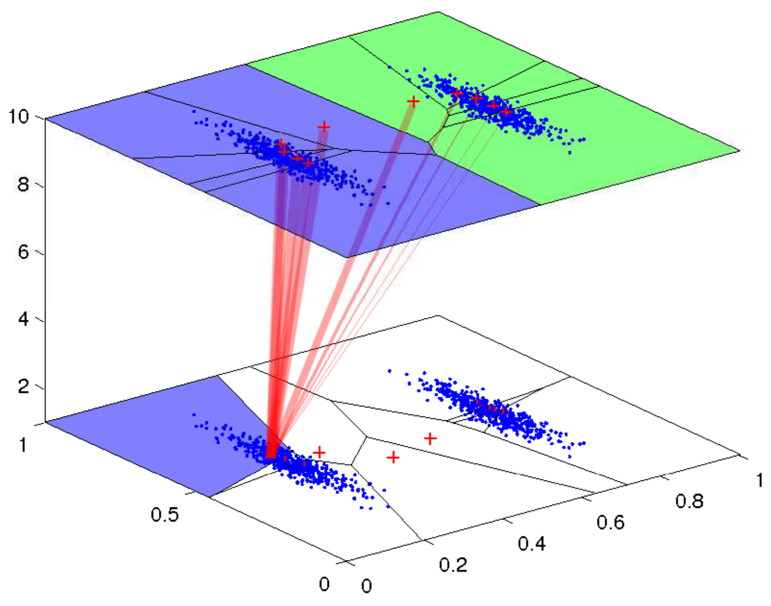


Figure **4.5:** Projecting meta-clusters from one data view codebook onto another for cross-view clustering. After both the codebooks and co-occurrence mapping are trained (cf. Sections 4.1.1 and 4.1.2, Figure 4.2), the upper codebook is meta-clustered using the k-means algorithm (cf. Section 4.2.2.1) to form class labels. These class labels are then projected onto the prototypes in the lower codebook using the Hebbian co-occurrence weights to determine the appropriate labels (cf. Section 4.2.2.2).

where $R^{V^l} = \bigcup_{\mathbf{v}^j \in V^l} R^{\mathbf{v}^j}$ is the receptive field for class cluster V^l.

This allows us to assign an output view class cluster label to each of the prototypes in the input view by maximising the class cluster posterior probability for each of them. Thus, given $\mathbf{w}^i$, we define a labelling function

$$L(\mathbf{w}^i) = \underset{l=1,\ldots,k^*}{\arg\max}\, P(V^l|\mathbf{w}^i) \tag{4.36}$$

that labels the input view prototypes on that basis.

4.2.2.4 Culling Unreliable Prototypes

Using the above labelling technique, we can assign labels to each of the input view prototypes using the output view class cluster posterior probabilities. However, on that basis alone, not all of the prototypes in the input view will be reliable for classification. In particular, if the class cluster posterior probabilities for a given input view prototype are more or less the same, it may be a mistake to assign a class label to it at all. It might make more sense to cull that prototype from the classification process altogether. By applying this idea to all of the prototypes in the input view codebook, it would therefore be left to the remaining, more reliable, labelled prototypes to classify input samples. This would have the effect of broadening the hypothesis margins [34] between the remaining prototypes, thus potentially increasing classification accuracy. To accommodate this prototype culling idea, we may modify the prototype labelling function from Equation (4.36) as follows

$$L(\mathbf{w}^i) = \begin{cases} \arg\max\limits_{l=1,\dots,k^*} P(V^l|\mathbf{w}^i) & \text{if } P(V^l|\mathbf{w}^i) > \epsilon, \\ 0 & \textit{otherwise}, \end{cases} \tag{4.37}$$

where ϵ is some threshold, which in practice we have taken to be the mean class cluster posterior probability

4.2.2.5 Unsupervised Classification

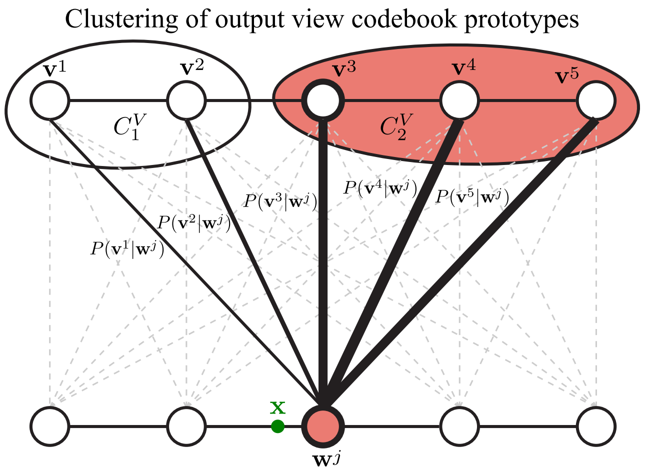

Clustering of output view codebook prototypes
$\mathbf{v}^1$
$\mathbf{v}^2$
$\mathbf{v}^3$
$\mathbf{v}^4$
$\mathbf{v}^5$
C_1^V
C_2^V
$P(\mathbf{v}^3|\mathbf{w}^j)$
$P(\mathbf{v}^4|\mathbf{w}^j)$
$P(\mathbf{v}^5|\mathbf{w}^j)$
$P(\mathbf{v}^2|\mathbf{w}^j)$
$P(\mathbf{v}^1|\mathbf{w}^j)$
$\mathbf{x}$
$\mathbf{w}^j$

Figure **4.6:** Cross-view classification: output nodes are meta-clustered and the cluster with the strongest weighted connections to the input nearest neighbour (BMU) wins.

The techniques outlined above lead us to the classification procedure depicted in Figure 4.6 and formulated in Algorithm 4. In this algorithm, by employing cross-view Hebbian projection, input view test samples may be classified in terms

Algorithm 4 Cross-View Classification

```
Input: Input view Codebook $\mathcal{W}$, output view codebook $\mathcal{V}$, Hebbian co-occurrence mapping $\mathcal{H}(\mathcal{W}, \mathcal{V})$, test set $\mathcal{X}_{\text{test}}$.
Output: Class cluster labels $\mathcal{Y}'_{\text{labels}}$ and class cluster posterior probabilities $\mathcal{Y}'_{\text{probs}}$.
 1: if $\mathcal{X}_{\text{test}}$ is not normalized then
 2:     Normalize $\mathcal{X}_{\text{test}}$.
 3: end if
 4: Find optimal $k$-clustering $C^{\mathcal{V}}_{k^*}$ by meta-clustering output view codebook $\mathcal{V}$ using the algorithm described in Section 4.2.2.1.
 5: for $i = 1, \ldots, N$ do
 6:     Given test sample $\mathbf{x}^i \in \mathcal{X}_{\text{test}}$, find input view nearest neighbour (BMU) $\mathbf{w}^c$ (cf. Equation (4.6)).
 7:     Perform Hebbian projection $\vec{H}^{\mathcal{V}}_{\mathcal{W}}(\mathbf{w}^i)$ to find posterior probabilities $\{P(\mathbf{v}^j|\mathbf{w}^c) \mid \mathbf{v}^j \in \mathcal{V}\}$ using Equation (4.23).
 8:     for $k = 1, \ldots, k^*$ do
 9:         Record $\mathcal{Y}'_{\text{probs}}(i, k) = P(V^k|\mathbf{w}^c)$, the posterior probability for class cluster $V^k \in C^{\mathcal{V}}_{k^*}$.
10:     end for
11:     Find the optimal class cluster label for $\mathbf{w}^c$ using Equation (4.36) or (4.37) and record as $\mathcal{Y}'_{\text{labels}}(i)$, the output class label for sample $\mathbf{x}^i$.
12: end for
```

of the class clusters found in the output view via unsupervised meta-clustering of the output view prototypes. Moreover, class cluster posterior probabilities may also be associated with the test samples. It should also be noted that the classification procedure can be employed at any given point during training. The meta-clustering of the output view prototypes would give the best current estimate of the class clusters at a given training step t, and those would be used as the current class labels at time t when classifying. At this juncture, it would seem that some of the main requirements for our multi-view learner have been met. However, while the combination of the multi-view training and classification machinery outlined thus far might well work for unsupervised learning and classification, it would not be entirely appropriate to refer to it as self-supervised yet. It would perhaps be more apt to refer to it as a type of unsupervised classification since, while class clusters are indeed discovered in the output view and cross-view associated with the prototypes in the input view, the input view codebook is not being explicitly trained as a classifier. There is no supervisory signal during training, decision boundaries between prototypes are not being properly accounted for (aside from the prototype culling procedure in Section 4.2.2.4, and the input view is not generally being optimized for classification purposes. In the following section, we address this deficiency.

4.3 Self-Supervised Multi-View Discriminative Learning

So far, Algorithm 4 gives us a means of associating prototypes in the input data view with class clusters of prototypes in the output data view, but there is no guarantee that the input view prototypes are appropriately positioned in the input space for this purpose. Given an input view sample, the nearest neighbour rule of Equation (4.6) will find the nearest neighbour prototype in the input view based on the distance between the sample and the prototypes, but since the positions of the prototypes are not optimized to be discriminative with respect to the output view class clusters, the sample may be misclassified. Using the nearest-neighbour rule and labelled prototypes in this way for the purposes of classification essentially interprets the labelled prototypes as piecewise-linear classifiers with linear decision borders between them. By re-positioning the prototypes in the input space, it is therefore possible to adjust such decision borders to improve discriminative capabilities.

In the usual case of fully-supervised learning, where labelled samples are present, there are a number of different potential classifiers that could be employed, such as the *Naive Bayes classifier*, *support vector machines (SVMs)*, *logistic regression*, amongst many others [6]. One particular breed of classifiers, however, that of *learning vector quantization (LVQ)* [86], fits rather naturally with our learning framework. LVQ-based classifiers take labelled prototype vectors as piecewise-linear classifiers and, given a training set of labelled input samples, optimize the decision borders by repositioning the prototypes with respect to the input samples depending on whether they were correctly or incorrectly classified using the nearest-neighbour rule. We describe LVQ-based classifiers in more detail in Section 4.3.1. Since our multi-view training data is unlabelled and we form class-cluster labels via unsupervised clustering in the output view, we could opt for something akin to the following procedure:

1. Dynamically generate class clusters in the output view codebook at each training step,
2. Match the output view sample to one of the class clusters, thus providing a label for the co-occurring input view sample.
3. Map the class cluster labels back to the input view codebook via Hebbian projection, thus labelling the input view prototypes.
4. Proceed with LVQ-based training using the labelled prototypes and input samples.

However, the first step of meta-clustering the output view prototypes at each training step could prove to be quite computationally expensive and, as we shall see in the following sub-sections, is unnecessary. Rather than using fully-fledged class labels, we could instead construct an error signal that determines whether or not a given input view prototype is an appropriate classifier for a given input sample based on its co-occurrence Hebbian projection to the output view, as well as the activation of the output view codebook given the co-occurring output sample. Before broaching that subject in earnest, we must first describe how learning vector quantization works.

4.3.1 Learning Vector Quantization

Learning vector quantization (LVQ) [86] provides an intuitive, and often highly effective, means for discriminative learning where prototype vectors are used to quantize the input feature space and given labels to form piecewise-linear classifiers using the nearest neighbour rule. The basis of LVQ learning rules hinges on a simple idea. Given a training sample, if the label of the nearest neighbour prototype matches the class of the sample, then the prototype is moved closer to the sample, whereas if the label does not match, the prototype is moved further away from the sample. The resultant effect of this is that the local piecewise-linear decision borders between prototypes are adjusted relative to the training data.

Since their introduction, LVQ algorithms have undergone various analyses and seen various improvements to their design. The original formulations *(LVQ1, LVQ2, LVQ3)* [86] have been shown to be divergent, inspiring the *generalized* *learning vector quantization (GLVQ)* algorithm [158] where the prototypes are updated such that a stochastic gradient descent [146] is performed over an error function. LVQ algorithms have also been shown to be a family of maximum margin classifiers [34], thus providing excellent generalization for novel data with high-dimensional inputs.

In order to formulate the LVQ algorithms, we begin by modifying our original dataset definition in Equation (4.1) to include class labels:

$$X = \left\{(\mathbf{x}^i, y^i) \subset \mathbb{R}^n \times \{1, \ldots, C\} \mid i = 1, \ldots, N\right\}. \tag{4.38}$$

Similarly, we alter the original codebook definition in Equation (4.2) such that the prototypes also have labels:

$$\mathcal{W} = \left\{(\mathbf{w}^i, c^i) \subset \mathbb{R}^n \times \{1, \ldots, C\} \mid i = 1, \ldots, M\right\}. \tag{4.39}$$

Given a sample $(\mathbf{x}, y) \in X$, we denote by $g(\mathbf{x})$ a function that is negative if $\mathbf{x}$ is classified correctly, i.e. $\mathbf{x} \in R^i$ with $c^i = y$, and is positive if $\mathbf{x}$ is classified incorrectly, i.e. $\mathbf{x} \in R^i$ with $c^i \neq y$. We also let $f : \mathbb{R} \to \mathbb{R}$ be some monotonically increasing function.

The goal of the GLVQ algorithm [158] is to minimize

$$E = \sum_{i=1}^{m} f(g(\mathbf{x}^i)) \tag{4.40}$$

via stochastic gradient descent. The update rules for GLVQ as well as many other LVQ algorithms can be expressed using the above notation. In the following subsections, the LVQ1 [86] and GLVQ [158] algorithms are reviewed before demonstrating in Section 4.3.2 how such algorithms can be trained using error signals other than explicit class labels. In Chapter 5 we review the RLVQ [8] and GRLVQ [69] algorithms, which include feature relevance determination mechanisms, before introducing novel LVQ algorithms with feature relevance mechanisms of our own creation.

4.3.1.1 LVQ1

Given a training sample $(\mathbf{x}, y) \in X$, by letting $f(x) = x$ and $g(\mathbf{x}) = \eta d_j$ where $d_j = d^2(\mathbf{x}, \mathbf{w}^j)$ with $\mathbf{w}^j$ being the closest prototype to $\mathbf{x}$ and $\{\lambda_i = 1\}_{i=1}^m$ (i.e. equal weights for regular Euclidean distance), with $\eta = 1$ if $\mathbf{x}$ is classified correctly (i.e.

$c_j = y$) and $\eta = -1$ if $\mathbf{x}$ is classified incorrectly (i.e. $c_j \neq y$), the following stochastic gradient descent update rule may be derived for LVQ1 [86]:

$$\mathbf{w}_{t+1}^j = \begin{cases} \mathbf{w}_t^j + \alpha(\mathbf{x} - \mathbf{w}_t^j), & \text{if } c^j = y \\ \mathbf{w}_t^j - \alpha(\mathbf{x} - \mathbf{w}_t^j), & \text{otherwise,} \end{cases} \tag{4.41}$$

where α is the learning rate and the t subscripts denote prototype states at different training steps. However, it should be noted that the error function as defined here is highly discontinuous, and thus can lead to instabilities in the algorithm. GLVQ, discussed next, was designed to resolve this issue.

4.3.1.2 GLVQ

Here, $d_j = d^2(\mathbf{x}, \mathbf{w}^j)$ is defined where $\mathbf{w}^j$ is the closest prototype to $\mathbf{x}$ with label $c^j = y$ and $d_k = d^2(\mathbf{x}, \mathbf{w}^k)$ where $\mathbf{w}^k$ is the closest prototype to $\mathbf{x}$ with some other label. By letting

$$g(\mathbf{x}) = \frac{d_j - d_k}{d_j + d_k} \tag{4.42}$$

and

$$f_t(g(\mathbf{x})) = \frac{1}{1 + \exp^{-g(\mathbf{x})t}}, \tag{4.43}$$

which is a sigmoidal function that redefines the error function in Equation (4.40) such that it is continuous over borders between the receptive fields for $\mathbf{w}^j$ and $\mathbf{w}^k$. When minimized, the error function yields the following update rules for $\mathbf{w}^j$ and $\mathbf{w}^k$ [158]:

$$\mathbf{w}_{t+1}^j := \mathbf{w}_t^j + \alpha \frac{\partial f}{\partial g} \frac{d_k}{(d_j + d_k)^2} (\mathbf{x} - \mathbf{w}_t^j) \tag{4.44}$$

$$\mathbf{w}_{t+1}^k := \mathbf{w}_t^k + \alpha \frac{\partial f}{\partial g} \frac{d_j}{(d_j + d_k)^2} (\mathbf{x} - \mathbf{w}_t^k) \tag{4.45}$$

where

$$\frac{\partial f}{\partial g} = f_t(g(\mathbf{x}))(1 - f_t(g(\mathbf{x})). \tag{4.46}$$

GLVQ, unlike LVQ1 or the rest of Kohonen's original LVQ formulations, has been shown to be convergent [69, 158], although it is quite sensitive to the initialization of the prototype vectors, a fact that is demonstrated in the experimental results of Chapter 7.

4.3.2 Training LVQ Classifiers with Probabilities instead of Labels

By making simple modifications to the LVQ1 update of Equation (4.41) and the GLVQ updates of Equations (4.44) and (4.45) it is possible to provide update equations that rely on error signals as opposed to explicit class labels. If, for example, we were to have a function $0 \leq \phi(\mathbf{w}^j) \leq 1$ indicating the probability of prototype $\mathbf{w}^j$ correctly classifying the current sample $\mathbf{x}$, then the LVQ1 update might be modified to:

$$\mathbf{w}^j_{t+1} := \begin{cases} \mathbf{w}^j_t + \alpha(\mathbf{x} - \mathbf{w}^j_t), & \text{if } \phi(\mathbf{w}^j_t) \geq \epsilon \\ \mathbf{w}^j_t - \alpha(\mathbf{x} - \mathbf{w}^j_t), & \text{otherwise,} \end{cases} \tag{4.47}$$

where $\mathbf{w}^j$ is the nearest-neighbour prototype as before and ϵ is some decision threshold, or

$$\mathbf{w}^j_{t+1} := \begin{cases} \mathbf{w}^j_t + \alpha(\mathbf{x} - \mathbf{w}^j_t), & \text{if } \phi(\mathbf{w}^j_t) \geq \epsilon + \delta \\ \mathbf{w}^j_t - \alpha(\mathbf{x} - \mathbf{w}^j_t), & \text{if } \phi(\mathbf{w}^j_t) \leq \epsilon - \delta \\ \mathbf{w}^j_t & \text{otherwise,} \end{cases} \tag{4.48}$$

if we wished to allow a window of uncertainty $\pm\delta$ around ϵ.

In the case of GLVQ, we would first have to decide on appropriate choices for both $\mathbf{w}^j$, the nearest neighbour prototype of matching class to sample $\mathbf{x}$, and $\mathbf{w}^k$, the nearest neighbour prototype of a non-matching class. Using the ϕ probability function, as opposed to class labels, we could use the following definitions:

$$\mathbf{w}^j = \arg\min_{\mathbf{w}^l} \left\{ d^2(\mathbf{x}, \mathbf{w}^l) \,\middle|\, \forall l : \mathbf{w}^l \in \left\{ \mathbf{w}^i \in \mathcal{W} \,|\, \forall i : \phi(\mathbf{w}^i) \geq \epsilon + \delta \right\} \right\}, \tag{4.49}$$

$$\mathbf{w}^k = \arg\min_{\mathbf{w}^l} \left\{ d^2(\mathbf{x}, \mathbf{w}^l) \,\middle|\, \forall l : \mathbf{w}^l \in \left\{ \mathbf{w}^i \in \mathcal{W} \,|\, \forall i : \phi(\mathbf{w}^i) < \epsilon - \delta \right\} \right\}, \tag{4.50}$$

where ϵ is once again a threshold value with window δ. Then by modifying $g(\mathbf{x})$ in Equation (4.42) to reflect the uncertainty about the class membership of prototypes $\mathbf{w}^j$ and $\mathbf{w}^k$, we may derive the following update rules:

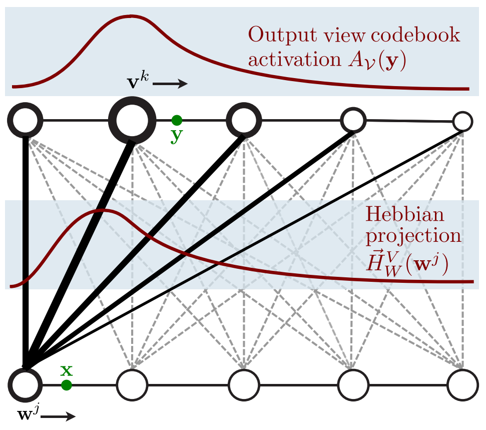
$$\mathbf{w}^j_{t+1} := \mathbf{w}^j_t + \alpha \frac{\partial f}{\partial g} \frac{d_k(\phi(\mathbf{w}^j_t) + \phi(\mathbf{w}^k_t) + 1)}{(d_j + d_k)^2} (\mathbf{x} - \mathbf{w}^j_t), \tag{4.51}$$

$$\mathbf{w}^k_{t+1} := \mathbf{w}^k_t + \alpha \frac{\partial f}{\partial g} \frac{d_j(\phi(\mathbf{w}^j_t) + \phi(\mathbf{w}^k_t) + 1)}{(d_j + d_k)^2} (\mathbf{x} - \mathbf{w}^k_t), \tag{4.52}$$

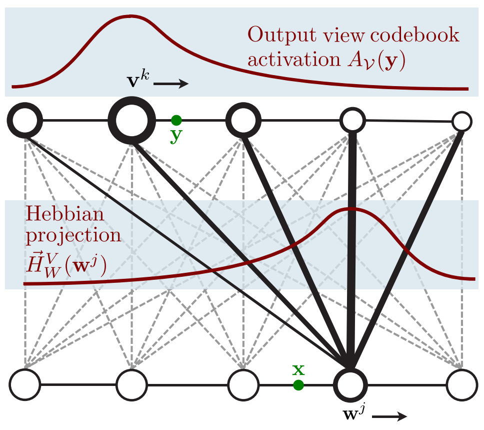
where

$$\frac{\partial f}{\partial g} = f_t(g(\mathbf{x}))(1 - f_t(g(\mathbf{x})) \tag{4.53}$$

as before. The specification of the $g(\mathbf{x})$ function in this case, as well as the derivation of the above update rules by stochastic gradient descent are included in Appendix A.


Output view codebook activation $A_{\mathcal{V}}(\mathbf{y})$
$\mathbf{v}^k$
$\mathbf{y}$
Hebbian projection $\vec{H}_W^V(\mathbf{w}^j)$
$\mathbf{x}$
$\mathbf{w}^j$

(a) A close similarity between the output view codebook activation and the Hebbian projection from the nearest neighbour in the input view codebook indicates that the input sample is correctly matched.


Output view codebook activation $A_{\mathcal{V}}(\mathbf{y})$
$\mathbf{v}^k$
$\mathbf{y}$
Hebbian projection $\vec{H}_W^V(\mathbf{w}^j)$
$\mathbf{x}$
$\mathbf{w}^j$

(b) A large difference between the output view codebook activation and the Hebbian projection from the nearest neighbour in the input view codebook indicates that the input sample is incorrectly matched.

Figure **4.7:** Constructing a cross-view error signal within our network topology. As co-occurring samples arrive online in each data view, nearest neighbour prototypes are found and the output view codebook activation distribution (cf. Section 4.1.1.3) may be compared with the Hebbian projection (cf. Section 4.1.2.2) from the nearest-neighbour prototype in the input view codebook to the output view codebook. By measuring the distance between these two distributions, an error signal may be formed that indicates whether or not the input sample is being matched correctly. Figure 4.7(a) illustrates the case for correct matching, whereas Figure 4.7(b) illustrates the case for incorrect matching. Note that by exploiting the multi-view information, class labels are not required for the formation of such an error signal.

Although we have now shown how LVQ classifiers may be trained using probabilities or an error function of some kind instead of class labels, we have yet to specify what that error function would be within our learning framework. In approaching the specification of this error signal, a key observation at this point is to note that the co-occurrence Hebbian mapping as defined in Section 4.1.2.1 provides a temporal record of cross-view codebook activation. At a given time-step, given a prototype in the input view for example, it is possible to obtain a Hebbian projection that indicates how the output view codebook was activated on average by output samples over the training time leading up to that time-step as viewed from the perspective of the input view prototype which matched the co-occurring input samples. Thus, at any given point during training, given co-occurring samples, we may observe the activation of the output view codebook given the output sample (cf. Section 4.1.1.3), as well as the Hebbian projection from the prototype in the input view that best matches the input sample (cf. Section 4.1.2.2), and compare them. If these two activation distributions are quite similar, we might conclude that the selected input view prototype is indeed a good match for the input view sample in terms of potentially discriminating between output view meta-clusters, whereas if the two distributions are quite different, we might conclude that the prototype is a poor match for such purposes. This concept is perhaps best illustrated visually as in Figure 4.7. In the next section, we describe how the distance between the activation distributions may be measured.

4.3.3 Measuring Similarity Between Data Views

To measure the distance between two distributions, in particular the codebook activation distribution and cross-view Hebbian projection as discussed previously, we require a suitable distance metric, and in our case we employ the *Hellinger distance* for this purpose. Using the definition from [63], given any measurable space Ω, if f, g are densities of the measures μ, ν with respect to a dominating measure λ,

$$d_H(\mu,\nu) := \left[\int_\Omega (\sqrt{f} - \sqrt{g})^2 \, d\lambda\right]^{\frac{1}{2}} \tag{4.54}$$

which is independent of the choice of dominating measure λ. For a countable state space Ω,

$$d_H(\mu,\nu) := \left[\sum_{\omega\in\Omega} \left(\sqrt{\mu(\omega)} - \sqrt{\nu(\omega)}\right)^2\right]^{\frac{1}{2}} . \tag{4.55}$$

The Hellinger distance takes values in the bounded interval $[0, \sqrt{2}]$, making it amenable to statistical analysis, e.g. calculating mean distance. Given input view node k, we define

$$f_k(t) = \{h_{kl}(t) : \forall l \text{ in the output view layer}\} , \tag{4.56}$$

or all the Hebbian link weights that connect node k in the input view layer to the nodes in the output view layer at time t. We define

$$g(t) = \{a_l(t) : \forall l \text{ in the output view layer}\}, \tag{4.57}$$

or all the node activations in the output view layer at time t.

The discrete set f_k can be thought of as a distribution of the Hebbian map activity from node k in the input layer projected onto the output layer. Loosely put, this gives us a picture of what the output map looks like from the perspective of node k in the input map based on previous training experience. The set g, on the other hand, gives us a distribution of the output map activity with respect to the current training sample. Thus, when given a training sample for the input view, if we employ the metric $d_H(f_k(t), g(t))$, we can get an impression of how well its best matching unit node in the input view layer predicts the activity of the output view layer given its co-occurring training sample. This heuristic can of course be used in the opposite direction, from the output view layer to the input view layer, but for the algorithm we present in this paper it is employed strictly in the above way to augment the training of the input view layer. This is primarily because in the case of the object affordance learning problem as we defined it in the previous chapter, we wish to focus on clustering classes as well as possible in the output object effects data view while driving discriminative learning in the input object features data view. Now that we have the necessary tools in place, we may proceed to present our fully-fledged self-supervised multi-view algorithm.

4.3.4 Self-Supervised Online Multi-View Training Algorithm

To achieve self-supervision, we retain the training framework of Algorithm 1, but modify and replace Algorithm 2 to accommodate the learning rules described in Section 4.3.2. This self-supervised updater is described in Algorithm 5. Here we introduce the notion of learning in phases, the idea being that self-supervised learning should proceed developmentally with an initial phase of unsupervised learning as in Algorithm 1 in order to settle the prototypes with respect to the data distributions in each data view and to build a sufficiently detailed cross-view Hebbian co-occurrence mapping, before progressing to a subsequent phase of self-supervised learning using selected update rules from Section 4.3.2.

The algorithm may be broadly summarized as follows, where the steps in parentheses refer to steps in Algorithm 5:

1. Calculate the output view codebook activation distribution (Step 3).
2. Update the output view codebook (Step 4).
3. Calculate the input view codebook activation distribution (Step 8).
4. Calculate $\phi(\mathbf{w}^c)$ (Step 15), or $\phi(\mathbf{w}^i)\,\forall i$ if GLVQ-based (Step 19), as the Hellinger distance between the Hebbian projection from the input view nearest-neighbour $\mathbf{w}^c$ and the output view codebook activation distribution.
5. Update the input view codebook using either unsupervised learning (Step 12) or self-supervised learning (Step 16 or Step 22) depending on the learning phase.
6. Update the cross-view Hebbian co-occurrence mapping using the codebook activation distributions from each data view (Step 26).

The key to enabling the self-supervised learning updates lies in the fourth entry on the above list in the calculation of the ϕ probabilities which are required by the self-supervised update equations. Naturally, we assume here that both the Hebbian projection from the input view and the output view codebook activation distribution are normalized as probability distributions, and that the Hellinger distance is normalized between $[0, 1]$. We have found it convenient in our experiments to calculate a running mean of $\phi(\mathbf{w}^c)$ during training and set the ϵ threshold in the self-supervised updates to be that number, while setting the δ window value to be the running standard deviation of same. The running mean and standard deviation were calculated using an algorithm by Knuth [84] and are detailed in a different context in Section 5.2.2 of the following chapter. It should be noted that the self-supervised GLVQ-based update rules of Equations (4.51) and (4.52) are less efficient than their LVQ1-based counterparts of Equations (4.47) or (4.48), involving a complexity of $O(M_{\mathcal{W}})$ with respect to the calculation of the $\phi(\mathbf{w}^i)$ as opposed to $O(1)$ for the LVQ1-based updates, where $M_{\mathcal{W}}$ is the number of prototypes in the input view. Nevertheless, though we do not provide analytical proof of convergence here, the self-supervised GLVQ-based updates, being based on a provably convergent underlying fully-supervised algorithm, do appear to perform with slightly more stability over longer training periods, as shall be shown in the experimental results of Chapter 7.

Algorithm 5 Self-Supervised Multi-View Learner Online Update

```
Input: Input view codebook $\mathcal{W}$, output view codebook $\mathcal{V}$, Hebbian co-occurrence
    mapping $\mathcal{H}(\mathcal{W}, \mathcal{V})$, training samples $\mathbf{x}$ and $\mathbf{y}$.
Output: Updated input view codebook $\mathcal{W}'$, updated output view codebook $\mathcal{V}'$,
    updated Hebbian co-occurrence mapping $\mathcal{H}'(\mathcal{W}, \mathcal{V})$.
 1: Given $\mathbf{y}$, find output view nearest-neighbour (BMU) $\mathbf{v}_t^c$ and its neighbours
    $\mathcal{N}_t^{\mathbf{v}^c}$ using Equations (4.6) and (4.10).
 2: for $i = 1, \ldots, M_{\mathcal{V}}$ do
 3:   Get prototype activations $a_t(\mathbf{v}^i)$ using Equation (4.11), (4.12), or (4.13).
 4:   Update $\mathbf{v}_{t+1}^i$ using Equation (4.7).
 5: end for
 6: Given $\mathbf{x}$, find input view nearest-neighbour (BMU) $\mathbf{w}_t^c$ and its neighbours
    $\mathcal{N}_t^{\mathbf{w}^c}$ using Equations (4.6) and (4.10).
 7: for $i = 1, \ldots, M_{\mathcal{W}}$ do
 8:   Get prototype activations $a_t(\mathbf{w}^i)$ using Equation (4.11), (4.12), or (4.13).
 9: end for
10: if current learning phase is SOM-based then
11:   for $i = 1, \ldots, M_{\mathcal{W}}$ do
12:     Update $\mathbf{w}_{t+1}^i$ using Equation (4.7).
13:   end for
14: else if current learning phase is LVQ1-based then
15:   Calculate $\phi(\mathbf{w}^c) = d_H\left(\vec{H}_{\mathcal{W}}^{\mathcal{V}}(\mathbf{w}^c), A_{\mathcal{W}}(\mathbf{x})\right)$, the Hellinger distance between
      the Hebbian projection of $\mathbf{w}^c$ and the activation distribution for $\mathbf{x}$ in the
      output view codebook.
16:   Update $\mathbf{w}_{t+1}^c$ using Equation (4.47) or (4.48).
17: else if current learning phase is GLVQ-based then
18:   for $i = 1, \ldots, M_{\mathcal{W}}$ do
19:     Calculate $\phi(\mathbf{w}^i) = d_H\left(\vec{H}_{\mathcal{W}}^{\mathcal{V}}(\mathbf{w}^i), A_{\mathcal{W}}(\mathbf{x})\right)$.
20:   end for
21:   Find $\mathbf{w}_t^j$ and $\mathbf{w}_t^k$ using Equations (4.49) and (4.50).
22:   Update $\mathbf{w}_{t+1}^j$ and $\mathbf{w}_{t+1}^k$ using Equations (4.51) or (4.52).
23: end if
24: for $i = 1, \ldots, M_{\mathcal{W}}$ do
25:   for $j = 1, \ldots, M_{\mathcal{V}}$ do
26:     Update $\gamma_{t+1}^{\mathbf{w}^i\mathbf{v}^j} \in \mathcal{H}(\mathcal{W}, \mathcal{V})$ using Equation (4.21).
27:   end for
28: end for
```

4.4 Chapter Summary

In this chapter we proposed a self-supervised multi-view learning algorithm built on a theoretical foundation of unsupervised clustering, Hebbian learning and learning vector quantization. Codebook layers of prototypes are used to represent separate data views, or sensory modalities, and may be trained online using competitive learning in a process we term hyper-clustering. Class clusters used for cross-view classification may be formed in a data view through a process of meta-clustering the prototypes in that data view. The separate data views are connected together via a cross-view Hebbian mapping that itself may be trained online using the co-occurrence of data between the separate layers. This cross-view mapping is used to project percepts from one data view onto another through a process known as Hebbian projection that allows us to measure how the data in each view correspond to each other. We have shown how, given the multi-view structure of the codebook layers of prototypes, we may derive learning rules based on both Hebbian projection and the learning vector quantization paradigm that can employ class probabilities instead of actual class labels during training, thus allowing us to bootstrap the self-supervised learning process in an online manner even when the classes are not yet fully known. This is an important consideration at the lower-level of a cognitive system like an autonomous robot where lower-order feature data co-occur in an online manner across multiple data views and higher-level concepts ought to be formed dynamically [166].

It is important to re-emphasize that class labels are not employed in the self-supervised training procedure. Such labels are generated dynamically at classification time during calls to Algorithm 4. There is an interesting observation to be made here with regard to such dynamic cross-view prototype labelling within the context of online learning. If we are to assume the hard online developmental learning constraints of Table 3.2 in Chapter 3, where learning commences from a starting state without access to a batch of training data previously observed or otherwise, then fully-supervised learners, at least of the LVQ variety, must arbitrarily label prototypes which themselves may be randomly initialized. A consequence of this is that it may take many update iterations, or perhaps many epochs of training, for the labelled prototypes to move to appropriate positions within the input space before being able to correctly classify samples. As we shall demonstrate in the experimental results of Chapter 7, this issue can actually be circumvented by the dynamic labelling of the multi-view self-supervised learning described in this chapter. Prototypes that are already appropriately positioned in the input view with respect to certain modes of the distribution may be dynamically assigned class labels that are most suited to them by the driving output view.

Though the self-supervised algorithm presented in this chapter is self-contained and operates as expected, as shall be seen in the experimental results of Chapter 7, its performance, particularly in the case of short training periods, can be boosted significantly by considering the relevance of individual feature dimensions. Thus, in the following chapter, we present two novel algorithms designed for that purpose.

Chapter 5

Feature Relevance Determination

In the previous chapter learning proceeded by moving prototypes around to cluster within data views, labelling the prototypes using dynamically formed classes and culling prototypes according to their relevance in terms of predictive accuracy. Something that was not considered however, is how relevant the individual feature dimensions are in supervised class prediction and how such relevancies might be determined. A standard approach to this issue is to pre-process the data using some form of feature selection or dimensionality reduction, but this might not be be feasible in many scenarios, particularly when hard online learning constraints such as those detailed in Chapter 3 are considered. In the case of learning vector quantization, feature relevance determination can be performed in an online manner, and in this chapter we review some methods for achieving that. Two new algorithms for LVQ-based relevance determination are also presented. Both methods exploit the positioning of the prototype vectors in the input feature space to inform estimates of the Fisher criterion score along the input dimensions, which are then used to form online estimates of the relevance of the input dimensions with respect to the classifier output. Both methods provide online updates that may be used alongside regular LVQ updates and neither method requires the specification of a learning rate, as in stochastic gradient descent. At the end of this chapter, we also describe how the proposed methods may be applied to the multi-view self-supervised learner of the previous chapter, both during training and at classification time.

5.1 Relevance Determination for LVQ

Much attention has been paid in recent years to the role that the distance metric plays in the effectiveness of LVQ methods. LVQ ordinarily relies on the Euclidean metric to measure the distance between data points, which provides equal weighting to all input dimensions. Many of the input dimensions, however, may have little relevance when considering the desired output function and may even have a detrimental effect on the output if considered with equal weighting in the metric to the more important dimensions. Various reformulations of LVQ have been proposed that can adjust the metric during training such that the impact of the individual input dimensions are dynamically re-weighted during training in accordance with the data under consideration. This can make a crucial difference, both during training for more efficient adjustment of the prototypes, and when classifying test samples where the undue consideration of irrelevant dimensions can mean the difference between a correct and incorrect classification.

One early adaptation of LVQ3 known as *distinction sensitive learning vector quantization (DSLVQ)* [138] achieves this by using a heuristic to adjust weights along each of the input dimensions to modify the Euclidean metric. An adaptation of LVQ1 known as *relevance learning vector quantization (RLVQ)* [8] uses Hebbian learning to do similar, by adjusting weights for each of the input dimensions at every training step depending on the degree to which each dimension contributed to the correct or incorrect classification of a training sample. RLVQ was subsequently adapted for use with GLVQ producing a method known as *generalized relevance learning vector quantization (GRLVQ)* [69] such that the feature dimension weight updates also adhere to gradient descent dynamics in a similar way to the prototype updates. Another modified version of GLVQ [179] uses Fisher's discriminant analysis to create an alternative metric to the weighted Euclidean distance that employs a matrix transformation to reduce the feature space dimensionality. More recently, an adaptive metric was used in combination with training data selection for LVQ [134].

The following two sub-sections review both RLVQ and GRLVQ, perhaps two of the more well-known feature relevance LVQ-based methods, in more detail. The new algorithms are described in Section 5.2 and experimental results comparing our algorithms to both RLVQ and GRLVQ are provided in Chapter 7.

5.1.1 RLVQ

By allowing the λ_i weights in Equation (4.3) to be altered, the LVQ prototype update equations can be accompanied by updates that also alter the metric weights dynamically during training, hence allowing for an adaptive Euclidean metric. In RLVQ [8], LVQ1 training is adjusted such that the following weighting factor update rule is applied alongside Equation (4.41):

$$\lambda_l := \begin{cases} \lambda_l - \beta(x_l - w_l^j)^2 & \text{if } c^j = y \\ \lambda_l + \beta(x_l - w_l^j)^2 & \text{otherwise,} \end{cases} \tag{5.1}$$

for each l-th dimension where $\beta \in (0, 1)$ is a learning rate for the weighting factor adjustments. The weights are normalized at each update such that $\sum_{i=1}^{n} \lambda_i = 1$. The motivation for the above update is derived from Hebbian learning, the idea being that when $\mathbf{w}^j$ classifies the sample $\mathbf{x}$ correctly, the weights for the dimensions that contributed to the classification the most are increased, whereas the weights of the dimensions that contributed the least are decreased. When $\mathbf{w}^j$ incorrectly classifies $\mathbf{x}$, the weights for dimensions that contributed most are decreased, whereas the weights for dimensions that contributed the least are increased.

5.1.2 GRLVQ

GRLVQ is an application of the above idea to GLVQ, such that the updates for the weights for the metric also follow a stochastic gradient descent on the error function defined by GLVQ. The prototype updates in Equation (4.44) and Equation (4.45) are thus accompanied by the following metric weight update [69]:

$$\lambda_l := \lambda_l - \beta f' \left(\frac{d_k}{(d_j + d_k)^2} \left(x_l - w_l^j\right)^2 - \right. \tag{5.2}$$

$$\left. \frac{d_j}{(d_j + d_k)^2} \left(x_l - w_l^k\right)^2 \right), \tag{5.3}$$

for each l-th dimension, where β is once again the learning rate, and the weights are once again normalized after each update.

One disadvantage of both RLVQ and GRLVQ is that they require the specification of an additional learning rate, β, which can be difficult to specify appropriately with respect to its α counterpart in the prototype updates. Another disadvantage is that they fail to take into consideration the additional statistical information provided by the remaining prototypes other than the ones currently being updated at a given training step when making relevance estimates. These issues are addressed with the following two proposed LVQ relevance determination algorithms.

5.2 Relevance Determination for LVQ using Fisher Criterion Score

The Fisher criterion, while ordinarily associated with Fisher's discriminant analysis [54], can also serve as an effective means for feature relevance determination when applied across individual data dimensions. Letting $\overline{x}^A = \frac{1}{N}\sum_{x^i \in A} x^i$ be the mean of a set of points A with cardinality N, the Fisher criterion score for a given individual dimension l is defined as

$$F(l) = \frac{S_B(l)}{S_W(l)}, \tag{5.4}$$

where

$$S_B(l) = \sum_{c=1}^{C} N^c \left(\overline{x}_l^{X^c} - \overline{x}_l^{X}\right)^2 \tag{5.5}$$

is the between-class variance and

$$S_W(l) = \sum_{c=1}^{C} \sum_{\mathbf{x} \in X^c} \left(x_l - \overline{x}_l^{X^c}\right) \tag{5.6}$$

is the within-class variance over the l-th dimension.

With regard to relevance determination for LVQ, $F(l)$ could be calculated for each dimension over the entire training set X in advance of LVQ training and applied to the weighting factors in Equation (4.3) by setting $\lambda_l = F(l)$ for all l to form a weighted metric. However, for many applications it is more desirable to have an online feature relevance training mechanism that is not reliant on having access to the entire training set at once. In the following, two such online algorithms are presented where estimation of the Fisher criterion score is integrated into the training scheme for LVQ.

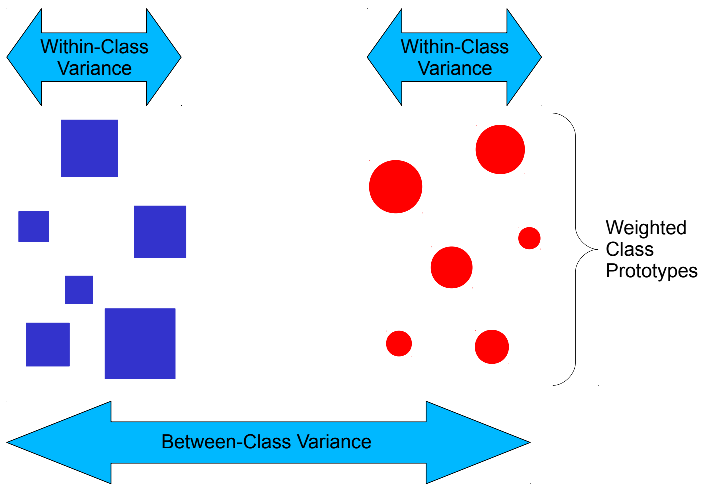


Figure **5.1:** Visualisation of the first proposed feature relevance determination technique. The Fisher criterion score is calculated over weighted class prototypes as opposed to actual data.

5.2.1 First Proposed Algorithm: FC1

With the first algorithm, rather than calculating $F(l)$ over the data in X, at a given time-step t the score is estimated over the values of the prototype vectors in $\mathcal{W}$. This is plausible since the distribution of the prototype vectors should approximate the distribution of the data over time. During training, certain prototypes will quantize more significant modes of the distribution than others, thus to account for this, weighted means and variances are calculated for each class based on the classification accuracy of each of the prototypes of that class, then the Fisher criterion score is calculated over the weighted means and variances for all classes. Firstly, the definition of $\mathcal{W}$ is altered to

$$\mathcal{W} = \{\ (\mathbf{w}^i, c^i, p^i) \in \mathbb{R}^n \times \{1, \ldots, C\} \times \mathbb{R} \tag{5.7}$$

$$|\, i = 1, \ldots, M\ \} \tag{5.8}$$

where, given random variable $(\mathbf{x}, y)$, $p^i = p(\mathbf{x} \in R^i | y = c^i)$ is the conditional probability of x lying in receptive field R^i of prototype $\mathbf{w}^i$ given that $\mathbf{w}^i$ correctly classifies x. The p^i form probability distributions over class prototypes such that $\sum_{p^i \in \mathcal{W}^c} p^i = 1$ for each class c. A definition of the estimated Fisher criterion score may now be formed as

$$F(l) \simeq \hat{F}(l) = \frac{\hat{S}_B(l)}{\hat{S}_W(l)}, \tag{5.9}$$

where

$$S_B(l) \simeq \hat{S}_B(l) = \sum_{c=1}^{C} \frac{N^c}{N} \left(\hat{w}_l^{\mathcal{W}^c} - \hat{w}_l^{\mathcal{W}} \right)^2 \tag{5.10}$$

is the estimated between-class variance over the l-th dimension,

$$S_W(l) \simeq \hat{S}_W(l) = \sum_{c=1}^{C} \frac{N^c}{N} \sum_{(\mathbf{w}^i, c^i, p^i) \in \mathcal{W}^c} p^i \left(w_l - \hat{w}_l^{\mathcal{W}^c} \right) \tag{5.11}$$

is the estimated within-class variance over the l-th dimension, and

$$\hat{w}_l^{\mathcal{W}^c} = \sum_{(\mathbf{w}^i, c^i, p^i) \in \mathcal{W}^c} p^i w_l^i \tag{5.12}$$

is a weighted mean over the l-th dimension of prototypes in a given set $\mathcal{W}^c \subseteq \mathcal{W}$.

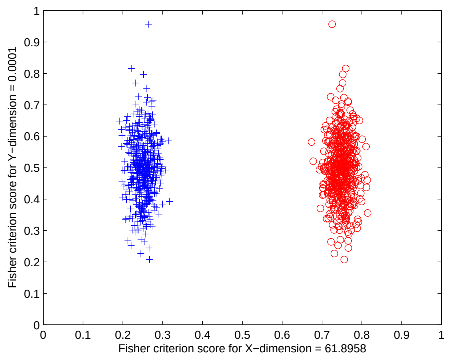
The λ_m relevance factors may then be updated at each time-step by taking a running mean of the normalized estimated Fisher criterion score:

$$\lambda_{l,t+1} := \lambda_{l,t} + \frac{\frac{\hat{F}(l)}{\sum_{l=1}^{n} \hat{F}(l)} - \lambda_{l,t}}{t+1}. \tag{5.13}$$

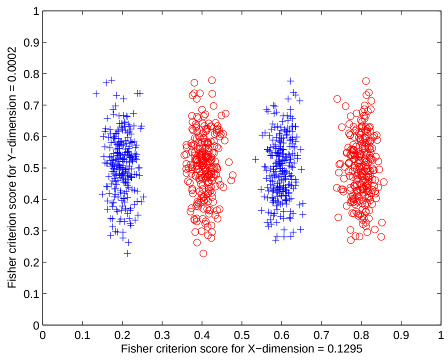
While the Fisher criterion score is suitable for feature relevance determination in many cases, its main drawback is that it does not cope well with multi-modal feature distributions. An example of this is shown in Figure 5.2. This problem remains in the estimation proposed above, since Equation (5.10) and Equation (5.11) are calculated over all class prototypes. The second proposed algorithm was designed to account for this issue.

5.2.2 Second Proposed Algorithm: FC2

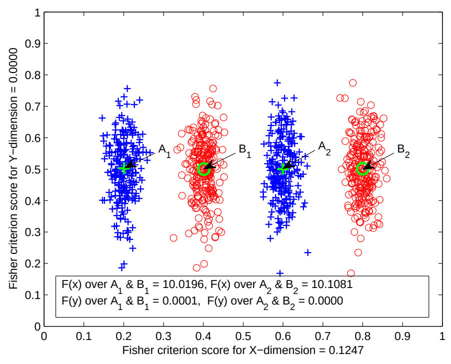
The second proposed algorithm is based on the idea of calculating the Fisher criterion score between single prototype vectors of opposing classes, where the assumption is made that each class prototype vector may be quantizing different modes of the underlying class distribution. During training, Gaussian kernels are used to maintain estimates of the accuracies of each of the prototypes over the parts the data distribution accounted for by each of their receptive fields. At a given training step, the nearest single prototypes of each class to the training



(a)



(b)



(c)

Figure **5.2:** A simple 2-dimensional, 2-class example of how the Fisher criterion score (see Equation (5.4)) can fail as a feature relevance metric over multi-modal data distributions. (a) shows uni-modal class data distributions, linearly separable in the x-dimension, but with large overlap in the y-dimension. The score reflects the relevance of each dimension to class discrimination ($F(x) \simeq 61.9, F(y) \simeq 0$). (b) by comparison, shows the same number of data points, but with a multi-modal distribution (yet still linearly separable in x). The score is significantly lower for the x-dimension in this case ($F(x) \simeq 0.1, F(y) \simeq 0$). (c) shows the improvement provided by calculating the score between pairs of clusters with centres at points A_1, B_1, A_2 and B_2, ($F(x) \simeq 10$ over A_1 & B_1, $F(x) \simeq 10$ over A_2 & B_2). See Section 5.2 for more details.

sample are found, and their Gaussian kernels are used to calculate an estimate of the Fisher criterion score for that local portion of the distribution, which is subsequently averaged over the entire training period.

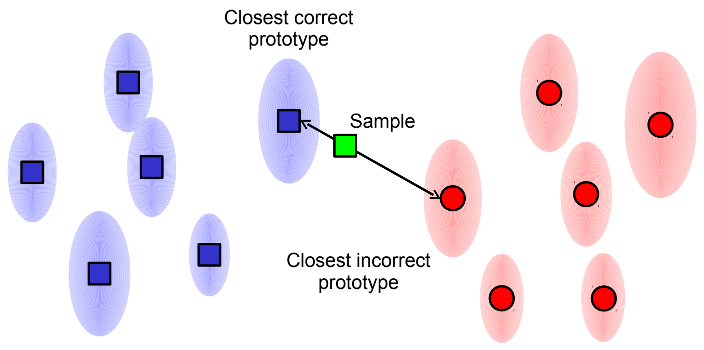


Figure **5.3:** Visualisation of the second proposed feature relevance determination technique. Prototypes have associated Gaussian kernels used to maintain estimates of their individual classification accuracies. At a given training step, the Fisher criterion score is calculated locally between the closest prototypes of opposing classes to the current training sample. A running average of this locally calculated score is maintained as training progresses. This technique helps counter the problem illustrated in Figure 5.2.

The definition of $\mathcal{W}$ is this time altered to accommodate a Gaussian estimate of the accurate portion of the receptive field for each prototype, such that

$$\mathcal{W} = \{ \ \left(\mathbf{w}^i, c^i, \mathcal{N}(\mathbf{x}; \mu^i, \Sigma^i)\right) \in \mathbb{R}^n \times \{1, \dots, C\} \times \tag{5.14}$$

$$(\mathbb{R}^n \times \mathbb{R}^{n \times n}) \mid i = 1, \dots, M \ \}, \tag{5.15}$$

where $\mathcal{N}$ approximates $\tilde{R}^i = \{\mathbf{x} \in R^i | y = c\}$ with mean μ^i and covariance matrix $\Sigma^i = \mathrm{diag}([s^i_1, \dots, s^i_n])$ where the $\{s^i_l\}^n_{l=1}$ are variances along each l-th dimension. During LVQ training, given a random sample $(\mathbf{x}, y) \in X$ at training step t, if the closest prototype $\mathbf{w}^j$ classifies x correctly, i.e. $c^j = y$, then μ^j_l and s^j_l are updated in each l-th dimension as follows [84]:

$$\mu^j_{l,t} := \mu^j_{l,t-1} + \frac{x_l - \mu^j_{l,t-1}}{t} \tag{5.16}$$

$$\hat{s}^j_{l,t} := \hat{s}^j_{l,t-1} + (x_l - \mu^j_{l,t-1})(x_l - \mu^j_{l,t}) \tag{5.17}$$

where $\mu^j_{l,t}$ is the running mean estimate and $s^j_{l,t} = \frac{\hat{s}^j_{l,t}}{t-1}$ is the running variance estimate for the l-th dimension at training step t. If $c^j \neq y$, then the above updates are not performed. Assuming a sufficient number of updates have been performed on the relevant prototypes up until step t, a Fisher criterion score estimate may be calculated between

$$\mathcal{W}' = \{\, \omega^k = \left(\mathbf{w}^k, c^k, \mathcal{N}(\mathbf{x}; \mu^k, \Sigma^k)\right) \in \mathcal{W} \tag{5.18}$$

$$\mid \forall \mathbf{w}^i, c^i = c^k, d(\mathbf{x}, \mathbf{w}^k) \leq d(\mathbf{x}, \mathbf{w}^i) \,\}\,, \tag{5.19}$$

the closest prototypes of different classes (including $\mathbf{w}^j$), as follows:

$$F(l) \simeq \tilde{F}(l) = \frac{\tilde{S}_B(l)}{\tilde{S}_W(l)}, \tag{5.20}$$

where

$$S_B(l) \simeq \tilde{S}_B(l) = \frac{1}{C}\sum_{c=1}^{C} (\mu^c_l - \overline{\mu}_l)^2 \tag{5.21}$$

is the between-class variance estimate in the l-th dimension with

$$\overline{\mu}_l = \frac{1}{C}\sum_{\omega^k \in \mathcal{W}'} \mu^k_l, \tag{5.22}$$

and

$$S_W(l) \simeq \tilde{S}_W(l) = \sum_{\omega^k \in \mathcal{W}'} s^k_l \tag{5.23}$$

is the within-class variance estimate in the l-th dimension. The relevance factors may then be updated in a similar way to Equation (5.13), this time using the new estimates:

$$\lambda_{l,t+1} := \lambda_{l,t} + \frac{\frac{\tilde{F}(l)}{\sum_{l=1}^{n}\tilde{F}(l)} - \lambda_{l,t}}{t+1}. \tag{5.24}$$

Since each prototype carries an accompanying Gaussian kernel that estimates its accuracy, it is now possible to estimate the Fisher criterion score using only single prototypes from each class, as opposed to the previous algorithm where multiple prototypes in each class have to be considered to achieve variance estimates. Though the model is made slightly more complex, it is more capable of successfully handling the multi-modal distribution issue described in Fig. 5.2 as shown by the experimental results in the Chapter 7.

5.3 Applying Relevance Determination to the Self-Supervised Multi-View Learner

5.3.1 During Training

When employing fully-supervised learning vector quantization, in the case of both the RLVQ and GRLVQ algorithms, the feature weights provided by Equations (5.1) and (5.3) respectively are generally both updated and applied during training. Thus, at any given training step the adaptive Euclidean metric of Equation (4.3) may have different λ weights in each feature dimension, meaning that the prototype updates of Equation (4.41) and Equations (4.44) and (4.45) can potentially alter prototype weights to varying degrees in each dimension during training. Both RLVQ and GRLVQ feature relevance updates can also easily be applied to the self-supervised learner of Algorithm 5, requiring only knowledge of the nearest-neighbour prototype to the training sample, or in the case of GRLVQ, knowledge of the nearest correct prototype and the nearest incorrect prototype.

Only one of our proposed feature relevance algorithms, FC2, is amenable to being applied during the training process in this way, at least in the case of the self-supervised learner. FC1 cannot easily be applied to the self-supervised learner during training since it requires the prototypes to be labelled. The reason why FC2 can be applied during training of the self-supervised learner is because it does not necessarily require prototype labels. It simply needs to know the identity of the closest correct prototype to the training sample as well as that of the closest incorrect prototype. Since steps 18-21 of Algorithm 5 provide that information using the ϕ-probabilities given by the Hellinger distance calculations, this knowledge is available during training and thus Equation (5.24) can be applied concurrently.

5.3.2 At Classification Time

Once an instance of the self-supervised learner has been trained using Algorithms 1 and 5 and classification Algorithm 4 is called upon to classify test samples, the prognosis changes for FC1 because, since the prototype labels in the input view are provided by a combination of meta-clustering in the output view and Hebbian projection, Equation (5.9) may now be applied directly over the prototypes, though without the temporal averaging step of Equation (5.13) that is performed during training in the fully-supervised case. We may also technically forgo the application of adaptive feature weighting during training in the cases of RLVQ, GRLVQ and FC2, choosing instead to update the weights during training and waiting until classification time to actually apply them in the metric when classifying test samples. Interestingly, of the above methods, FC2 is the only one that can be applied directly at classification time without needing any additional information provided by the training procedure, thus it can potentially serve to augment the performance of the other methods, even if they have been applied adaptively during training. In the experimental results of Section 7.2 in Chapter 7 we demonstrate how this can indeed sometimes be the case in practice.

5.3.2.1 In the Output View

The obvious application of our Fisher-based feature relevance approach in the multi-view setting lies in the input space where some form of cross-view class information is available. However, in the experiments described in Sections 7.3.3.2, 7.3.3.3, and 7.4 of the experimental results chapter, we also made use of it in the output space to augment the class discovery process. In the output space, even though the class cluster structure is initially unknown, it is still possible to select features that are more likely to be relevant to forming good cluster hypotheses. Firstly, given i, we split the i-th dimension of V into multiple histogram partitions $Q_1^V, \ldots, Q_k^V, \ldots$, where each $Q_k^V = \{B_1^{V_i}, \ldots, B_k^{V_i}\}$ subdivides V along the range of i into k evenly-sized bins containing V-prototypes. Then using Q_k^V in place of the class labels in (5.9), (5.10), (5.11) and (5.12), and calculating (5.9) for a range of k values (we used $k = 2, 4, 6, 8, 10$ in our experiments), averaging over k, and assigning a λ_i weight to the k-averaged value, we selected features over a certain threshold (we used the λ_i-average in our experiments) that were likely to contribute most to good clustering over a range of k-values. We then used these features as input for the clustering algorithm described in Section 4.2.2.1.

5.4 Chapter Summary

In conclusion, two new relevance determination algorithms have been proposed for LVQ that exploit the positioning of prototypes in the input feature space to calculate Fisher criterion score estimates in the input dimensions for an adaptive metric. An advantage provided by these methods over other metric-adaptive LVQ methods based on gradient descent, is that they do not require a learning rate or other parameters to be specified. Moreover, they provide incremental update rules that operate alongside regular LVQ update rules and can therefore be applied to any algorithms based on the general LVQ paradigm. Towards the end of the chapter, we also showed how the new methods may be applied to the multi-view self-supervised learner of Chapter 4 both during training and at classification time, depending on the method involved. In Section 7.1 of Chapter 7, experimental evaluations for the fully-supervised case are provided under various stress conditions and over various datasets and the proposed methods are shown to perform competitively against various other LVQ-based methods, and against SVM. As well as that, in Sections 7.2, 7.3 and 7.4 of Chapter 7, we show results for various combinations of the application of these techniques both during training and at classification time in the case of self-supervised learning.

Chapter 6

Robot and Vision Systems

Researching and developing real-world systems for affordance learning is difficult because, given that both perception and action are involved, a number of different, but necessary sub-systems need to be accounted for. On the perception side, such systems must possess sufficiently robust vision or sensory modules of some kind so that they can “see” in their environment and recognize invariant features and salient visual events. Of course, such affordance learning systems should also be able to interact with their environments using actuators. This can involve anything from relatively simple mobile robot actuation to complex kinematics and motion planning. One of the main tasks in our work for this thesis was the design of an experimental platform that married sufficient functionality in each of these areas, so as to facilitate exploratory learning of the basic affordances of objects, using a robotic arm and various camera systems.[1] In addition, further experiments were performed using a secondary platform involving a different robot arm and an RGB-D depth sensor.[2] We devote the first part of this chapter to describing each of these experimental platforms, and the second part to describing the visual feature extraction methods used in each of them.

[1]This work was performed at the Visual and Cognitive Systems Lab, Faculty of Computer and Information Science, University of Ljubljana.

[2]This work was performed at the Humanoid and Cognitive Robotics Lab, Department of Automation, Biocybernetics and Robotics, Jožef Stefan Institute.

6.1 Experimental Platforms

6.1.1 Katana/Camera Setup

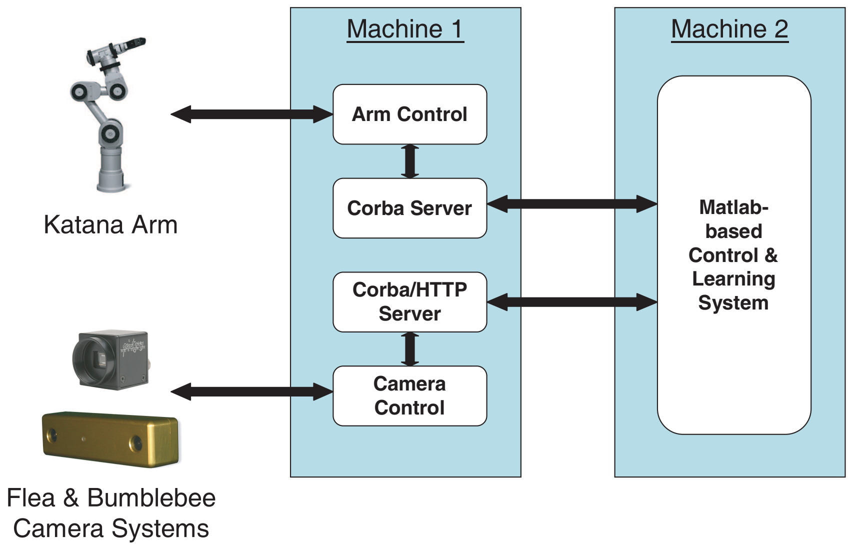


Figure **6.1:** Katana/Camera setup system architecture.

In terms of hardware, this experimental setup primarily consisted of both monocular and stereo cameras for perception and a robotic arm for action. The system that we developed was designed to be controlled from the MATLAB® software environment. MATLAB was chosen as it allows for rapid prototyping of high-level control programs and provides extensive functionality for computer vision manipulations as well as other procedures. In order for experiments involving the robot arm to be performed via MATLAB and to aid cross-platform integration, a CORBA (Common Object Request Broker Architecture) interface was developed to sit between the low-level arm control software and high-level Java control client which could easily be called from MATLAB. This allowed for swift arm/work-space calibration from within MATLAB and allowed for simple $moveTo(x, y, z)$ functionality for moving the end-effector to a localised position (x, y, z) in the workspace. Similar functionality was set up to allow capture of images, 3-D point clouds, and video from the various camera systems. See Figure 6.1 for an overview of the system architecture.

6.1.1.1 Action: 5-DOF Robotic Arm with 2-Finger Gripper

In this setup, we used a Neuronics Katana 6M robotic arm which features 5 DC motors for main arm movement, as well as a 6th motor to power a 2 fingered gripper that houses both infrared and haptic sensors. See Figure 6.2 for various views of the arm in our learning environment. The arm control software that was used for this work is a modified version of Golem[3] [88], control software for the Katana arm. Given desirable parameters, Golem uses forward kinematics to generate arm joint orientations and motion paths, then uses cost functions and searches to select the ones that most closely fit the parameters.

In order to ensure that the actions, and by extension the object affordances, that were available to the system were as consistent and learnable as possible, we restricted the motions of the arm in the following two ways:

- Firstly, we constrained the arm's path-finder software so that the orientation of the end-effector was as orthogonal to the work surface as possible. This ensured that the angle of contact between the end effector and objects remained consistent over repeated actions. While this may appear to be an exercise in 'freezing' out degrees-of-freedom, the resultant arm movements used to achieve these orthogonal orientations remained complex and all arm joints tended to be used during movement.

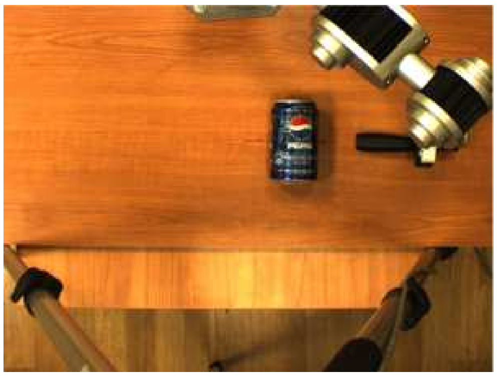
- Secondly, we tried to achieve a linear end-effector motion trajectory when moving between workspace positions. Because the path-planner uses randomly generated motion paths, if we did not place such a constraint on the search, the movement paths of repeated actions that move the end effector from position (x_1, y_1, z_1) to position (x_2, y_2, z_2) would be arbitrary.

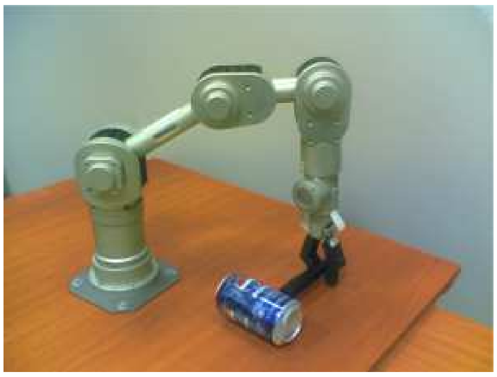
6.1.1.2 Perception: Monocular and Stereo Camera Systems

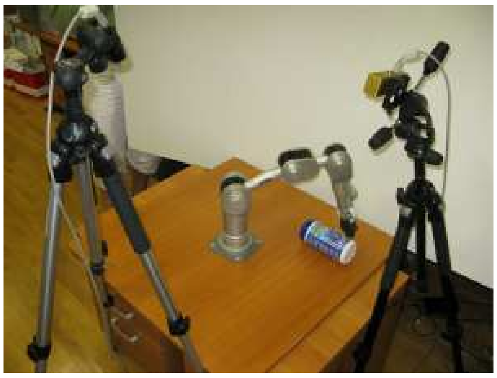
Two Point Gray Research cameras- the Flea® monocular camera (640x480 @ 60FPS or 1024x768 @ 30FPS) and the Bumblebee® 2 grayscale stereo camera (640x480 at 48FPS or 1024x768 at 20FPS) were used to gather intensity images, 3-D point clouds from range data and video. The camera systems were operated

[3]Golem was developed by Marek Kopicki at the University of Birmingham who kindly provided us with a copy for our research.

using a similar interface to that of the robotic arm. A Java client was called from MATLAB to interface with a CORBA server that implemented the low-level camera functionality. During experiments, after an action command was issued to the robotic arm, the camera system started recording images and continued recording until movement in the scene has ceased. These images were then used to create a video, which was passed to a compression module, after which it could be gathered from a web server.


(a) Top-down view of arm holding a pushing tool.


(b) Side view of arm holding a pushing tool.


(c) Side view of two-camera setup.

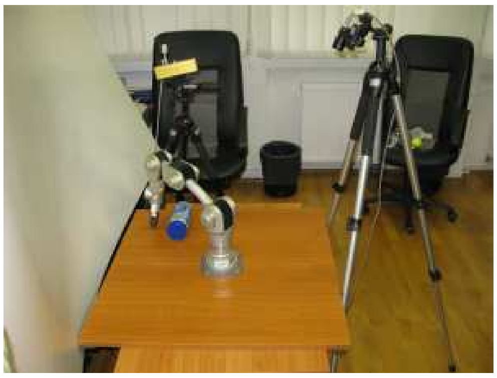

(d) Back view of two-camera setup.

Figure **6.2:** Learning environment for the Katana/Camera setup.

6.1.1.3 Learning Environment

In the learning environment for this setup, the Katana arm was mounted on a flat table with a wooden laminate surface, whilst the camera systems were mounted on tripods in fixed positions facing the work surface. Lighting conditions were

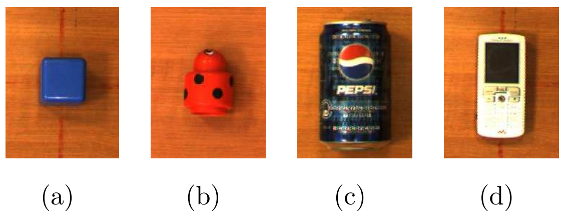


Figure **6.3:** Sample rolling versus non-rolling objects as seen by the Flea colour camera: (a) blue toy cube (non-rolling) (b) ladybird rattle (rolling) (c) cola can (rolling) (d) mobile phone (non-rolling).

controlled to facilitate various techniques for segmenting the object from the scene, including background subtraction and graph-cut segmentation. Household objects (See Figures 6.3 and 7.15 for examples) were placed on the work surface so that the arm may interact with them. Figure 6.2 shows various views of the learning environment.

6.1.2 KUKA-LWR/Kinect Setup

In the second setup, as depicted in Figure 6.4, a 7-DOF KUKA-LWR® arm as well as a 3-fingered BarrettHand® were used for object push interactions, while a Microsoft Kinect® RGB-D sensor was used to gather 3-D point clouds of the scene. The Point Cloud Library (PCL)[4] was used to extract and manipulate object clouds from the resulting data.

6.1.2.1 Learning Environment

As can be seen in Figure 6.4, the environment for this setup was rather similar to that of the Katana/Camera setup, apart from the inclusion of small hubs around the edges of the table to prevent the ball objects from rolling off of the table.

[4] http://pointclouds.org/

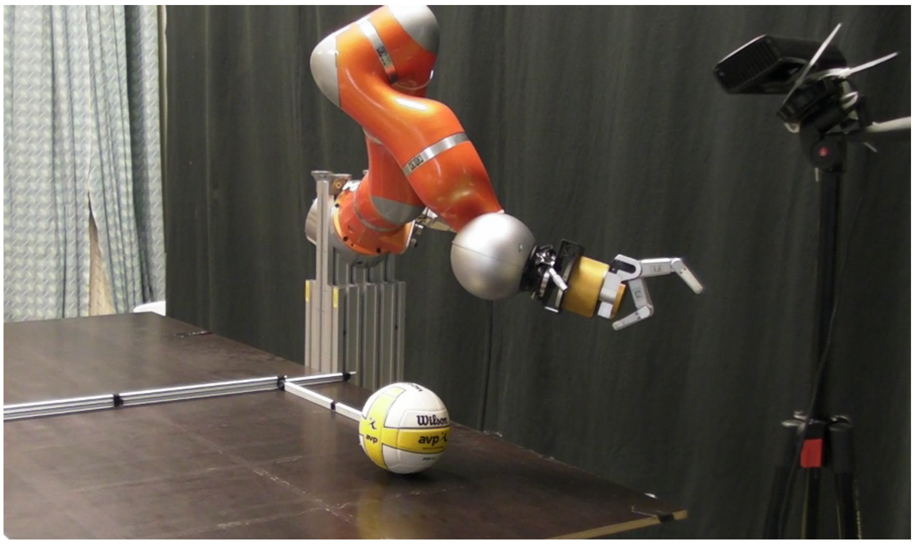


Figure **6.4:** A second experimental setup and learning environment using a KUKA-LWR robot arm, a BarrettHand, and a Microsoft Kinect RGB-D sensor.

6.2 Visual Feature Extraction

Recall from the affordance learning formalisation discussed in Chapters 2 and 3, we aim to represent object affordances as tuples of the form $< O, A, E >$ where O, A and E are feature vectors describing the object features, the action performed on the object and the resultant effects respectively. In our particular affordance learning scenarios we used visual features derived from the camera outputs as a basis for vectors O and E. With regard to the O vectors, we required visual features that described the intrinsic properties of objects, such as object shape, and with the E vectors, we required visual features that captured the effects that objects undergo when actions are performed on them, such as motion when pushed. Different sets of features were extracted for our experiments in various different ways depending on the setup.

For instance, in the case of the Katana/Camera setup, in order to simplify the process by which these sets of features were gathered, we extracted them in two separate stages. When objects were placed in the scene, images were taken both from the Bumblebee stereo camera and the Flea monocular camera prior to interaction. The Bumblebee stereo camera produces both regular intensity images and range data (3-D point clouds) accompanied a point correspondence mapping between them. Once such data was acquired, objects could be segmented from the scene and object features could be extracted as depicted in the feature extraction pipeline described in Section 6.2.1 and depicted in Figure 6.5. When actions were performed on the objects using the arm, video was recorded of the objects in motion and processed for effect features. This process is described in Section 6.2.2.

In the following, referring to each of the experimental setups as necessary we discuss how object features were extracted in Section 6.2.1 and how effect features were extracted in Section 6.2.2.

6.2.1 Object Features

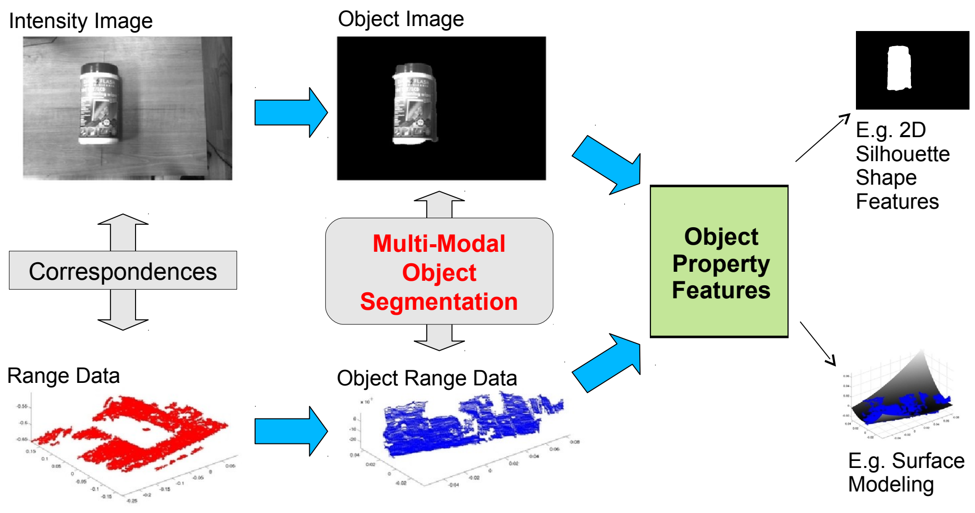


Figure **6.5:** The object feature extraction pipeline in the Katana/Camera setup.

With regard to the object features, for the purposes of our particular affordance learning scenario, we were primarily interested in extracting features that describe the global shape of an object as they were likely to be more relevant for determining how the object would behave when pushed than the types of local invariant visual features used predominantly in object recognition scenarios. However, in theory, any types of features that describe properties of the objects under consideration could be used here, as evidenced by the diversity of approaches taken in the literature. For example, Fritz [59, 60] used SIFT features to distinguish between liftable and non-liftable objects. Object shape features from 3-D vision have been used in prior work on push affordance learning [90, 144, 187, 188], and they form one of the mainstays of our approach here.

In order to extract such global shape features, we required a means of segmenting the objects, both from the image data and from the 3-D point clouds derived from the range data produced by the stereo cameras.

6.2.1.1 Object Detection and Segmentation

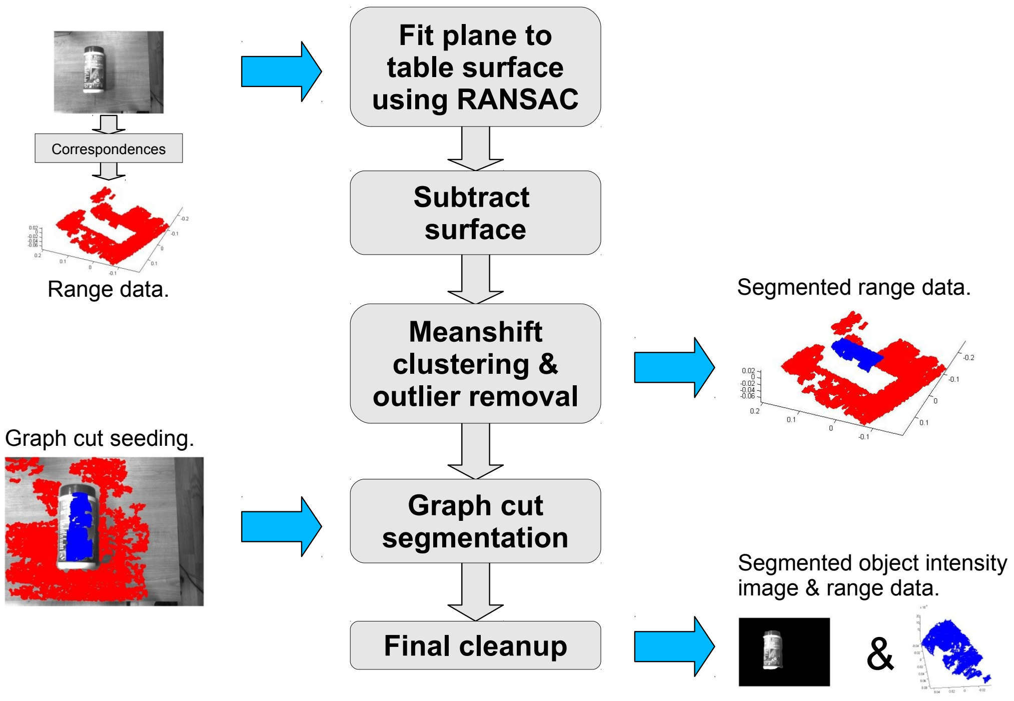

Correspondences
Range data.
Fit plane to table surface using RANSAC
Subtract surface
Meanshift clustering & outlier removal
Graph cut segmentation
Final cleanup
Segmented range data.
Graph cut seeding.
Segmented object intensity image & range data.
&

Figure **6.6:** The multi-modal object segmentation pipeline in the Katana/Camera setup.

We developed an algorithm for reliably segmenting the objects multi-modally from both regular images and their corresponding 3-D point clouds. The main idea behind the algorithm leverages a notable feature of our affordance learning environment; that is that the objects always lie on a flat table surface. This fact

Algorithm 6 Multi-Modal Object Segmentation

```
Input: Image $X_I \subset \mathbb{R}^1$ or $X_I \subset \mathbb{R}^3$, Range data $X_R \subset \mathbb{R}^3$, Correspondences
    $X_C \subseteq X_I \times X_R$.
Output: Foreground image $Y_{FI} \subset X_I$, Foreground range data $Y_{FR} \subset X_R$, Correspondences $Y_{FC} \subseteq Y_{FI} \times Y_{FR}$, Background image $Y_{BI}$, Background range data
    $Y_{BR}$, Correspondences $Y_{BC} \subseteq Y_{BI} \times Y_{BR}$.
1: Fit plane $P$ to table surface in $X_R$ using RANSAC: $Y_{BR} = \mathrm{ransac}(X_R, P)$,
   leaving background planar points $Y_{BR}$.
2: Subtract background planar points from $X_R$: $X_{F_1R} = X_R / Y_{BR}$.
3: Perform mean-shift clustering on $X_{F_1R}$ using a large bandwidth $\theta$ to find the
   largest cluster, discarding outliers: $X_{F_2R} = \mathrm{meanshift}(X_{F_1R}, \theta)$.
4: Letting $S_{FI} = \{x_i \,|\, (x_i, x_r) \in (X_I \times X_{F_2R}) \cap X_C \,\forall i\}$ be foreground seed points
   and $S_{BI} = \{x_i \,|\, (x_i, x_r) \in (X_I \times X_{BR}) \cap X_C \,\forall i\}$ be background seed points,
   we perform a graph-cut segmentation of the image $X_I$: $(Y_{FI}, Y_{BI}) =$
   $\mathrm{graphcut}(X_I, S_{FI}, S_{BI})$.
5: Erode the foreground image $Y_{FI}$ by some small amount $\alpha$: $M_{FI} =$
   $\mathrm{erode}(Y_{FI}, \alpha)$. Use the result to get a final clean segmentation of the fore-
   ground range data: $Y_{FR} = \{x_r \,|\, (M_{FI} \times X_{F_2R}) \cap X_C \,\forall i\}$.
6: Set $Y_{FC} = (Y_{FI} \times Y_{FR}) \cap X_C$ and $Y_{BC} = (Y_{BI} \times Y_{BR}) \cap X_C$.
```

can be exploited since in the 3-D point cloud produced by range data images of the objects lying on the table surface, the data points corresponding to the objects tend to be well separated from the background table surface data points. Moreover, since the background points represent the flat surface of the table we can relatively easily fit a plane to the 3-D data, find the table surface points, and subtract those points from the data to find the object points. Since we also have correspondences between the 3-D points and the 2-D image data, we may then also use the 3-D segmentation of the object to aid in segmenting the object in the 2-D image. This idea may therefore be regarded as a form of multi-modal object segmentation, the modalities in this case being the 2-D image data and the 3-D point cloud from the range data respectively.

The method we developed is depicted visually in Figure 6.6 and in algorithmic form in Algorithm 6. The first step of the algorithm uses RANSAC (RANdom SAmple Consensus) [53] to fit a plane to the 3-D point cloud to retrieve the background table surface points. In the second step, these points are subtracted to reveal the object points. Often, however, the object points contain some outliers so in the third step we perform a clustering operation to find the largest cluster and discard the remainder. To do this, we use mean-shift clustering [31] which, although it is a non-parametric method, requires the specification of a kernel size. We choose a relatively large value for this in order to capture the largest cluster. Since we have correspondences between the image points and 3-D points, we may use this information to seed a graph-cut segmentation in the image. The image points that correspond to the object cluster points from the 3-D segmentation are used as foreground seeds, while the points from the plane-fitting are used as background points. The graph-cut segmentation method we used was from [109], which uses the min-cut/max-flow algorithms outlined in [10, 11, 87] to apply the standard graph cut technique to segmenting multi-modal tensor valued images.

In the case of the KUKA-LWR/Kinect setup, once again the objects always lie on a flat table surface, so we were able to make use of the *DominantPlaneSegmentation* module from the PCL library to segment the objects from the scene point cloud captured by the Kinect.

6.2.1.2 Visual Shape Features from Object Images

In the case of the Katana/Camera setup, visual features were extracted from objects segmented from the grayscale scene images produced by the Bumblebee camera. The multi-modal segmentation technique produced reasonably good intensity image segmentations of objects, and these were then used to calculate the following 9 shape features:

O_1^a: *Area*; number of pixels in the object region.

O_2^a: *Convex area*; number of pixels in the convex hull of the object region.

O_3^a: *Eccentricity*; ratio of the distance between the foci of the ellipse that has the same second-moments as the object region and its major axis length.

O_4^a: *Equivalent circular diameter*; diameter of a circle with the same area as the object region.

O_5^a: *Extent*; ratio of pixels in the object region to the pixels in its bounding box.

O_6^a: *Filled area*; number of pixels in the filled object region.

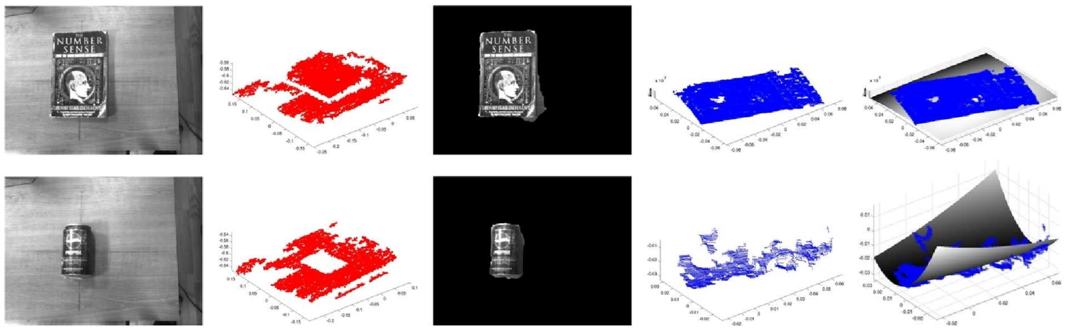


Figure **6.7:** Examples of image and range data taken with the stereo camera in the Katana/Camera setup for two different types of objects: a book which slides when pushed by the robotic arm, and a cola can which rolls when pushed by the arm. From left to right: intensity image, range data of the scene, segmented object, segmented object range data, object range data with a fitted quadric surface.

O^a_7: *Major axis length*; length of the major axis of the ellipse that has the same normalized second central moments as the object region.

O^a_8: *Minor axis length*; length of the minor axis of the ellipse that has the same normalized second central moments as the object region.

O^a_9: *Perimeter*; of the boundary of the object region.

6.2.1.3 3-D Shape Features from Object Point Clouds

In order to capture object surface shape properties, 3-D shape features were extracted from segmented object point clouds derived from both the Katana/Camera and the KUKA-LWR/Kinect experimental setups. This was intended to aid in differentiating between rolling and non-rolling objects for example, the subject of both Fitzpatrick's classic affordance experiments [55–57] and, indeed, our own experiments in Chapter 7. In both setups we took a strategy of surface fitting. For instance, in the case of the Katana/Camera setup, a quadratic surface was fitted to an object point cloud, or to a part of an object point cloud, in order to derive curvature features from the object surface. Given the points for an object or part of an object, we fit the following quadratic polynomial function,

$$Z = aX^2 + bXY + cY^2 + dX + eY + f \tag{6.1}$$

solving for the coefficients $a, b, c, d, e,$ and f. The principal curvatures of the surface are then given by taking *eigenvalues* of the matrix $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$, providing two features which form a good description of the curvature of the surface. As well as that, we fit planes to object or object part point clouds and used the *normal to the plane* to indicate the orientation of the points, which provided two features (we discard the z-coordinate since the point clouds were normalized with respect to the workspace coordinate frame).

Using these four surface features, two for planar orientation and two for curvature, in the Katana/Camera setup, we divided the object cloud into parts and extracted the four features for each part. These parts consisted of the global object cloud itself, as well as eight other parts found by dividing the object cloud evenly along two of its axes in various ways. This yielded the following list of features:

$O^b_{1...4}$: Global object point cloud plane/curve features.

$O^b_{5...8}$: x-axis split, left plane/curve features.

$O^b_{9...12}$: x-axis split, right plane/curve features.

$O^b_{13...16}$: y-axis split, front plane/curve features.

$O^b_{17...20}$: y-axis split, back plane/curve features.

$O^b_{21...24}$: x-y-axis split, front-left plane/curve features.

$O^b_{25...28}$: x-y-axis split, front-right plane/curve features.

$O^b_{29...32}$: x-y-axis split, back-left plane/curve features.

$O^b_{33...36}$: x-y-axis split, back-right plane/curve features.

In the case of the KUKA-LWR/Kinect setup we partitioned the object point clouds slightly differently as depicted in Figure 6.8, this time dividing each of the x, y and z axes evenly into two parts each and fitting planes to each of their respective part point clouds as well as to the entire object point cloud. The part division along the third dimension was motivated by the need to detect the additional object toppling affordance that resulted from the experiments performed

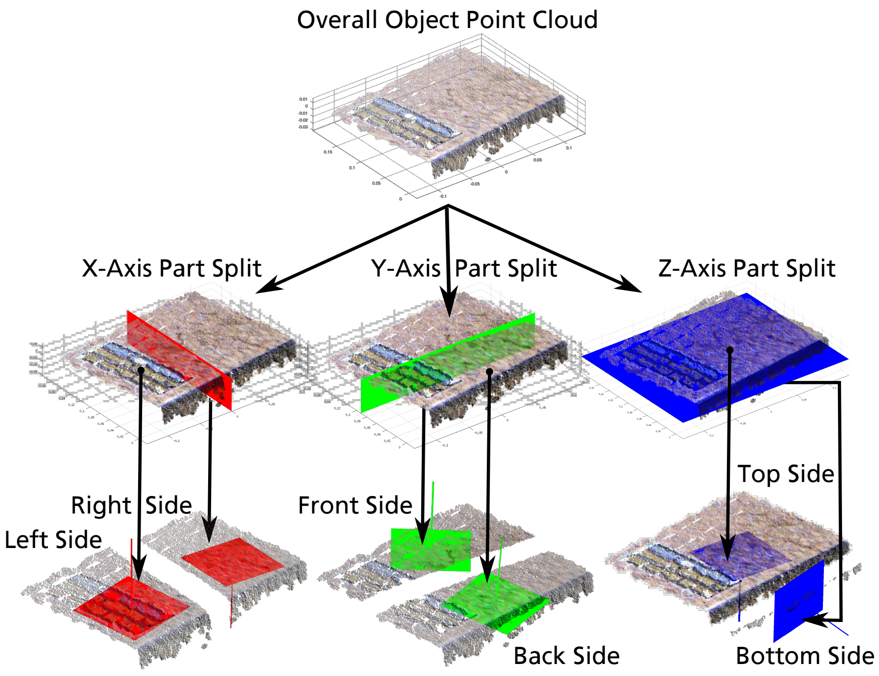


Figure **6.8:** Dividing a 3-D point cloud of a book into parts in the KUKA-LWR/Kinect setup. Top row: original object point cloud. Middle row: partitioning planes divide the point cloud evenly in each dimension to create sub-parts. Bottom row: planes are fitted to each sub-part for feature extraction.

with this setup. This time, object part centroids as well as plane normals and curvature features were used, resulting in seven features per part, three for part centroids, as well as two for the plane normals and two for the curvature features as before, resulting in the following list of features:

O^a_1: *Area*; number of pixels in the object region.

$O^c_{1...7}$: Global point cloud centroid/plane/curve features.

$O^c_{8...14}$: x-split, left centroid/plane/curve features.

$O^c_{15...21}$: x-split, right centroid/plane/curve features.

$O^c_{22...28}$: y-split, front centroid/plane/curve features.

$O^c_{29...35}$: y-split, back centroid/plane/curve features.

$O^c_{36...42}$: z-split, top centroid/plane/curve features.

$O^c_{43\ldots49}$: z-split, bottom centroid/plane/curve features.

6.2.2 Object Effect Features

When it came to the effect features, we chose to both track the objects in motion globally in the workspace and compare changes in object properties locally, deriving three main sets of features based on global motion of the object, changes in 3-D shape, and local appearance changes of the object respectively, depending on the platform.

6.2.2.1 Tracking Object Motion

In the Katana/Camera setup, after an arm action was performed on an object, the resulting videos of the interaction gathered from the Flea camera were processed for tracked object motion features. This was primarily achieved by tracking the object in motion using a probabilistic tracker from [93]. This tracker is in essence a colour-based particle filter, which also makes use of background subtraction using a pre-learned background image. Background subtraction by itself was insufficient to localise the object in our experimental setup due to changes in lighting and the motion of the arm, but it was helpful in reducing ambiguities for the tracker. Object shapes were approximated by elliptical regions, while their colour was encoded using colour histograms. The dynamics of objects were modelled using a dynamic model from [92], which allowed for tracking with a smaller number of particles, and consequently, near real-time tracking performance.

Estimating how the appearance of the object changed locally required developing an extension to the particle filter tracker previously described. The tracker by itself is sufficient for tracking the motion of objects, but it is slightly inaccurate at times. For example, if an object is rolling and stops suddenly, the tracker sometimes briefly overshoots the object before returning to it a few frames later. While this does not affect global motion tracking too significantly, it is an unacceptable starting point if we wish to use the output of the tracker to segment an object from frame to frame before calculating local appearance estimates. If the tracker overshoots the object, this causes subsequent segmentation to be inaccurate, producing major inaccuracies in appearance calculations. To avoid this, we use the output of the tracker to define a broad window around the object in the video frames, before using colour histogram back-projection [175] to localise the object within the window. This tracking refinement process is described algorithmically in 7 and illustrated in Figures 6.11 and 6.12.

Algorithm 7 Object Tracking Refinement

```
Input: Sequence of images $\mathcal{X} = \{X^1, \ldots, X^n \,|\, X^i \subset \mathbb{R}^3 \,\forall i\}$, Object trajectory
    from particle filter tracker $\mathcal{T} = \{p^1, \ldots, p^n \,|\, p^i \in \mathbb{R}^2 \,\forall i\}$, Background model
    $B$.
Output: Refined object trajectory $\mathcal{T}' = \{q^1, \ldots, q^n \,|\, q^i \in \mathbb{R}^2 \,\forall i\}$.
 1: Convert background model image to HSV colour map: $B = \mathrm{rgb2hsv}(B)$.
 2: for $i = 1, \ldots, n$ do
 3:     $X^i = \mathrm{rgb2hsv}(X^i)$.
 4:     Crop image $X^i$ around point $p_i$ to window size $\theta$: $X^i_C = \mathrm{crop}(X^i, p^i, \theta)$.
 5:     Crop background model $B$: $B_C = \mathrm{crop}(B, p_i, \theta)$.
 6:     if $i == 1$ (we do not have an object model) then
 7:         Background subtraction: $S^i_C = X^i_C - B^i_C$.
 8:         Given $S^i_{Cj} = \{S^i_{Ch}, S^i_{Cs}, S^i_{Cv}\}$, the HSV components of $S^i_C$, set $k =$
            $\arg\max_j(\mathrm{area}(\mathrm{threshold}(S^i_{Cj})))$.
 9:         Create mask: $S^i_{Mk} = \mathrm{threshold}(S^i_{Ck})$.
10:         Detect seperate mask objects: $\mathcal{S} = \{O^1, \ldots, O^m\} = \mathrm{detectBlobs}(S^i_{Ck})$.
11:         Select largest object: $O = \arg\max(\mathrm{area}(\mathcal{S}))$.
12:         Crop object to use as our object model: $M =$
            $\mathrm{crop}(S^i_C, \mathrm{centre}(O), \mathrm{sizeof}(O))$.
13:     end if
14:     Perform colour histogram back-projection: $q^i_C = \mathrm{histBackProj}(S^i_C, M)$.
15:     Transform cropped coordinates to un-cropped image frame: $q^i =$
        $\mathrm{transform}(X^i, X^i_C, p^i, \theta)$.
16: end for
```

The KUKA-LWR/Kinect setup, on the other hand, made use of the *libpcl_tracking* library from the PCL, which also uses a particle filter to estimate 3-D object poses using Monte Carlo sampling techniques and calculates the likelihood using combined weighted metrics for hyper-dimensional spaces including Cartesian data, colors, and surface normals. These 3-D object models were gathered from scene point clouds using the same dominant surface segmentation method discussed in Section 6.2.1.1. The real-time 3-D object tracking provided by this method is illustrated in Figure 6.10. It is optimized to perform com-

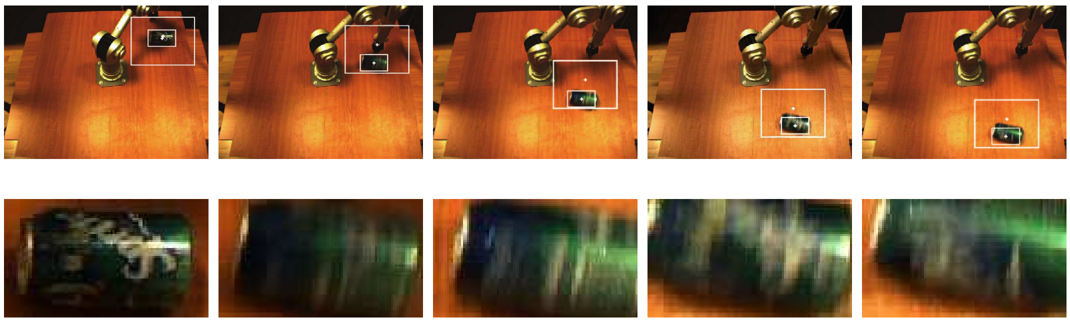


Figure **6.9:** An example of the object tracking mechanism from the Katana/Camera setup described in Section 6.2.2.1 using images obtained by the Flea monocular camera (cf. Section 6.1.1). The images in the first row show a progression of frames tracking a soda can being pushed by the arm. The outer rectangle is a likelihood window around the object obtained using the particle filter tracker. The inner rectangle is the result of using histogram back-projection within that window to localise the object. The second row of close-up images shows how the appearance of the object within the inner rectangle changes during the course of object motion.

putations in real-time, by employing multiple CPU core optimization, adaptive particle filtering (KLD sampling) and other modern techniques.

6.2.2.2 Detecting Arm-Object Contact

An important consideration in the calculation of the features we describe below is how to determine when the arm has made contact with the object. This task would be greatly simplified by the integration of haptic sensors to the scenario, but since we did not make use of such sensors, a visual solution had to be employed. We used a technique very similar to that described by Fitzpatrick [57] in his Ph.D. thesis, which is based on the idea of detecting an explosion of movement between frames. In our case, background subtraction and thresholding we identify the largest connected component in the thresholded image. We assume that the largest connected component is the mask of the segmented object, and by recording its mean area over a progression of frames and comparing it to the area of the largest connected component in the current frame, if there is a significant difference in these values, then it is likely that the object has been disturbed by the arm. This is visualized in Figures 6.11 and 6.12.

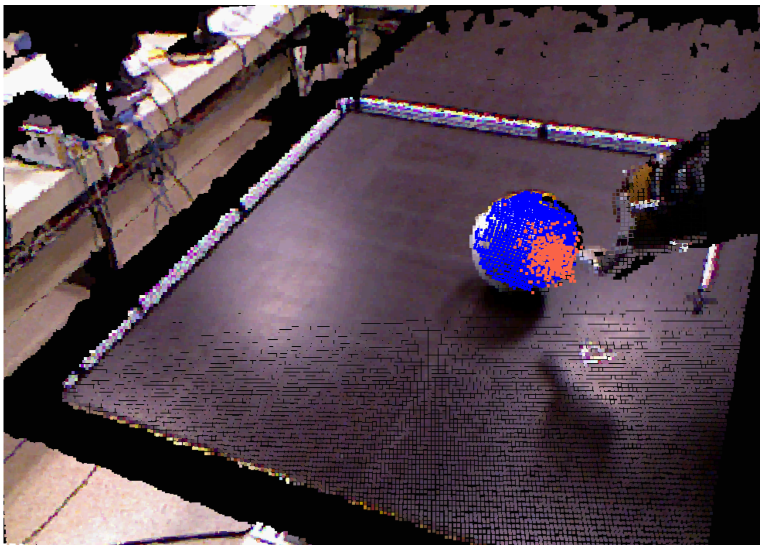


Figure **6.10:** PCL-based particle filter tracking from KUKA-LWR/Kinect setup. Blue points: model of the ball object derived from segmenting the ball from the table prior to interaction. Orange points: particle filter points used to track the object during interaction and update the model.

6.2.2.3 Object Motion Features

Using the output of the visually-based particle filter tracker of the Katana/Camera setup, the following 9 features were calculated:

$E^a_{1...2}$: Total distance travelled in x & y dimensions,

E^a_3: Total Euclidean distance travelled,

$E^a_{4...5}$: Mean velocity in x & y dimensions,

$E^a_{6...7}$: Velocity variance in x & y dimensions,

$E^a_{8...9}$: Final x & y positions,

In the case of the KUKA-LWR/Kinect setup, since the 3-D point cloud-based particle filter tracker from the PCL library allowed for tracking objects in three dimensions, we were able to extend some of 2-D features used on the previous setup into the z dimension. Some alternative features were also designed in addition to suit the experiments performed with this platform. The 17 resulting features are listed as follows:

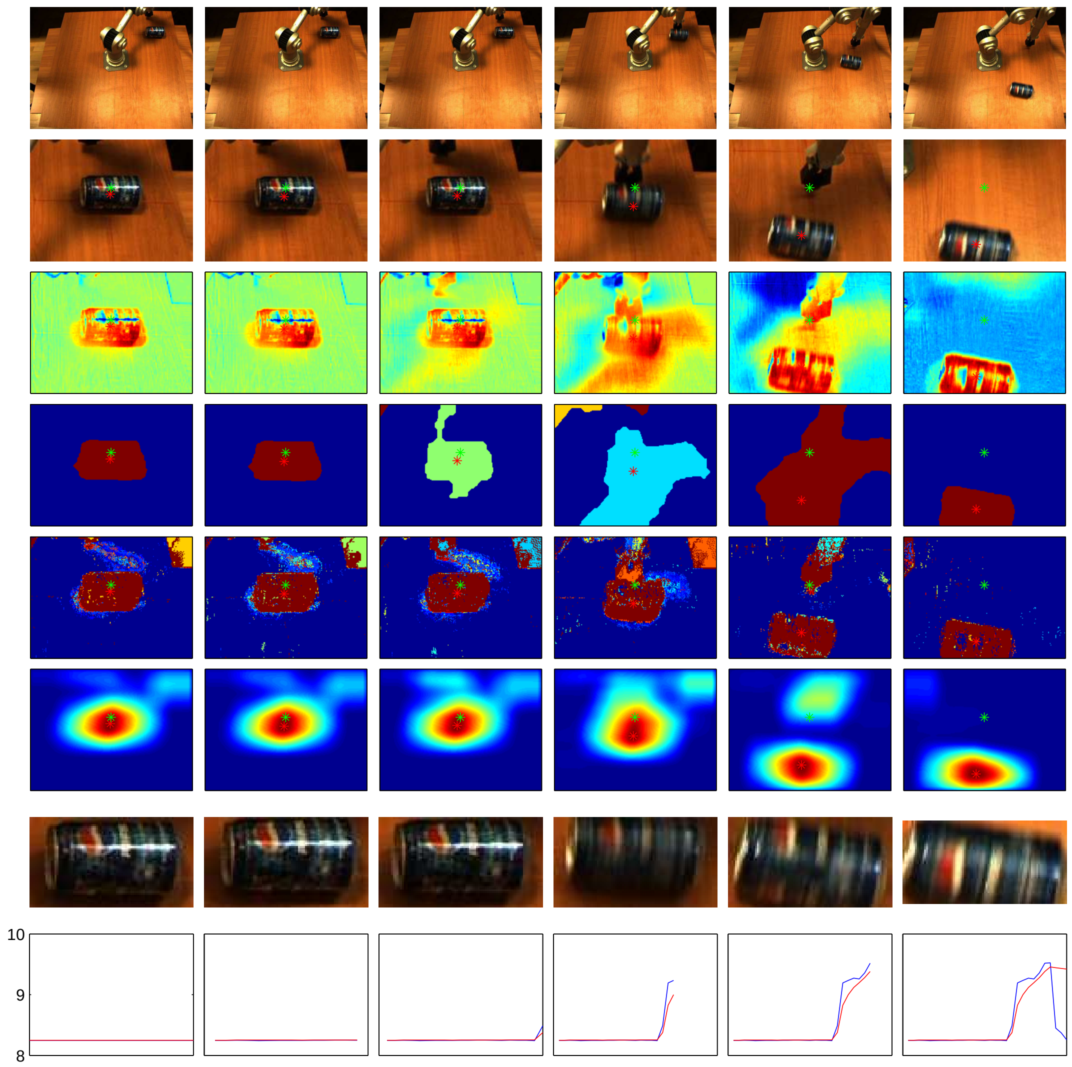


Figure **6.11:** Object tracking refinement in the Katana/Camera setup using histogram back-projection for the rolling cola can object. First row: original image sequence. Second row: cropped windows around particle filter tracker trajectory points. Third row: background subtraction. Fourth row: labelled components after background subtraction and thresholding. Fifth row: back-projected images. Sixth row: convolved back-projected images. Seventh row: localised image of object after tracking refinement. Eight row: area of of largest component in fourth row (blue line) versus mean area of largest component.

$E^b_{1\ldots3}$: Total distance travelled in x, y & z dimensions,

E^b_4: Total Euclidean distance travelled in $\mathbb{R}^2$,

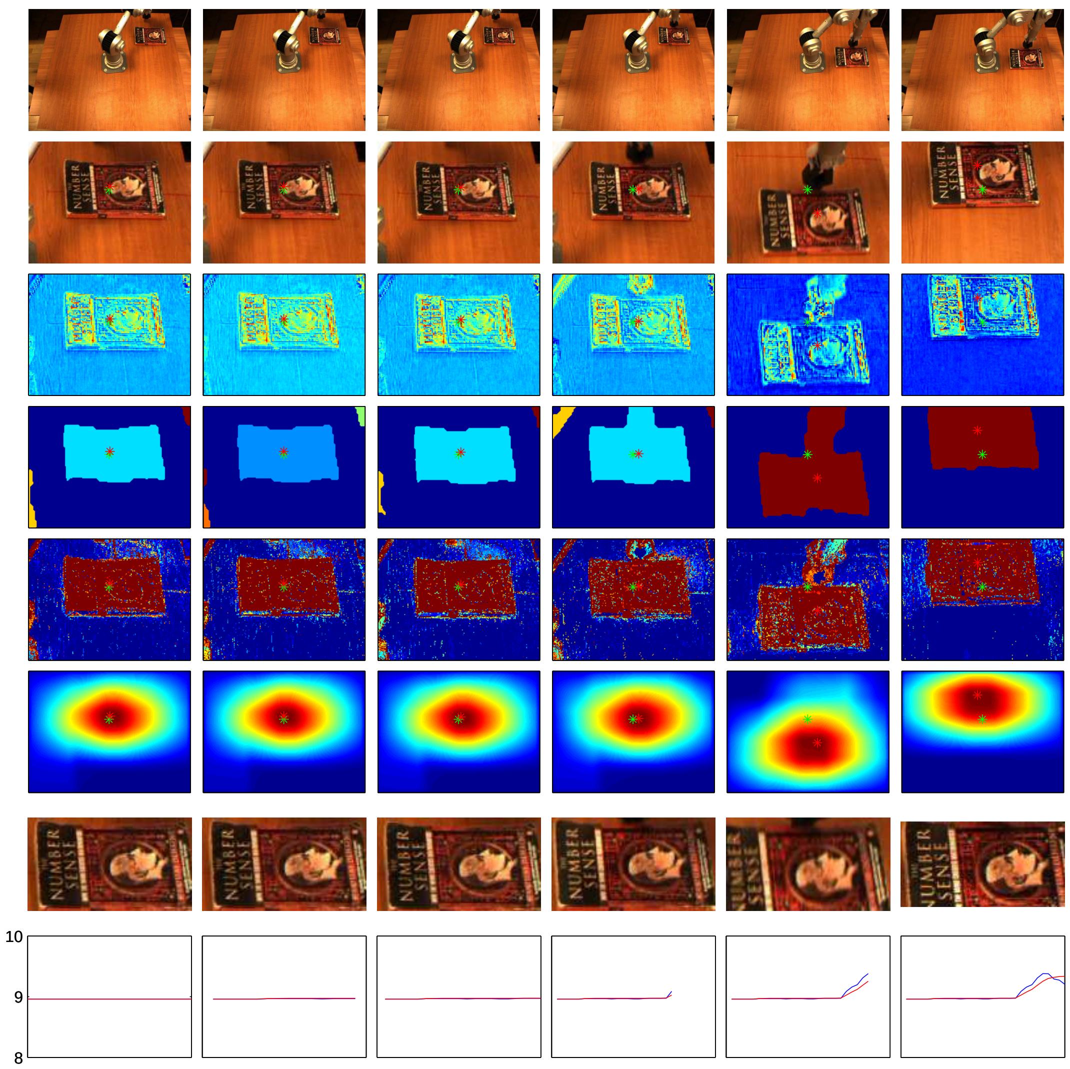


Figure **6.12:** Object tracking refinement in the Katana/Camera setup using histogram back-projection for the non-rolling book object. First row: original image sequence. Second row: cropped windows around particle filter tracker trajectory points. Third row: background subtraction. Fourth row: labelled components after background subtraction and thresholding. Fifth row: back-projected images. Sixth row: convolved back-projected images. Seventh row: localised image of object after tracking refinement. Eight row: area of of largest component in fourth row (blue line) versus mean area of largest component.

E_5^b: Total Euclidean distance travelled in $\mathbb{R}^3$,

$E_{6...8}^b$: Final x, y & z positions.

E^b_9: Time in motion.

E^b_{10}: Trajectory extent volume.

E^b_{11}: Trajectory convex hull volume.

E^b_{12}: Trajectory convex hull surface area.

E^b_{13}: Summed trajectory point distance from start position.

E^b_{14}: Mean trajectory point distance from start position.

E^b_{15}: Trajectory point distance variance from start position.

E^b_{16}: E^b_{14}, weighted to favour points near end of trajectory.

E^b_{17}: E^b_{15}, weighted to favour points near end of trajectory.

6.2.2.4 Object Shape Change Features

In the case of the experiments performed using the KUKA-LWR/Kinect setup, since we intended to study toppling affordances, we required features capable of detecting the types of changes in object shape in three dimensions that would result from such affordances, thus, using a similar methodology to that of Ugur *et al.* [188], we derived a set of 3-D shape change features by taking the difference between the $O^c_{1\ldots49}$ 3-D shape features from Section 6.2.1.3 recorded from the object pre-interaction and the same features recorded from the object post-interaction. These new features, $E^c_{1\ldots49}$, were therefore derived as follows:

$E^c_{1\ldots49}$: $O^c_i(t_e) - O^c_i(t_s)$ for $i = 1\ldots49$ where $O^c_i(t_s)$ and $O^c_i(t_e)$ are the object shape features O^c_i extracted from the object at time t_s, when it is at its start position, and at time t_e, when at its end position respectively.

6.2.2.5 Object Appearance Change Features

In the experiments involving the Katana/Camera setup, in order to estimate how the appearance of the objects change during, motion (cf. Figure 6.9), and thus potentially capture some of the differences in visual effects between rolling and non-rolling objects, We derive 3 output modality features from this procedure: average colour histogram difference, average edge histogram difference, and the product of these two values. we calculated the average difference of both colour and edge histograms between video frames of the objects, the aim being to detect both motion blur and the texture changes characteristic of many rotating objects. Histogram difference averages were then calculated from the start of object motion until the end. See Figure 6.9 for sample frames from an interaction with an object that illustrates this technique at work.

We derived three effect features from this procedure:

E^d_1: Average colour histogram difference.

E^d_2: Average edge histogram difference.

E^d_3: Product of E^d_1 and E^d_2.

6.3 Chapter Summary

In this chapter we described the development of the main robotic system that we initially used to address the object affordance learning problem, as well as a secondary system that was used subsequently. In the first section we discussed the hardware that was used in these systems, including robotic arms and various camera systems. We also highlighted some important considerations with respect to the operation of such robotic arms within the context of object pushing experiments, as well as for the working environments of the systems for the purposes of performing experiments. In the second section we addressed the computer vision methods we used for feature extraction from the camera images in two main categories: object features extracted from static images of objects and 3-D point cloud data, and effect features extracted from videos of objects in motion as well as post-interaction point clouds. We presented two different algorithms in each of these areas that built on existing computer vision methods to aid in the feature extraction process. The first algorithm used information in the 3-D point cloud data to seed a graph cut segmentation in image data in order to segment objects from background before extracting shape features. The second algorithm used colour histogram backprojection to refine the output of a particle filter tracker such that local object appearance change features could be extracted.

Chapter 7

Experimental Results

The experimental results presented in this chapter are broadly broken down into four sections. We wanted to evaluate the feature relevance determination algorithms from Chapter 5 in isolation in a fully-supervised setting since, if they were proven not to be useful in such a setting where the class labels are present during training, they were unlikely to be of any additional use in a self-supervised setting where the class labels are unknown. Thus, in Section 7.1 we present the results of evaluating the feature relevance determination algorithms on both synthetically generated labelled data as well as on some popular real-world datasets. In Section 7.2, we test the self-supervised learner in a controlled setting using synthetic cross-modal data, examining various combinations of the base learning algorithms, prototype culling and feature relevance algorithms. Finally, in Sections 7.3 and 7.4, we describe object push affordance learning experiments performed using the robotic setups described in Chapter 6 and demonstrate how the self-supervised learner may be successfully applied to the resulting data.

7.1 Experiments on Feature Relevance Determination Algorithms

The proposed feature relevance determination algorithms from Chapter 5 were evaluated over simulated data and real-world datasets from the UCI repository [58]. In the following, the datasets are described in more detail in Section 7.1.1, our evaluation procedure is discussed in Section 7.1.2, algorithm setup is provided in Section 7.1.3, and experimental results are presented in Section 7.1.4.

Table **7.1:** An attribute list for the datasets in Section 7.1.1.

DATASET	# FEATURES	# SAMPLES	# CLASSES
SIMULATED	10	90	3
IRIS	4	150	3
IONOSPHERE	34	351	2
WINE	13	178	3
SOYBEAN	35	47	4
WBC	30	569	2

7.1.1 Data

Two simulated datasets were proposed in [8, 69] to test the RLVQ and GRLVQ algorithms, the first of which was replicated for the experiments here. The data is composed of three classes, each separated into two clusters with some small overlap to form multi-modal class data distributions in the first two dimensions. Eight further dimensions are generated from the first two dimensions as follows: assuming (x_1, x_2) is one data point, $x_3 = x_1 + \eta_1, \ldots, x_6 = x_1 + \eta_4$ is chosen where η_i comprises normally-distributed noise with variances 0.05, 0.1, 0.2, and 0.5 respectively. The remaining $x_7, \ldots, x_{10}$ components contain pure noise uniformly distributed in $[-0.5, 0.5]$ and $[-0.2, 0.2]$. This dataset is multi-modal for each class in the two relevant dimensions and thus provides a good test for the potential difference between the two proposed algorithms.

Five different datasets from the UCI repository [58] were tested: Fisher's Iris dataset, the ionosphere dataset, the wine dataset, the soybean dataset (small), and the Wisconsin breast cancer (WBC) dataset.

7.1.2 Evaluation Procedure

The primary goal of the investigation was to evaluate whether or not the newly proposed feature relevance algorithms from Chapter 5, when applied to standard LVQ methods such as LVQ1 and GLVQ, offer performance improvements over those methods in their original form, as well as over other relevance determination techniques for LVQ, such as RLVQ and GRLVQ. The results of these comparisons are outlined in Table 7.2 and are discussed in more detail in the following. In the results, the proposed Fisher criterion score-based relevance determination algorithms (cf. Sections 5.2.1 and 5.2.2) are referred to as FC1LVQ1 and FC2LVQ1 respectively when applied to LVQ1, and FC1GLVQ and FC2GLVQ when applied to GLVQ.

A secondary consideration was to test the methods under the duress of various different conditions. GLVQ, for example is known to perform poorly if the prototype vectors are not initialized within the data distribution [140], thus in the evaluations discussed below, both random prototype initializations as well as initializations where the prototypes are placed at the mean points of class clusters were considered. Note that random prototype initialization in this case refers to selecting random values for each prototype dimension scaled within the data range, not random sampling of vectors from the training data. K-means clustering was used to determine class clusters in the latter case.

We evaluate the learners by training them online over several epochs, that is, by updating the algorithms online with each sample in the training set and then repeating this process several times, possibly with randomized sample ordering. Each run through the training data is referred to as an epoch. This type of training usually performed in order to simulate large datasets, or to simulate extended training durations, or both. The performance of LVQ algorithms over short training periods, e.g. one epoch of training, is not often considered in the literature, which tends to favour evaluations of the algorithms over several hundred training epochs to test for convergence. Training over multiple epochs implies having access to the entire training set such that the algorithm has the opportunity to revisit previous training data over again as often as is specified. Given that LVQ algorithms have online training mechanisms, and that the proposed relevance determination techniques were explicitly developed to also function online, sample-by-sample without access to the rest of the training set, such short-term training evaluations are important if the methods are to be considered useful in real-world online settings, autonomous robotics for instance, where the entire training set is often unavailable during training. With this in mind, we present the following evaluations over both short and long term epoch periods.

7.1.3 Algorithm Setup

Thus, the results in Table 7.2 are divided into four main evaluations: both one epoch and 300 epochs of training from random initialization, and both one epoch and 300 epochs of training from class cluster mean initialization. The 300 epoch sessions used the relatively slow learning rates of $\alpha = 0.1$ for the prototype updates (cf. Equations (4.41), (4.44) & (4.45)) and $\beta = 0.01$ for the dimensional relevance updates where required (cf. Equations (5.1) & (5.3)), whereas the one epoch training sessions used the faster rates of $\alpha = 0.3$ and $\beta = 0.1$. Note that the FC1 and FC2 methods do not require the additional β learning rate. In each of the one epoch evaluations, 20 trials of ten-fold cross validation were performed with random data orderings in each trial, and results were averaged over test data performance, whereas in the 300 epoch evaluations, 5 trials were performed. We run multiple trials to account for variation in both the random prototype initialization as well as in the data orderings. 10 prototypes were used for every dataset and the data dimensions were scaled prior to training.

7.1.4 Results

7.1.4.1 Classification

The results in Table 7.2 show that when trained over a single epoch from random initialization, of the algorithms tested FC2LVQ1 and FC2GLVQ offer the best performance, in some cases offering substantial improvements over either LVQ1 or GLVQ, while also generally beating both RLVQ and GRLVQ. Over long-term training of 300 epochs from random initialization, the results for all algorithms aside from GLVQ, tend to improve significantly with FC2LVQ1 and FC2GLVQ again tending to be more competitive than their counterparts with certain marginal exceptions. It is worth noting here the significant impact relevance determination has on improving the results of GLVQ when exposed to poor prototype initialization. When the prototypes are initialized optimally at the class cluster mean points the results tend to improve dramatically across all of the classifiers in short-term training, with both FC1 and FC2 relevance determination doing well over both short-term and long-term training periods, with FC1 out-performing FC2 in some cases and vice versa. Over all the evaluations, FC1GLVQ and FC2GLVQ trained over 300 epochs with class cluster mean initialization tend to provide the best performance. It should also be noted that, when the class distribution in the data is multi-modal, as is the case with the simulated dataset, FC2-based methods tend to be a better choice than FC1-based methods, as predicted.

Table **7.2:** Evaluation of various supervised feature relevance determination algorithms over a simulated dataset and five datasets from the UCI repository. 10-fold cross validation, 10 prototypes. Best results for LVQ1 & GLVQ based algorithms are shown in bold. Mean correct classification rates over all folds/trials are shown with errors in standard deviations.

Dataset	LVQ1	RLVQ1	**FC1LVQ1**	**FC2LVQ1**	GLVQ	GRLVQ	**FC1GLVQ**	**FC2GLVQ**
Random Initialization, one Epoch of Training, 20 Trials								
Sim	53± 18%	64± 22%	54± 19%	**69± 18%**	37± 17%	63± 22%	51± 20%	**70± 19%**
Iris	90± 8%	91± 9%	93± 9%	**95± 5%**	63± 24%	**89± 13%**	83± 19%	88± 15%
Iono	81± 8%	75± 11%	**85± 6%**	84± 7%	66± 13%	80± 9%	82± 7%	**84± 7%**
Wine	93± 6%	79± 13%	92± 9%	**94± 6%**	52± 19%	92± 8%	85± 14%	**94± 7%**
Soy	**89± 17%**	83± 24%	89± 18%	85± 21%	34± 27%	84± 22%	83± 21%	**85± 20%**
WBC	92± 4%	86± 8%	93± 4%	**93± 3%**	71± 19%	93± 5%	90± 10%	**94± 3%**
Random Initialization, 300 Epochs of Training, 5 Trials								
Sim	79± 14%	79± 13%	77± 16%	**87± 12%**	38± 17%	**96± 7%**	90± 12%	94± 9%
Iris	92± 7%	92± 8%	95± 5%	**96± 5%**	47± 24%	96± 5%	91± 16%	**96± 4%**
Iono	85± 7%	80± 10%	**86± 8%**	85± 7%	60± 16%	**90± 5%**	90± 6%	89± 6%
Wine	95± 5%	77± 11%	95± 5%	**96± 5%**	42± 18%	96± 5%	97± 4%	**98± 3%**
Soy	99± 6%	97± 10%	**100± 4%**	98± 7%	33± 26%	97± 8%	**97± 7%**	96± 9%
WBC	93± 3%	87± 7%	**94± 3%**	**94± 3%**	62± 20%	96± 3%	96± 3%	**96± 2%**
Class Cluster Mean Initialization, one Epoch of Training, 20 Trials								
Sim	82± 12%	**98± 5%**	78± 17%	93± 8%	90± 9%	91± 9%	85± 13%	**93± 8%**
Iris	**96± 5%**	**96± 5%**	**96± 5%**	**96± 5%**	95± 5%	95± 5%	95± 5%	**96± 5%**
Iono	87± 6%	80± 10%	**88± 6%**	**88± 6%**	**90± 5%**	88± 6%	89± 5%	**90± 5%**
Wine	95± 5%	86± 11%	**96± 5%**	**96± 5%**	**97± 4%**	97± 5%	97± 5%	97± 5%
Soy	**100± 2%**	95± 10%	100± 3%	99± 5%	**100± 2%**	99± 4%	**100± 2%**	99± 5%
WBC	**95± 3%**	88± 7%	94± 3%	94± 3%	96± 3%	96± 3%	**97± 3%**	95± 3%
Class Cluster Mean Initialization, 300 Epochs of Training, 5 Trials								
Sim	84± 11%	86± 16%	87± 12%	**91± 10%**	90± 9%	**97± 6%**	90± 10%	96± 8%
Iris	96± 5%	95± 6%	**96± 4%**	96± 5%	96± 6%	95± 5%	**97± 4%**	96± 4%
Iono	88± 5%	82± 9%	**89± 5%**	88± 5%	89± 5%	90± 5%	90± 5%	**91± 5%**
Wine	96± 5%	82± 12%	**97± 4%**	96± 5%	97± 4%	**98± 3%**	**98± 3%**	**98± 3%**
Soy	**100± 0%**	94± 11%	99± 6%	98± 7%	**100± 0%**	98± 8%	99± 5%	99± 6%
WBC	**96± 2%**	89± 5%	95± 3%	95± 3%	96± 3%	96± 3%	**97± 2%**	**97± 2%**

Table **7.3:** FC1GLVQ & FC2GLVQ versus SVM.

Dataset	FC1GLVQ	FC2GLVQ	SVM
Simulated	90±10%	**96±8%**	78±14%
Iris	**97±4%**	96±4%	96±6%
Ionosphere	90±5%	91±5%	**94±4%**
Wine	**98±3%**	**98±3%**	**98±3%**
Soybean	99±5%	99±6%	**100±0%**
WBC	97±2%	97±2%	**98±2%**

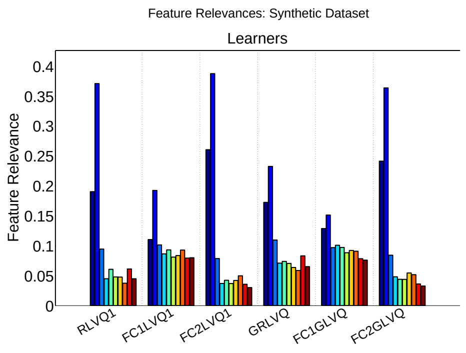
A third consideration was to compare the new methods to a state-of-the-art batch method such as the *support vector machine (SVM)*. Batch methods, as opposed to online methods that are trained sample-by-sample, have access to the entire training set during training, and therefore usually provide superior results. Table 7.3 shows the results of a comparison between FC1GLVQ, FC2GLVQ and a multi-class SVM trained with a radial basis function (RBF) kernel [19]. For this comparison, the results for FC1GLVQ and FC2GLVQ from the 300 epoch, class cluster mean-initialized evaluation described previously were used, while ten-fold cross validation over five trials was also used for the SVM, where the test data results were averaged over the five trials and SVM parameters were optimized using cross validation over the training data prior to training. The results show both FC1GLVQ and FC2GLVQ either bettering or equalling SVM on four out of the seven tested datasets and coming within 1% of its performance on two of the remaining three datasets. Both methods also perform significantly better than SVM on the multi-modal simulated dataset.

7.1.4.2 Feature Relevance Determination

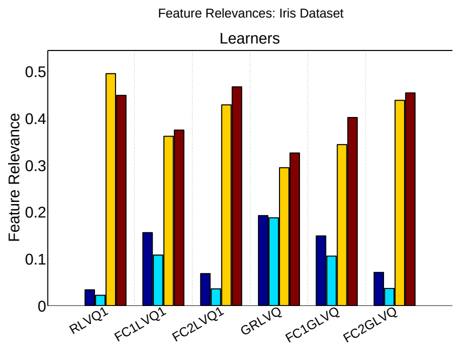
It is difficult to evaluate the feature relevance determination itself since we do not always have ground truth information for which features in a given dataset are most relevant for classification, but we can analyse how feature weights are distributed by the various algorithms and find correlations between these and the respective classification scores. Figure 7.1 shows bar plots of feature weights determined by each of RLVQ1, FC1LVQ1, FC2LVQ1, GRLVQ, FC1GLVQ and FC2GLVQ for each of the datasets tested in the case of random prototype initialisation and one epoch of training, the setting we are most interested in. These bar plots were calculated by averaging the final feature weights of each learner averaged over all folds and all trials for each dataset. In the case of the synthetic dataset, FC1, the least effective of the feature relevance methods in terms of classification scores, appears to not attribute sufficient relevance to the first two features, unlike RLVQ1, the FC2-based methods, and to a lesser extent GRLVQ. This was an expected result given the multi-modal nature of the data distribution in the synthetic dataset and the reason behind this is explained in Section 5.2.1 of Chapter 5. In the case of the Ionosphere dataset, RLVQ1 is the worst performer of the feature relevance methods in terms of classification results and Figure 7.1(c) indicates that this could be attributed to an over-emphasis on the first and second features of the dataset. Similar could be said of the performance of RLVQ1 over both the Soybean and WBC datasets, as indicated by Figures 7.1(e) and 7.1(f) respectively, where the middle range of features appear to be accorded too much weight in each case.

7.1.5 Summary

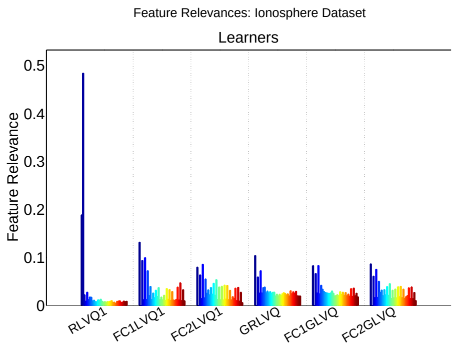
Overall, we can conclude that feature relevance determination can have a significant impact on classification results for fully-supervised LVQ methods, particularly over short-term training periods. It also appears to be helpful in compensating for the problems GLVQ suffers with with poor prototype initialisation, as is evident in the results of Table 7.2. Both of our proposed methods perform competitively in most situations, often out-performing the competitors evaluated in the experiments presented here. In the following section, we turn our attention towards self-supervised learning and demonstrate how feature relevance determination can be beneficial in boosting classification results in that context.



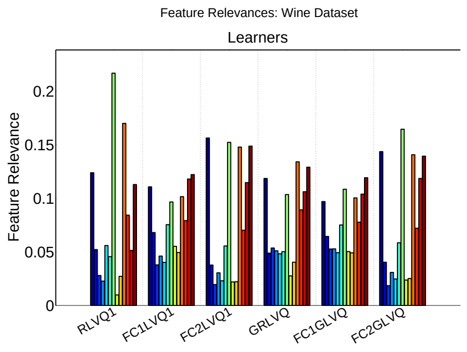
(a) Synthetic



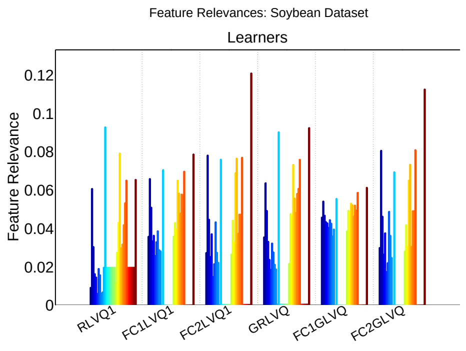
(b) Iris



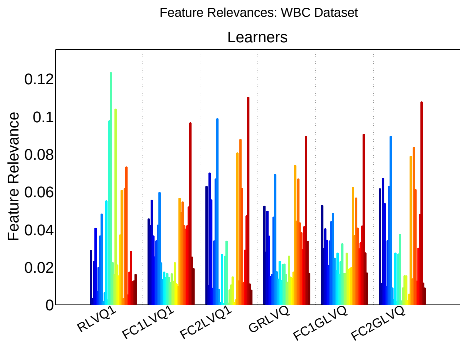
(c) Ionosphere



(d) Wine



(e) Soybean



(f) WBC

Figure **7.1:** Feature relevance bar plots for fully-supervised learners acting on the synthetic dataset and the UCI datasets. Results correspond to the first row of Table 7.2 for random initialization, one epoch of training and 20 trials of 10-fold cross-validation. The bar plots show feature weights at the end of training averaged over all folds and trials.

7.2 Experiments on Self-Supervised Multi-View Learning Algorithms

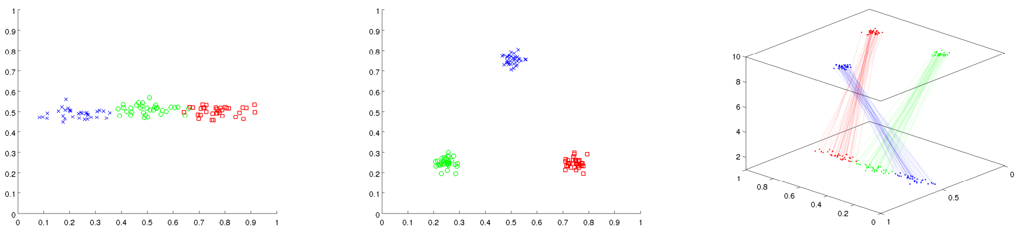
In this section, we describe experimental results for self-supervised multi-view learning over synthetically generated multi-view datasets, or more specifically, dual-view datasets, i.e. datasets with co-occurring data in two separate data views. The data are described in more detail in the next sub-section, followed by the evaluation procedure in Section 7.2.2, the experiments that were performed in 7.2.3, and finally, the results in Section 7.2.4. In describing the results we analyse a number of different aspects of self-supervised multi-view learning as we have presented it in this thesis, namely, unsupervised class discovery, unsupervised cross-modal discriminative learning, full-supervision versus self-supervision, feature relevance determination in the self-supervised setting, and LVQ1-based training versus GLVQ-based training.

7.2.1 Data

The bulk of the analysis work presented in this section on self-supervised multi-view learning was based on dual-view synthetic data generated from three-class ground-truth distributions. This data is described in the following Section 7.2.1.1. In a follow-up study, we performed additional experiments using a five-class dual-view dataset, which is described in Section 7.2.1.2.

7.2.1.1 Three-Class Synthetic Dual-View Dataset

For this synthetic dataset, two sets of 99 co-occurring data points were generated in two separate 13-dimensional data views. The data in the first two dimensions of the first (input) view were distributed in three Gaussian clusters as depicted in Figure 7.2(a). To generate the remaining 11 dimensions, Gaussian-distributed noise was added to the first dimension with progressively larger variances in each dimension as follows: $0.05, 0.145, 0.24, 0.335, 0.43, 0.525, 0.62, 0.715, 0.81, 0.905, 1.0$. The data in the first two dimensions of the second (output) view were distributed in three Gaussian clusters as depicted in Figure 7.2(b). Again, to generate the remaining 11 dimensions, Gaussian-distributed noise was added to the first dimension with the same progressively larger variances in each dimension as in the first view. The data points were given ground-truth labels reflecting their cluster patterns as in Figure 7.2. It should be noted here that these ground-truth labels were not provided to the self-supervised algorithms during training in the results described below, whereas they were provided to the fully-supervised algorithms, which were trained solely on data from the input view.

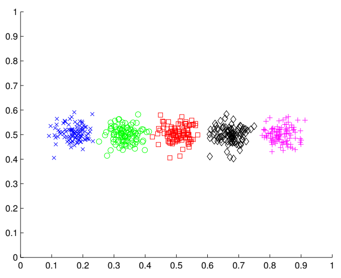


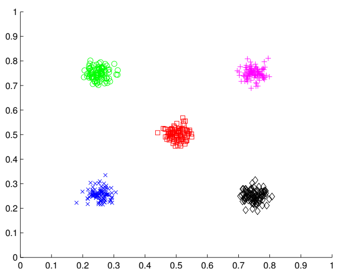
(a) First two dimensions of input view. (b) First two dimensions of output view. (c) First two dimensions of each view cross-view connected.

Figure **7.2:** Projections from a three-class dual-view synthetic dataset. Only the first two dimensions of each data view are shown. Each of the views contain a further 11 dimensions of noise progressively distorting each of their first dimensions.

7.2.1.2 Five-Class Synthetic Dual-View Datasets

For this synthetic data, this time, two sets of co-occurring data points were generated in two separate 12-dimensional data views and the data in the first two dimensions of the first (input) view were distributed in five Gaussian clusters as depicted in Figure 7.3(a). To generate the remaining 10 dimensions, Gaussian-distributed noise was added to the first dimension with progressively larger variances in each dimension as follows: $0.225, 0.394, 0.511, 0.606, 0.687, 0.76, 0.826, 0.888, 0.946, 1.0000$ The data in the first two dimensions of the second (output) view were distributed in five Gaussian clusters as depicted in Figure 7.3(b) Again, to generate the remaining dimensions, noise was added to the first dimension with the same progressively larger variances in each dimension as in the input view. In addition, rather than gathering a single small dataset, as in the previous three-class case and re-updating our learners with the data repeatedly over multiple epochs in order to simulate a larger dataset, this time, we generated two datasets for the five-class case, one small (500 samples) and one large (50,000 samples), via repeated sampling.


(a) First two dimensions of input view.


(b) First two dimensions of output view.

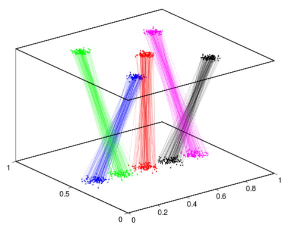

(c) First two dimensions of each view cross-view connected.

Figure **7.3:** Projections from a five-class dual-view synthetic dataset. Only the first two dimensions of each data view are shown. Each of the views contain a further 10 dimensions of noise progressively distorting each of their first dimensions.

7.2.2 Evaluation Procedure

To test our self-supervised learning paradigm on the synthetic datasets described above, we performed k-folds cross-validation with a k value of 10 for multiple trials on various self-supervised classifiers, evaluating their performance online at regular intervals over the training period. Given the self-supervision aspect, the evaluation criteria, by necessity, differed from the traditional match-counting utilized to evaluate fully-supervised classifiers. Meta-clusters of prototypes were found in the output view and subsequently matched to the ground truth by first matching all ground truth labelled training data to nearest-neighbour output view prototypes, then assigning each meta-cluster the ground truth label which their respective prototypes matched to most frequently. Then, given a test sample consisting of an input view test vector $\mathbf{x}^i$ and an output view test vector $\mathbf{y}^i$, the input view codebook was tasked with predicting an output view meta-cluster V^j using $\mathbf{x}^i$ as input for Algorithm 4. The output view test vector $\mathbf{y}^i$ was then matched to a meta-cluster V^k in the output view via the nearest-neighbour rule. If the V^j meta-cluster predicted by the input view codebook matched the V^k

cluster and that cluster also matched the ground truth label for the test sample, this was deemed to be a true positive.

7.2.3 Experiments

7.2.3.1 Three-Class Synthetic Dual-View Dataset

Test runs were performed for the unsupervised multi-view discriminative learning described in Section 4.2 the self-supervised multi-view discriminative learning described in Section, 4.3 4.3.4, alongside fully-supervised algorithms, using various combinations of both the prototype culling described in Section 4.2.2.4 and the feature relevance determination described in Chapter 5. It should be noted here that prototype culling may also be applied to the fully-supervised learners in a similar way to the self-supervised learners, by recording histograms of prototype update frequency during training. In the results figures shown below, we denote instances of the use of prototype culling with the abbreviation PC. With regard to feature relevance determination, we tested both fully-supervised and self-supervised versions of RLVQ1, FC2LVQ1, GRLVQ and FC2GLVQ. In such cases, the feature relevance weights generated by the algorithm were applied adaptively during training. We also ran tests where FC1 feature relevance determination was applied at classification time (cf. Section 5.3.2 for more details) for all instances of both fully-supervised and self-supervised learners. We denote such applications of feature relevance at classification time with the abbreviation FRC.

We performed two sets of experiments on all learners using one epoch of training and 100 epochs of training respectively. In all cases of self-supervised learning, two phases of training were used in the input view. In the 1-epoch case, the phase transition occurred half-way through the training, whereas in the 100-epoch case, the phase transition occurred one tenth of the way through training after 10 epochs. Unless otherwise stated, in the first phase, regular SOM training was used using the update rule of Equation (4.7) with an α learning rate linearly decreasing from 0.2 to 0 over the duration of the learning phase. In the second phase, in the case of unsupervised multi-view discriminative learning, training would switch to SOM with a constant learning rate of 0.2, whereas in the self-supervised case, training would switch to one of the LVQ1-based or GLVQ-based self-supervised update rules defined in Section 4.3.2 with a constant learning rate of 0.1. In the case of fully-supervised learning, the regular LVQ1 or GLVQ update rules defined in Section 4.3.1 were used throughout both training phases with a constant learning rate of 0.1. In all cases, SOM training was used in the output view with a learning rate linearly decreasing from 0.2 to 0 over the total training duration. Wherever RLVQ or GRLVQ feature relevance determination was used during training, a constant β learning rate of 0.1 was used (cf. Sections 5.1.1 and 5.1.2). In all cases codebooks in both the input and output views consisted of 49 prototypes arranged in a 7×7 hexagonal lattice with a sheet-shaped topology.

In the results described below we use a particular style of notation to distinguish between different supervised and self-supervised classifiers. For example, the notation $(A \rightarrow B) \Leftrightarrow C$ depicts a bi-modal self-supervised learner where the double-headed arrow $\Leftrightarrow$ indicates cross-view connectivity between two data views where, in the input view, the single-headed arrow $\rightarrow$ indicates a learning phase transition between algorithms A and B, and in the output view, algorithm C presides over the entire learning period. A fully-supervised learner is denoted either simply by A, $A \rightarrow B$ or similar, where the lack of a cross-view connection $\Leftrightarrow$ assumes a corresponding lack of self-supervision. Thus, for example, (SOM $\rightarrow$ LVQ1) $\Leftrightarrow$ SOM denotes unsupervised SOM training (i.e. Equation (4.7)) followed by self-supervised LVQ1 training (i.e. Equation (4.48)) in the input view, with unsupervised SOM training in the output view throughout the training period, whereas LVQ1 by itself would denote fully-supervised LVQ1 training (i.e. Equation (4.41)). Learning phases are explained in more detail in Section 4.3.4 of Chapter 4.

7.2.3.2 Five-Class Synthetic Dual-View Datasets Follow-up Study

In the follow-up study using the five-class datasets, our self-supervised learning vector quantization algorithm (SOM $\rightarrow$ LVQ1) $\Leftrightarrow$ SOM (abbreviated to SS-LVQ in this follow-up study) was compared to the self-supervised self-organizing map (SOM $\rightarrow$ SOM) $\Leftrightarrow$ SOM (abbreviated to SSSOM), as well as variations of both employing feature relevance determination at classification time (SSLVQ (FRC) and SSSOM (FRC)) alongside the supervised algorithms GLVQ, GRLVQ, FC1GLVQ, and, the previously untested supervised relevance neural gas (SRNG) [70], which itself also provides feature relevance estimates. Codebooks in each data view consisted of 49 prototypes arranged in a 7×7 hexagonal lattice with a sheet-shaped topology [86]. In contrast to the previous experiment, this time, in the case of the large dataset of 50,000 samples constant learning rates of $\alpha_t^W = 0.1$, $\alpha_t^V = 0.1$, were used for the prototype update of Equation (4.7) in codebooks W and V respectively in the unsupervised learning phase, and $\alpha = 0.1$ was used for the prototype update of Equation (4.47) in the self-supervised learning phase. The training phase shift from unsupervised to self-supervised took place one tenth of the way through training (after 4,500 samples in 10-fold CV). In the case of the small dataset of 500 samples constant learning rates of $\alpha_t^W = 0.2$, $\alpha_t^V = 0.2$, were used for the prototype update of Equation (4.7) in codebooks W and V respectively in the unsupervised learning phase, and $\alpha = 0.2$ was used for the prototype update of Equation (4.47) in the self-supervised learning phase. In this case, the training phase shift from unsupervised to self-supervised took place one fifth of the way through training (after 90 samples in 10-fold CV).

7.2.4 Results: Three-Class Synthetic Dataset

The results presented here are broken into a number of different sub-sections where we we highlight various different properties of our self-supervised learning framework. First of all we check whether or not class clusters are correctly formed in the output view by unsupervised prototype clustering, then we analyse the unsupervised multi-view discriminative learning of Section 4.2. We follow that by comparing fully-supervised learning with self-supervised learning in Section 7.2.4.5. Later in Section 7.2.4.4 we take a closer look at feature relevance determination in the self-supervised setting, before finally comparing LVQ1-based self-supervision to GLVQ-based self-supervision in Section 7.2.4.5.

7.2.4.1 Unsupervised Class Discovery

An important consideration in evaluating whether or not our framework is capable of self-supervised multi-view learning is to examine if it is capable of successfully finding class clusters in the output view, without which self-supervised discriminative learning in the input view would not be possible. Recall that this is achieved by meta-clustering prototypes in the output view at classification time using k-means clustering (cf. Section 4.2.2.1). But how quickly do the prototypes position themselves such that this meta-clustering may happen successfully? Figure 7.4 answers this question by showing the rate at which ground truth labelled output view test samples fall within output view clusters with matching ground

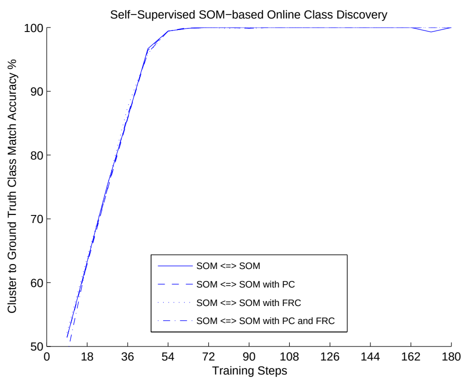
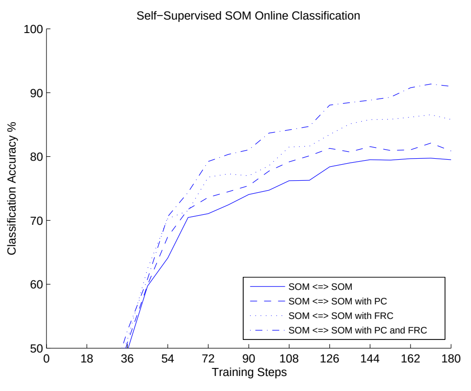


Figure **7.4:** Online class discovery results for various unsupervised SOM-based multi-view learners.

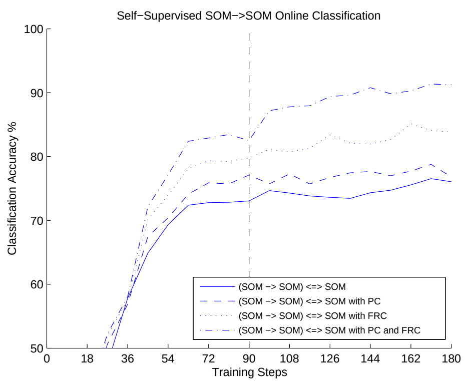
labels (cf. Section 7.2.2) over time. As can be seen from the figure, optimal performance is achieved in the 2-epoch case just over a quarter of the way through the training procedure meaning that, after that point in training, cross-view prediction from input view test samples should at least have the opportunity to reach optimal ground truth prediction rates. In the next section, we examine whether or not this is achieved by unsupervised multi-view discriminative learning.

7.2.4.2 Unsupervised Cross-Modal Discriminative Learning

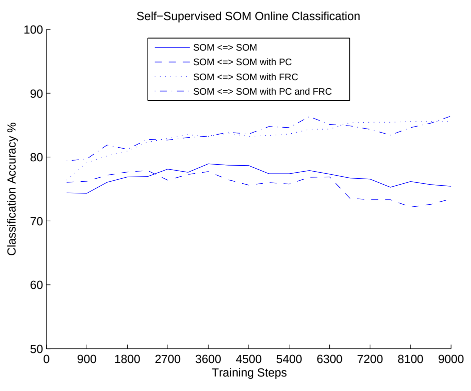
In this section we take a look at relatively simple unsupervised multi-view learning comparisons where the multi-view learners lack an LVQ-based self-supervised discriminative training component. Figure 7.5 shows online classification results for training SOM $\Leftrightarrow$ SOM and (SOM $\rightarrow$ SOM) $\Leftrightarrow$ SOM unsupervised multi-view learners where, in the former case SOM training proceeds with constant α learning rates in both data views ($\alpha = 0.1$ in the input view, $\alpha = 0.2$ in the output view), whereas in the latter case, linearly decreasing α learning rates (0.2 in both cases) are employed in each view before transitioning to a constant α of 0.1 in the input view at a given point during training. Each of these learning templates were tested in their original form, as well as with prototype culling at classification (PC) (cf. Section 4.2.2.4), with feature relevance determination at classification (FRC) (FC1 from 5.2.1), and with a combination of both PC and FRC. Each of these combinations were evaluated over 10 trials of 10-fold cross validation as described above in Section 7.2.2 using random prototype weight



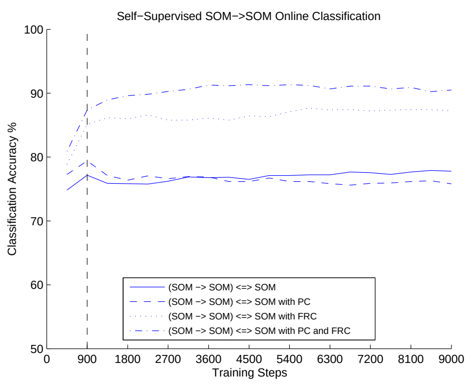
(a) SOM ⇔ SOM, 2 epochs.



(b) (SOM → SOM) ⇔ SOM, 2 epochs.



(c) SOM ⇔ SOM, 100 epochs.



(d) (SOM → SOM) ⇔ SOM, 100 epochs.

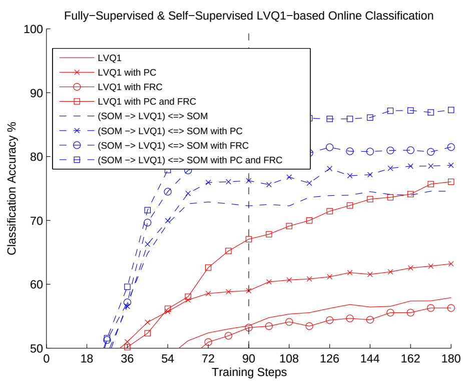
Figure **7.5:** Online classification results for various unsupervised multi-view SOM-based learners. The learners in the left column used SOM training with constant α learning rates in both data views, whereas the learners in the right hand column used linearly decreasing α learning rates in each view before transitioning to a constant α learning rate in the input view at a point during training indicated by the vertical dashed line in the figures. In each figure, learners with various different combinations of prototype culling (PC) and feature relevance determination at classification (FRC) are compared.

initialization over both a short-term training period of 2 epochs and a long-term training period of 100 epochs.

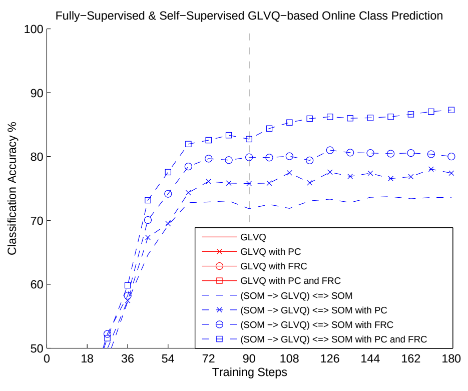
The most striking result from these initial tests is that both prototype culling and feature relevance determination appear to have a significant impact on the performance of the underlying learners. Prototype culling by itself does not appear to enhance results to any great extent, and over longer training periods can even prove to be mildly detrimental (cf. Figures 7.5(c) and 7.5(d)), yet when combined with feature relevance determination, which is clearly effective in its own right, the pair offer quite a noticeable improvement. With regard to to feature relevance determination, the fact that SOM training employs a neighbourhood function that updates many prototypes at each training step is likely to aid FC1 (our first proposed feature relevance algorithm from Section 5.2.1) performance. This is because FC1 takes all codebook prototypes into account when calculating feature relevance. As we shall see later, FC1 can struggle with LVQ-based learning where prototypes are updated one at a time, thus ensuring an increased likelihood of the presence of redundant or harmful prototypes in the codebook. The prototype culling technique helps to guard against this possibility which is why the two techniques work well in unison.

7.2.4.3 Full-Supervision Versus Self-Supervision

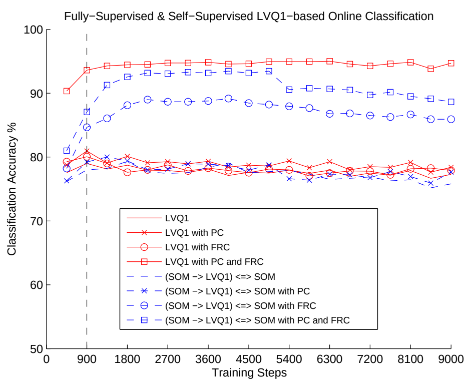
In this section we examine and compare both fully-supervised and self-supervised LVQ-based learners of various varieties. Figure 7.6 juxtaposes results for fully-supervised LVQ1 with those for self-supervised (SOM $\rightarrow$ LVQ1) $\Leftrightarrow$ SOM as well as results for fully-supervised GLVQ with those for self-supervised (SOM $\rightarrow$ LVQ1) $\Leftrightarrow$ SOM in both 2-epoch and 100-epoch runs. Some important effects are discernible from these graphs, the most surprising of which being that self-supervised learning appears to out-perform fully-supervised learning in all but one of them. This was an unanticipated result, but nevertheless belies a logical explanation. In the fully-supervised case, the codebook prototypes are labelled arbitrarily in advance of training, whereas in the self-supervised case, the prototypes in the input view are dynamically labelled at classification time via the use of multi-view information as summarized in Algorithm 4. In effect, this means that it takes more updates and therefore more time for the fully-supervised learners to appropriately adjust the positions of the pre-labelled prototypes to more accurately represent the decision borders. The self-supervised learners, on the other hand, can bypass the adjustment of prototype positions in favour of directly labelling the codebook prototypes with the most appropriate labels based on their current positions and the cross-view perspective of a guiding data view. This problem for the fully-supervised learners is exacerbated by the fact that we randomly initialized the prototype weights in these experiments. It would,



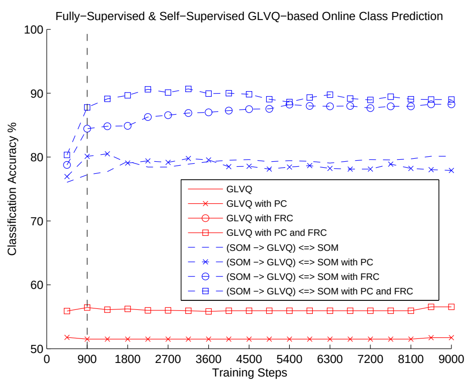
(a) LVQ1-based, 2 epochs.



(b) GLVQ-based, 2 epochs.



(c) LVQ1-based, 100 epochs.

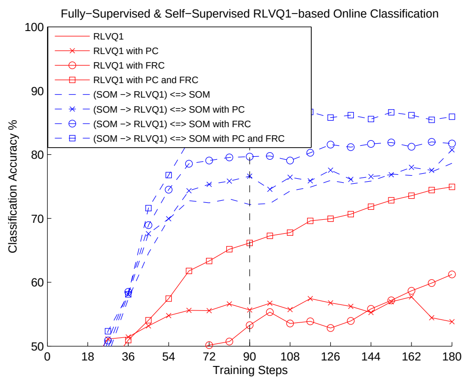


(d) GLVQ-based, 100 epochs.

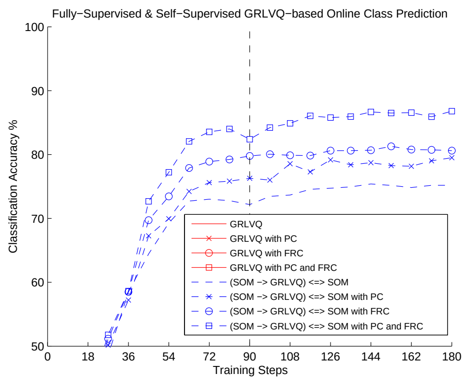
Figure **7.6:** Fully-supervised versus self-supervised LVQ1-based and GLVQ-based learners. LVQ1-based learners are shown in left column, GLVQ-based learners are shown in the right column, and each row shows training for two and 100 epochs respectively. The vertical dashed lines in each figure indicate learning phase shifts in the input data view from unsupervised SOM training to self-supervised LVQ-based training.

of course, be possible to initialize the prototypes more appropriately based on prior knowledge, i.e. by analysing the training set in advance of training, but this would go against the online learning principles we hoped to adhere to as outlined in Chapter 3.

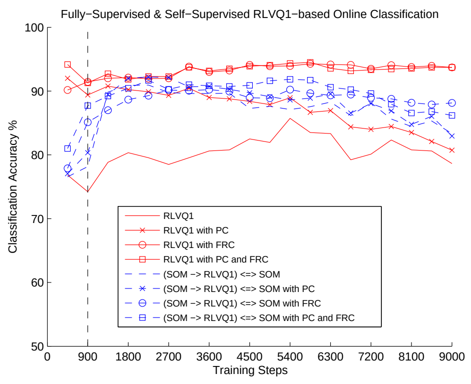
Figures 7.7 and 7.8 show results for using base algorithms that apply feature relevance determination during training; RLVQ1 and GRLVQ-based in the first figure, and FC2LVQ1 and FC2GLVQ-based in the second. These experiments



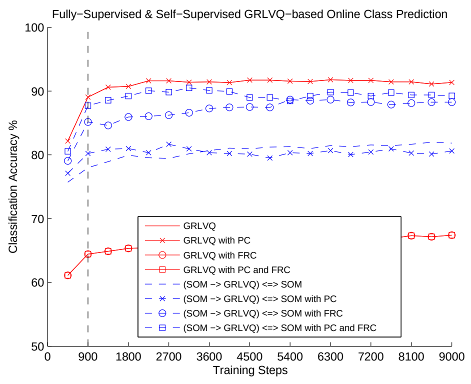
(a) RLVQ1-based, 2 epochs.



(b) GRLVQ-based, 2 epochs.



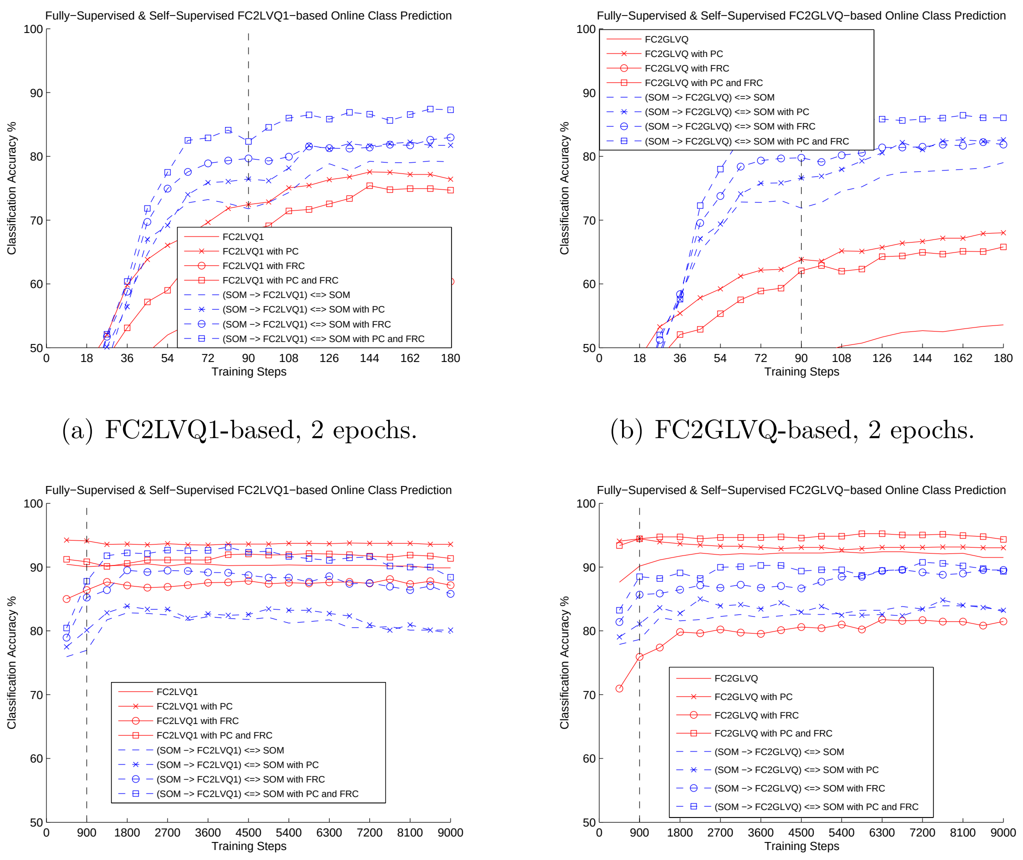
(c) RLVQ1-based, 100 epochs.



(d) GRLVQ-based, 100 epochs.

Figure **7.7:** Fully-supervised versus self-supervised RLVQ1-based and GRLVQ-based learners. RLVQ1-based learners are shown in left column, GRLVQ-based learners are shown in the right column, and each row shows training for two and 100 epochs respectively. Again, the vertical dashed lines in each figure indicate a learning phase shift in the input view.

were run concurrently on the same data as in the previous ones over 2 epochs and 100 epochs on randomly initialized codebook prototypes. We shall return to the issue of using feature relevance determination during training in more detail momentarily in the following section, but for now we can identify some interesting features of these new results as compared to the results shown in Figures 7.5 and 7.6 within the context of comparing full-supervision to self-supervision. It is evident that when feature relevance determination is added into the mix for adaptive feature weighting during training that it helps the fully-supervised GLVQ-based methods to avoid the problems they would otherwise suffer with



(a) FC2LVQ1-based, 2 epochs.

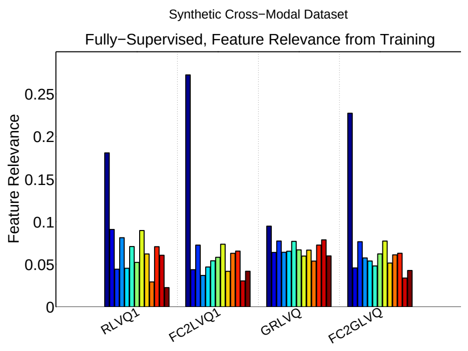
(b) FC2GLVQ-based, 2 epochs.

(c) FC2LVQ1-based, 100 epochs.

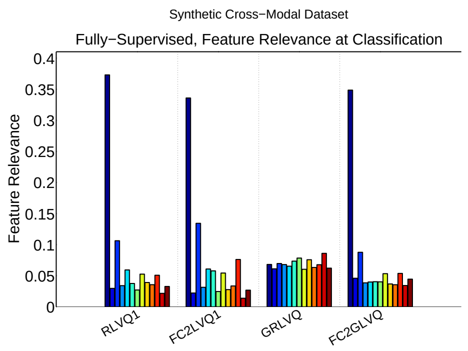
(d) FC2GLVQ-based, 100 epochs.

Figure **7.8:** Fully-supervised versus self-supervised FC2LVQ1-based and FC2GLVQ-based learners. FC2LVQ1-based learners are shown in left column, FC2GLVQ-based learners are shown in the right column, and each row shows training for two and 100 epochs respectively. Again, the vertical dashed lines in each figure indicate a learning phase shift in the input view.

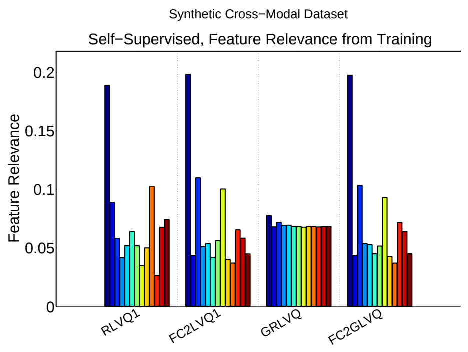
poor prototype initialisation, at least over long-term training periods. This can be seen in Figure 7.7(d) where GRLVQ with prototype culling outperforms the self-supervised methods over 100 epochs and in Figure 7.8(d), where the same is true of FC2GLVQ, FC2GLVQ with PC, and FC2GLVQ with PC and FRC respectively. Although fully-supervised LVQ1-based methods are less vulnerable to this problem, it is clear that when aided by feature relevance determination they do substantially better, particularly when FC2 is employed, allowing them to generally out-perform the self-supervised methods over long training periods, as shown in Figure 7.8(c).



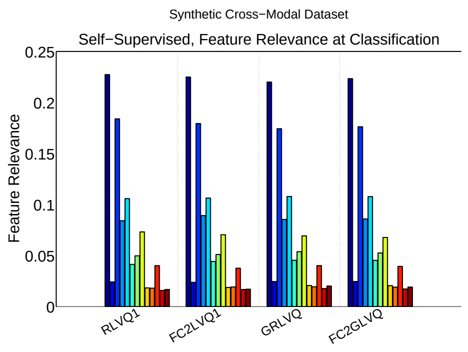
(a) Fully-supervised, feature relevance during training.



(b) Fully-supervised, feature relevance at classification.



(c) Self-supervised, feature relevance during training.

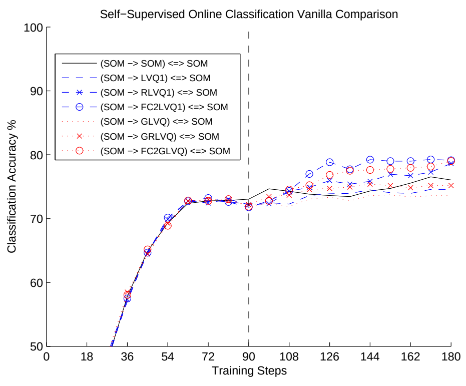


(d) Self-supervised, feature relevance at classification.

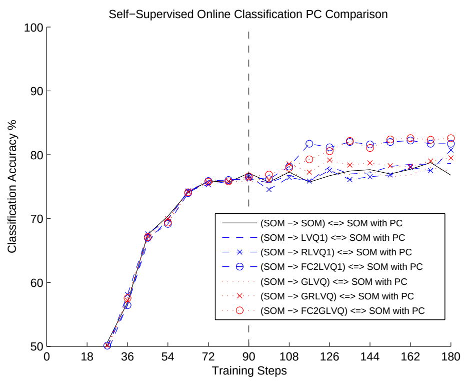
Figure **7.9:** Feature relevance bar plots for both fully-supervised and self-supervised learners acting on the synthetic multi-view dataset. The bar plots show feature weights at the end of training averaged over all folds and trials for 2 epochs of training.

7.2.4.4 Feature Relevance Determination in the Self-Supervised Setting

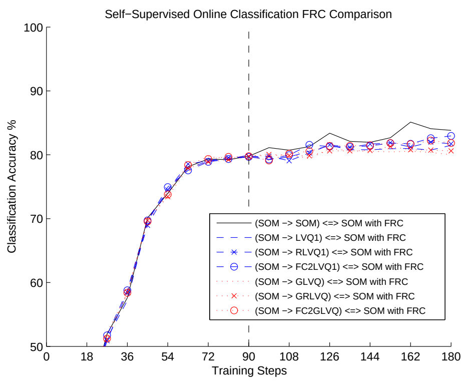
All of the figures mentioned so far also show different combinations of the application of prototype culling at classification time and feature relevance determination at classification time in the guise of FC1 from Chapter 5 as discussed in the algorithm setup of Section 7.2.3. Figure 7.9 shows separate feature weight bar plots for the fully-supervised learners using feature relevance both during training, and at classification time, alongside the same for the self-supervised learners. It is clear from the plots that GRLVQ-based methods do not manage to effectively discriminate the most relevant features, at least not quickly enough over such a short training period. Self-supervised GRLVQ also does not appear to be aided by the dynamic labelling of self-supervision, whereas self-supervised FC2-based methods appear to work well by comparison.



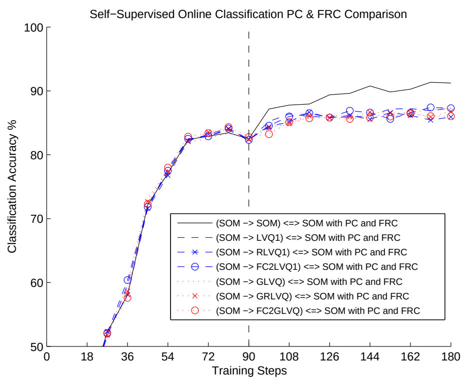
(a) Normal.



(b) With prototype culling (PC).



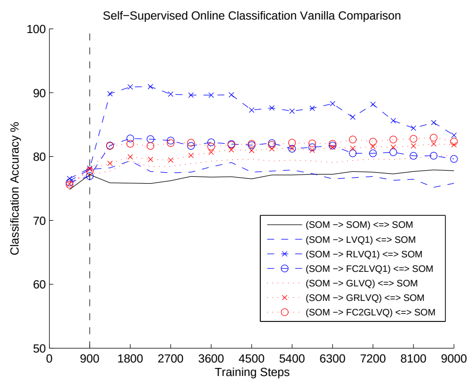
(c) With feature relevance determination at classification time (FRC).



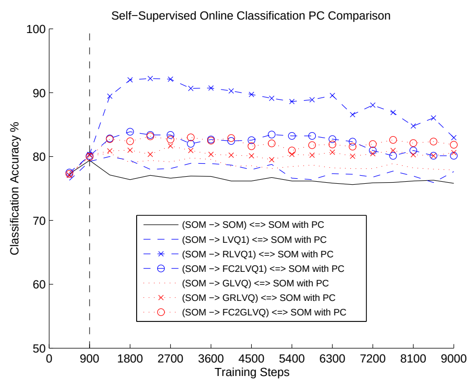
(d) With prototype culling and feature relevance determination at classification time (PC and FRC).

Figure **7.10:** Comparison of multiple self-supervised learners for two epochs of training.

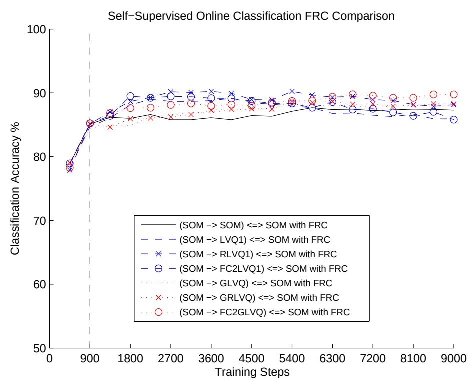
Figures 7.10 and 7.11 show comparisons between all of the self-supervised learners separated into the following four groups: normal, with prototype culling, with feature relevance determination at classification time, and with both prototype culling and feature relevance determination at classification time. From these figures, it becomes evident that the use of FC1 at classification time sta-



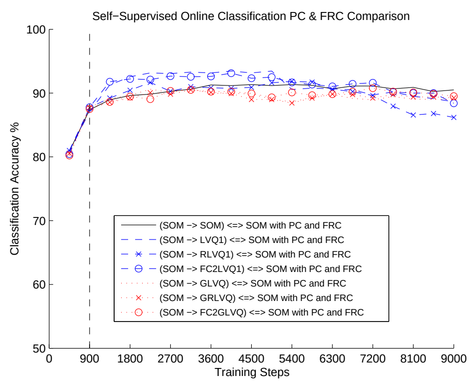
(a) Normal.



(b) With prototype culling (PC).



(c) With feature relevance determination at classification time (FRC).



(d) With prototype culling and feature relevance determination at classification time (PC and FRC).

Figure **7.11:** Comparison of multiple self-supervised learners for 100 epochs of training.

bilizes performance across multiple self-supervised classifiers over time, bringing their performance into alignment with one another. This is reflected in the feature weight bar plot of Figure 7.9(d) which shows almost identical feature weighting provided by FC1 to each of the self-supervised learners in the 2-epoch case. We can also see from Figures 7.10(a) and 7.10(b) that FC2-based feature relevance tends to fare better under self-supervision than its RLVQ or GRLVQ-based counterparts, at least in the shorter term, though it must be noted that this could also be dependent on the selection of learning rates for RLVQ and GRLVQ. Interestingly, from these graphs it also becomes clear that the best performer of all in short-term training is (SOM $\rightarrow$ SOM) $\Leftrightarrow$ SOM with PC and FRC, which lacks any LVQ-based self-supervised refinement. We would attribute this to the neighbourhood function of SOM providing regular updates to a broader number of prototypes as opposed to the single prototype updates of LVQ-based methods. This is likely to aid the performance of FC1, which relies on the prototypes approximating the data distribution to function optimally.

7.2.4.5 LVQ1-based training versus GLVQ-based training

The most noticeable difference between LVQ1-based training and GLVQ-based training is that while LVQ1-based methods can achieve better results faster, they tend to be less stable over time, whereas GLVQ-based methods are more reliable in this sense. Each of the Figures 7.6(c), 7.7(c), and 7.8(c) show divergent behaviour for LVQ1-based methods over 100 epochs, particularly in the self-supervised case. Such divergence is less evident in the GLVQ-based learners running over the same period in Figures 7.6(d), 7.7(d), and 7.8(d). Over shorter-term training periods, however, it is clear that LVQ1-based methods may be a more appropriate choice, as demonstrated in the comparison between Figures 7.6(a), 7.7(a), and 7.8(a), and Figures 7.6(b), 7.7(b), and 7.8(b). This observation is clear for the fully-supervised learners, but less-so for the self-supervised learners, which tend to be more agnostic to the underlying LVQ method, as evidenced in Figures 7.10 and 7.11, although self-supervised RLVQ1 does appear to provide an anomaly here with unexpectedly dominant performance over the medium term before diverging.

7.2.5 Results: Five-Class Synthetic Datasets Follow-up Study

In the experiments using the five-class datasets, we wished to test our learners in a more complex problem setting, in part in order to analyse whether or not the LVQ-based self-supervised discriminative methods might offer more of an advantage over the SOM-based unsupervised methods than had been evidenced previously. Here, we analyse the results of class discovery, class prediction and feature relevance determination in Sections 7.2.5.1, 7.2.5.2 and 7.2.5.3 respectively.

7.2.5.1 Class Discovery

What is perhaps most interesting about the class discovery graphs in Figure 7.12 is the fact that there is such a similarity between the left graph (small 500 sample dataset) and the right graph (large 50,000 sample dataset). We might have expected class discovery results to converge much faster in the latter case, but the improvement appears meagre when compared to the former case. We suspect this may be due to similarity in the rate of reduction of the SOM neighbourhood radius between cases, but this remains unproven. Nevertheless, both cases provide reasonable results by the end of training such that class prediction learning may occur, thus we take a closer look at this aspect next.

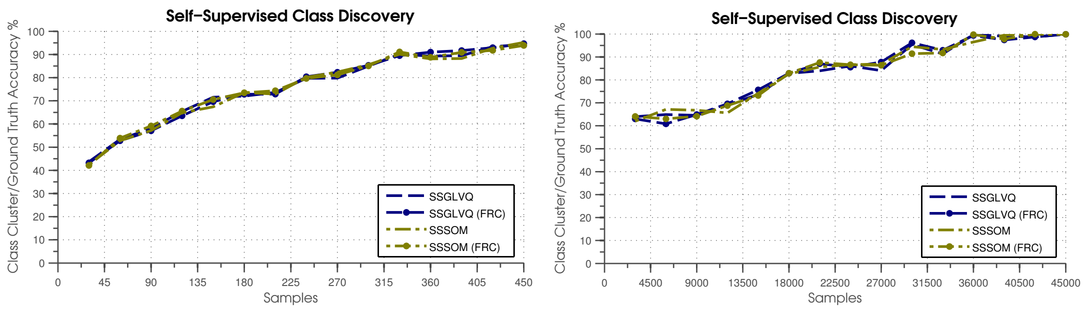


Figure **7.12:** Class discovery results for 10-folds cross validation on the 500 sample 5-class synthetic dataset (left graph) and for 5-folds cross validation on the 50000 sample 5-class synthetic dataset (right graph).

7.2.5.2 Class Prediction

The main result worth noting in the class prediction graphs of Figure 7.13 is that our initial hypothesis turned out to be correct: given a more complex categorisation scenario, in this case with five categories, the LVQ-based self-supervised discriminative methods do indeed out-perform their unsupervised SOM-based counterparts, at least over long training durations, as evidenced by the right graph of Figure 7.13. Moreover, this result appears to be independent of whether or not feature relevance is applied at classification time: plain SSGLVQ proves to be better than plain SSSOM, and SSGLVQ (FRC) outperforms SSSOM (FRC) by as much as 10% by the end of training. The self-supervised methods do not compete as well with the supervised methods in this instance over such long training periods, but do appear to hold their own against GRLVQ, for example, over the shorter period (left graph). FC1GLVQ also evidently performs well here.

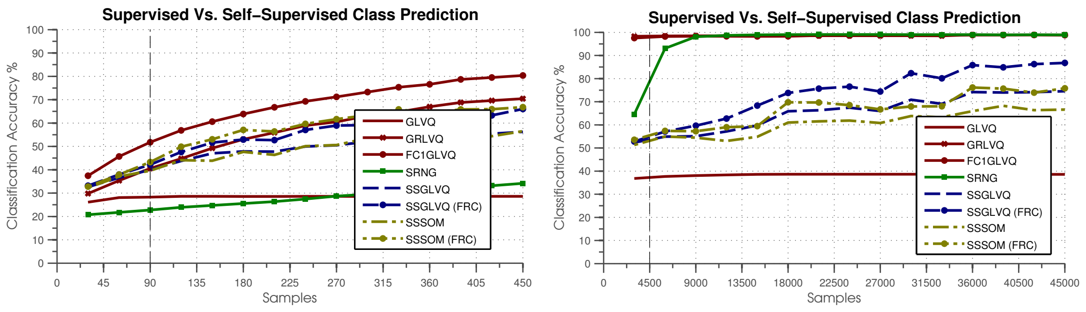


Figure **7.13:** Class prediction results for 10-folds cross validation on the 500 sample 5-class synthetic dataset (left graph) and for 5-folds cross validation on the 50000 sample 5-class synthetic dataset (right graph).

7.2.5.3 Feature Relevance Determination

With regard to feature relevance determination, the left and right graphs of Figure 7.14 show bar plot comparisons of feature weighting at the end of training for GRLVQ SRNG, FC1GLVQ, SSSOM (FRC) and SSGLVQ (FRC) after training with the small dataset and the large dataset respectively. It is evident from these graphs that the feature relevance determination method of Section 5.2.1, when applied at classification time, allows for rapid estimation of feature relevance when applied to SSGLVQ, correctly identifying the first feature dimension in the input modality to be the most important, giving generally progressively lower weighting from the third feature dimension onwards which were generated by progressively adding noise to the first dimension, and correctly giving one of the lowest weights to the second feature which is uninformative for class discrimination.

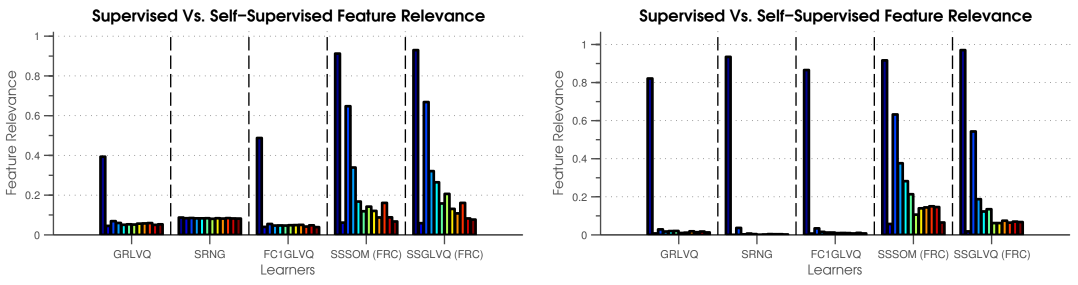


Figure **7.14:** Feature relevance determination results for 10-folds cross validation on the 500 sample 5-class synthetic dataset (left graph) and for 5-folds cross validation on the 50000 sample 5-class synthetic dataset (right graph).

7.2.6 Summary

After analysing the above experiments, we may conclude that our proposed self-supervised multi-view learning framework should function as we require it to for self-supervised object affordance learning. Working with quite noisy data in the guise of the synthetic multi-view dataset presented above in Section 7.2.1, the multi-view learners managed to correctly meta-cluster the classes present in the output view relatively rapidly, an important first step towards cross-view class prediction. In Section 7.2.4.2, in examining class prediction from unsupervised multi-view discriminative learning, we ascertained that both prototype culling and feature relevance determination can offer substantial boosts in performance.

After that, in Section 7.2.4.3 we demonstrated an unanticipated result showing that when comparing fully-supervised and self-supervised LVQ-based learning, the additional information provided by self-supervision from an additional data view can actually provide better performance than fully-supervised learning with ground truth labels, at least over short-term training periods and under the conditions defined in this thesis (cf. Chapter 3). In Section 7.2.4.4 we showed how the application of FC1-based feature relevance determination (cf. Chapter 5) at classification time, which exploits cross-view dynamic prototype labelling, tends to have a stabilizing effect on all of the self-supervised learners tested, bringing their respective performances into alignment with one another. We discussed how LVQ1-based learners, though less stable than GLVQ-based learners over longer training periods, may be more appropriate over shorter terms, offering better results, faster.

Perhaps the most interesting results however, came in the follow-up study reported in Section 7.2.5. There it was shown that when a more complex class distribution is involved, our GLVQ-based self-supervised discriminative method does indeed offer significant improvement over the SOM-based unsupervised multi-view alternative, at least over long enough training periods. This goes some way towards validating our approach in this thesis.

7.3 Object Push Affordance Experiments with Katana/Camera Setup

To conclude our experimental results chapter, we present, in the final two sections, experiments that were performed using real-world data gathered from the robotic systems described in Chapter 6, in which the systems were tasked with pushing household objects in their experimental environments using their robotic arm manipulators and learning about the affordances of the objects through visual observation and self-supervised learning. In the first of these sections, we describe experiments where the Katana/Camera setup was used to gather a dataset by performing learning trials involving objects and pushes that resulted in two different affordance ground-truth effect classes being produced: rolling and non-rolling classes.

Various learning trials were performed and comparisons were made between our proposed self-supervised algorithms and well-known supervised vector quantization algorithms as in the previous sections. These were divided into a number of sub-experiments, using different combinations of the feature sets described in Sections 6.2.1 and 6.2.2 depending on the particular platform being used and the goals of the experiment. In the following sub-sections, we describe the dataset that was gathered in Section 7.3.1, an evaluation procedure that was used to test the abilities of the algorithms to generalize to novel objects in Section 7.3.2, the experiments that were performed in Section 7.3.3, and finally, the results of the experiments in Sections 7.3.4, 7.3.5 and 7.3.6.

7.3.1 Data

To test our affordance learning system with the Katana/Camera platform, the experimental environment was set up as previously described in both the introduction to the thesis and in Chapter 6 and as shown in Figures 1.2 and 6.2. During experiments, objects were placed at a fixed starting position prior to interaction. The two camera systems were used to provide both sufficiently detailed close-up range data of the object surfaces and a sufficiently wide field of view to capture object motion over the entire work area. To achieve this, the stereo camera was positioned above the start position, while the monocular camera was positioned at a higher position in front of the workspace giving both cameras a top-down viewpoint of the work surface.

We selected eight household objects (cf. Figure 7.15) for the experiments: four flat-surfaced objects; a book, a CD box, a box of tea and a drink carton, and four curved-surfaced objects; a box of cleaning wipes, a Pepsi® can a Sprite® can and a tennis ball box. A dataset was gathered consisting of 20 object push

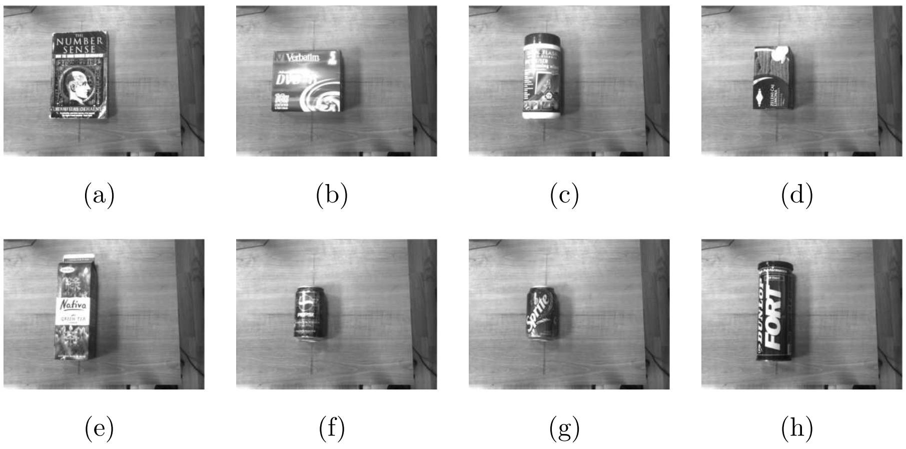


Figure **7.15:** Sample rolling versus non-rolling objects as seen by the Bumblebee 2 grayscale camera: 7.15(a) Book (non-rolling), 7.15(b) CD Box (non-rolling), 7.15(c) Cleaning wipes box (rolling), 7.15(d) Green tea box (non-rolling), 7.15(e) Green tea carton (non-rolling), 7.15(f) Pepsi can (rolling), 7.15(g) Sprite can (rolling), 7.15(h) Tennis ball box (rolling).

tests for each of the eight objects and the resulting data was processed, leaving 160 data samples. A projection of this data onto the dimensions of curvature features O_3^b and O_3^b of the input object feature data view is shown in Figure 7.16. Each of these objects was placed centred at the start position with a consistent orientation, and the robotic arm pushed the object at a fixed speed using a fixed pushing action. During tests, the curved objects would tend to roll after being pushed, whereas the flat objects would stop suddenly, so the samples were then hand-labelled with two ground truth labels: rolling and non-rolling. Some sample interactions with both rolling and non-rolling objects are shown in Figure 7.17. As before, the ground-truth labels were not used to train the self-supervised learners, but were required for the performance evaluation. Before an action was performed on an object, both intensity and range images were gathered from the stereo camera. After an action was performed on an object, images

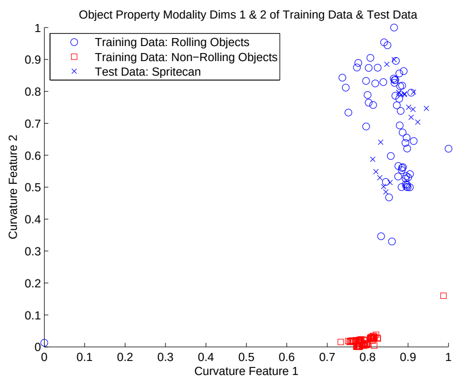


Figure **7.16:** A projection of object feature training data from the Katana/Camera experiment onto two of the feature dimensions, specifically the 3-D curvature features O^b_3 (x-axis) and O^b_4 (y-axis), along with a projection of test data for the Sprite can object. The data are well-separated along the second curvature feature dimension so, all other things being equal, it should be possible to construct a reliable classifier if it places appropriate emphasis on this dimension.

were gathered and passed to the tracking system described in Section 6.2.2 This data was processed to produce the visual shape features $\{O^a_1, \ldots, O^a_9\}$ and 3-D shape features $\{O^b_1, \ldots, O^b_{36}\}$ (cf. Sections 6.2.1.2 & 6.2.1.3), as well as the global motion effect features $\{E^a_1, \ldots, E^a_9\}$ and the local appearance change features $\{E^d_1, \ldots, E^d_3\}$ (cf. Sections 6.2.2.3 & 6.2.2.5).

7.3.2 Evaluation Procedure

In the following, a modified form of leave-one-out cross validation, which we call leave-one-object-out cross validation (LOOOCV) was employed. In order to evaluate our learning approach. Given an $\{X, Y\}$ dataset of features extracted from interactions with various objects from a given experiment, LOOOCV involved splitting the dataset into a test set consisting of all of the samples for a given object, and a training set of all of the samples for the remaining objects. The learning task was then to train learners using the training set, find the affordance classes in the output view and try to classify the test set samples of the left-out object on that basis. Cross validation was performed by using each of the objects in the full dataset in turn as the test object to form multiple test and training sets, performing the learning task using each of these, and averaging class discovery and class prediction scores across all of them. Performing the evaluation

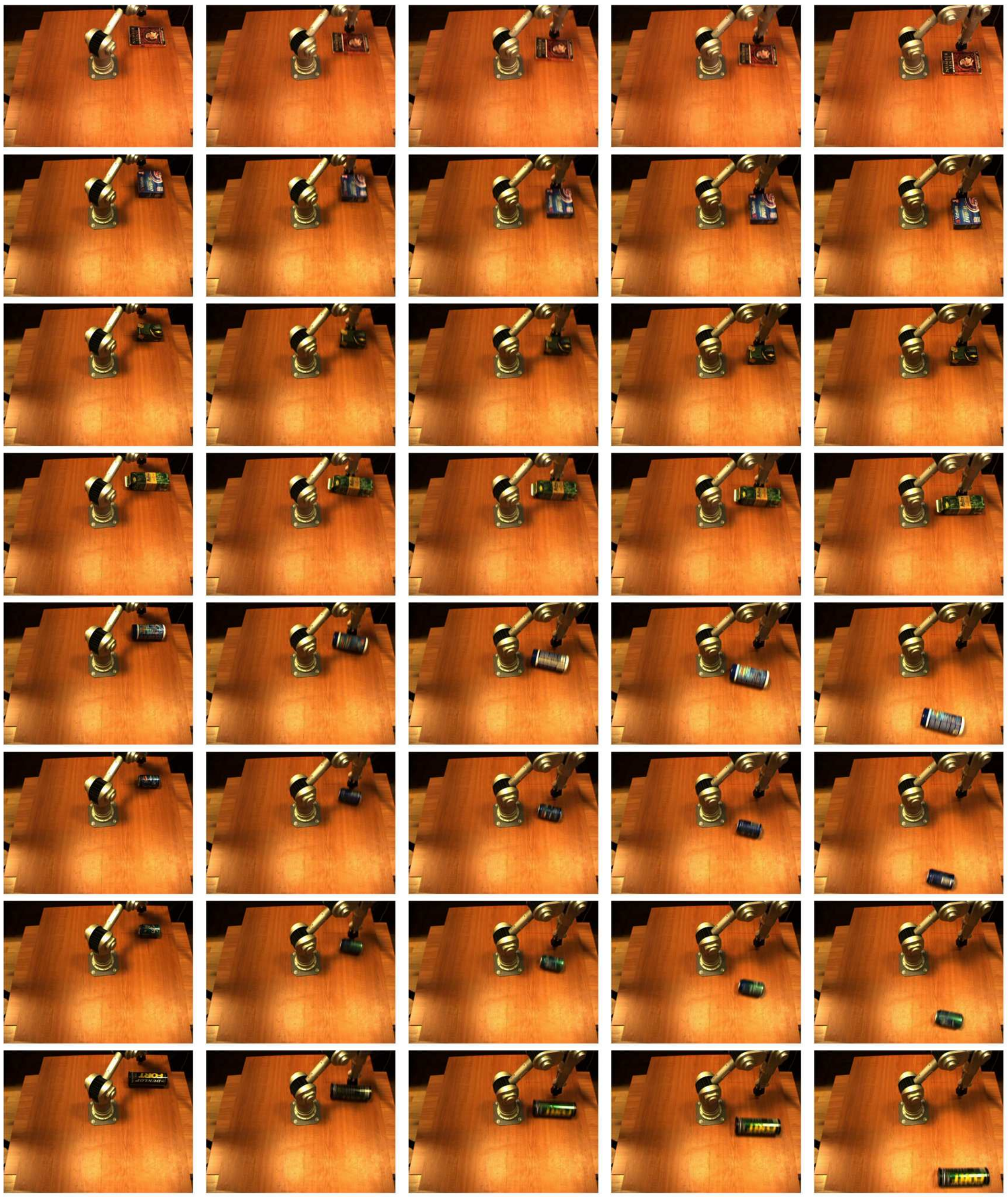


Figure **7.17:** Sample interactions as seen from the Flea camera of the test objects described in Section 7.3.1 being interacted with. From top row to bottom: a book, a CD box, a box of tea, a drink carton, a box of cleaning wipes, a Pepsi can, a Sprite can, and a tennis ball box. The first four objects tend to slide, while the last four tend to roll.

in this way, using LOOOCV, allowed us to test the performance of affordance prediction for novel objects that the algorithms do not encounter during training, a more stringent evaluation than would otherwise be provided by LOOCV or k-folds cross validation.

7.3.3 Experiments

In the first and second of three sub-experiments, the details of which are described below, the features $\{O^a_1, \ldots, O^a_9\}$ were paired with the two 3-D global object curvature features $\{O^b_3, O^b_4\}$ to produce input view space $X^{a,b}$ consisting of vectors of the form $\mathbf{x} = \{O^a_1, \ldots, O^a_9, O^b_3, O^b_4\}^T$, while in the third, vectors of the form $\mathbf{x} = \{O^b_1, \ldots, O^b_{36}\}^T$ using all of the 3-D shape features formed the basis for input view space X^b. In all three sub-experiments, both the global motion effect features and the local appearance change effect features were grouped and defined the output view space $Y^{a,d}$ composed of vectors of the form $\mathbf{y} = \{E^a_1, \ldots, E^a_9, E^d_1, \ldots, E^d_3\}^T$. The first experiment was designed to test the efficacy of the algorithms in a situation where where only some of the features are useful, as was likely to be the case for the small sub-set of 3-D features paired with a majority of less relevant 2-D features. The second experiment was performed as part of a follow-up study published in [145], and involved testing against a previously untested supervised classifier, a shorter training period, and tweaks to the algorithm parameters. The third experiment, which also formed part of the follow-up study, was designed to test the algorithms with a full compliment of parts-based 3-D features, perhaps many of which would be useful for affordance prediction, and perhaps many of which would, in turn, be revealed by the feature relevance determination mechanisms involved.

7.3.3.1 3-D + 2-D Object Features Dataset

In this experiment, both the fully-supervised and self-supervised algorithms were set up identically to what was described in Section 7.2.3 for the synthetic multi-view dataset, the only differences being that here, firstly, LOOOCV was used instead of k-fold cross validation, and secondly, rather than doing two experimental runs over one epoch of training and 100 epochs of training respectively, we performed experimental runs over two training epochs with a learning phase transition half-way through training (after one epoch). As before, the feature weights of the codebook prototype vectors were randomly initialized to test the abilities of the algorithms to learn from scratch. LOOOCV was therefore performed in 10 trials and results were averaged in order to account for the variation in codebook initialization between trials.

7.3.3.2 3-D + 2-D Object Features Dataset Follow-Up Study

In the follow-up study [145] using the same feature set, our self-supervised learning vector quantization algorithm (SOM $\rightarrow$ LVQ1) $\Leftrightarrow$ SOM (here abbreviated to SSLVQ) was compared to the self-supervised self-organizing map (SOM $\rightarrow$ SOM) $\Leftrightarrow$ SOM (here abbreviated to SSSOM), as well as variations of both employing feature relevance determination at classification time (SSLVQ (FRC) and SSSOM (FRC)) alongside the supervised algorithms GLVQ, GRLVQ and, the previously untested supervised relevance neural gas (SRNG) [70], which itself also provides feature relevance estimates. Codebooks in each data view consisted of 49 prototypes arranged in a 7×7 hexagonal lattice with a sheet-shaped topology [86]. In contrast to the previous experiment, this time, constant learning rates of $\alpha_t^W = 0.1$, $\alpha_t^V = 0.1$, were used for the prototype update of Equation (4.7) in codebooks W and V respectively in the unsupervised learning phase, and $\alpha = 0.1$ was used for the prototype update of Equation (4.47) in the self-supervised learning phase. In addition to that, since the rapid rate of class discovery in the previous experiment had proved promising, this time, training in experimental runs was performed over a single epoch (a single run through training data). Learning phases were switched from unsupervised to self-supervised halfway through training.

7.3.3.3 3-D Object Features Dataset Follow-Up Study

As part of the same follow-up study [145], we performed an experiment using the full set of 3-D object features $\mathbf{x} = \{O_1^b, \ldots, O_{36}^b\}^T$ in the input data view alongside the co-occurring features $\mathbf{y} = \{E_1^a, \ldots, E_9^a, E_1^d, \ldots, E_3^d\}^T$ in the output view. As in the previous experiment in Section 7.3.3.2, the multi-view self-supervised algorithms SSSOM, SSLVQ, SSSOM (FRC) and SSLVQ (FRC) were compared with the mono-view supervised algorithms GLVQ, GRLVQ and SRNG, and the experimental and algorithmic conditions also remained as before. In the following three sub-sections we report the results from these experiments.

7.3.4 Results: 3-D + 2-D Object Features Dataset

In this experiment, the self-supervised learners were trained using data from $\{X^{a,b}, Y^{a,d}\}$, while the supervised learners were trained with $\{X^{a,b}, L_{GT}(X^{a,b})\}$, where L_{GT} applies ground-truth labels to the input view samples. Online evaluation and comparison of the learners was performed using LOOOCV over two epochs of training. What follows here is a detailed discussion, similar to that of Section 7.2.4, of the performance properties of the various self-supervised multi-view learner variations as applied to this dataset. For a more high-level discussion, see the results of the follow-up study in the succeeding Section 7.3.5.

7.3.4.1 Affordance Class Discovery

As can be seen from Figure 7.18, the meta-clustering of object affordance classes in the output object effects view occurs very rapidly indeed in this instance, reaching optimal ground truth matching performance almost immediately (tests were performed every 20 training steps for the figures in this section). Compared

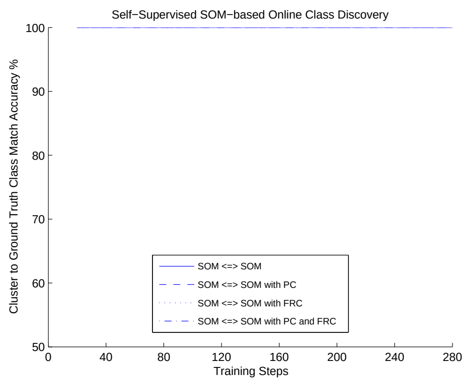
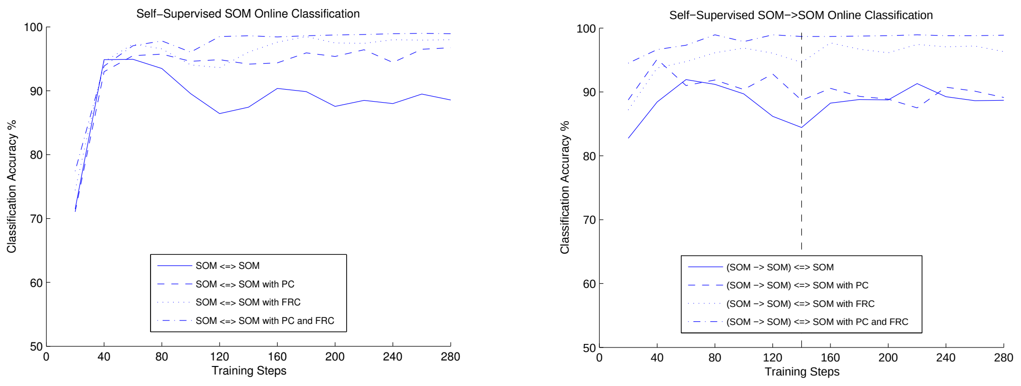


Figure **7.18:** Online affordance class discovery results for unsupervised multi-view SOM-based learners.

to the synthetic dataset of the previous section where there were three ground truth output view classes and a lot of noise in the additional feature dimensions, in the object affordance learning dataset presented here, there are two classes and less noise present, which may make the unsupervised class discovery task easier. As such, it may also be possible for cross-view class prediction from the input object features view to reach good performance levels more rapidly also, which is what we examine next.

7.3.4.2 Affordance Class Prediction

Looking first at the case of unsupervised multi-view discriminative learning (again, cf. Section 4.2), Figure 7.19 shows the basic unsupervised SOM $\Leftrightarrow$ SOM and (SOM $\rightarrow$ SOM) $\Leftrightarrow$ SOM algorithms performing reasonably well initially, but proving unstable as training progresses, perhaps as more diverse object samples are encountered. However, the addition of both prototype culling and feature

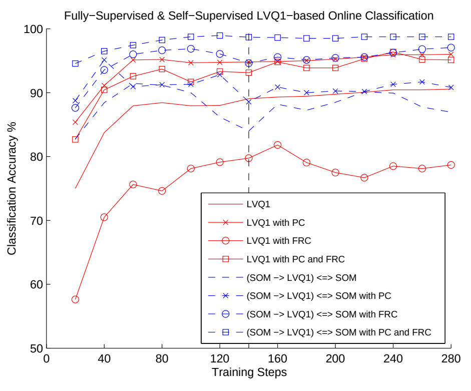


(a) SOM $\Leftrightarrow$ SOM, 2 epochs. (b) (SOM $\rightarrow$ SOM) $\Leftrightarrow$ SOM, 2 epochs.

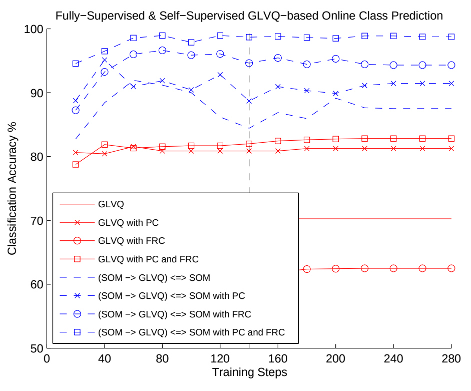
Figure **7.19:** Online object affordance prediction of unsupervised multi-view SOM-based learners over two training epochs.

relevance determination at classification time boosts performance significantly to the point where the learners reach close to optimal predictive performance roughly mid-way through training. It should be reiterated here that these evaluations are for predicting the affordance classes of novel objects that have not been encountered during training.

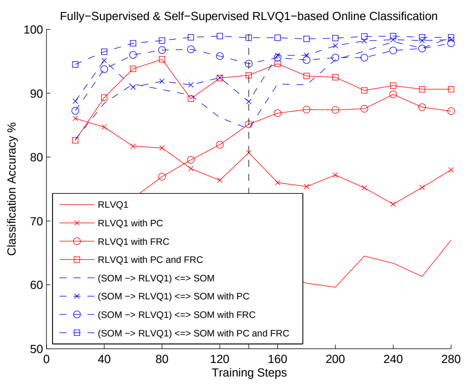
Turning to self-supervised multi-view discriminative learning, Figure 7.20 compares self-supervised learning for each of the base update types to their fully-supervised counterparts. Once again it is evident that the self-supervised learners outperform the fully-supervised learners over this short-term 2 epoch training period in almost all cases, FC2LVQ1 with prototype culling being the one exception. Why this happens is not clear from such prediction result plots, but begins to make sense when we look more closely at the underlying data.



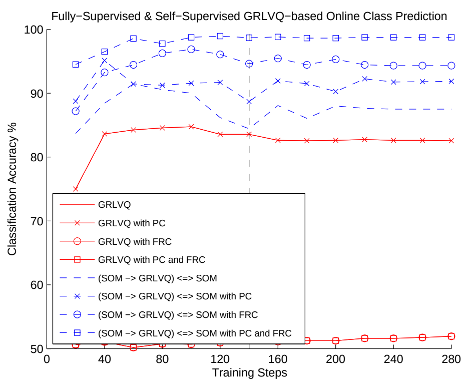
(a) LVQ1-based, 2 epochs.



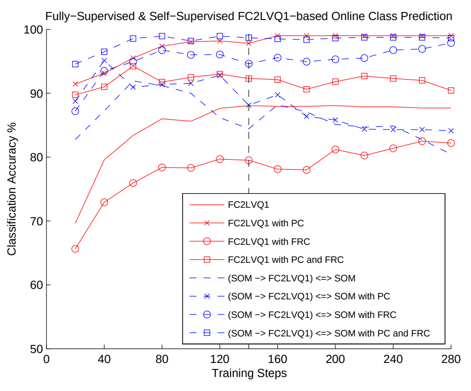
(b) GLVQ-based, 2 epochs.



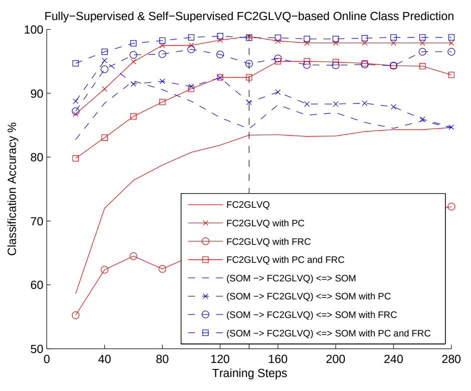
(c) RLVQ1-based, 2 epochs.



(d) GRLVQ-based, 2 epochs.



(e) FC2LVQ1-based, 2 epochs.



(f) FC2GLVQ-based, 2 epochs.

Figure **7.20:** Online object affordance prediction of fully-supervised and self-supervised learners over two training epochs. Left column: LVQ1-based learners. Right column: GLVQ-based learners. Top row: no feature relevance during training. Middle row: RLVQ1 and GRLVQ-based feature relevance during training. Bottom row: FC2-based feature relevance during training.

Figure 7.16 shows data projections along the O_3^b and O_4^b curvature feature dimensions of the input object feature data view for both training data and test data when the test object is the Sprite can. These two 3-D object curvature features are the two most important features in the data for predicting rolling versus non-rolling objects, and the figure shows good separability in the data with respect to the O_3^b feature (Curvature Feature 2 in Figure 7.16). Figures 7.21 and 7.22 show similar plots, but instead of projecting the training data, they show the input view codebook prototypes for both fully-supervised and self-supervised learners respectively projected along the feature dimensions. The potential importance of prototype initialisation, dynamic labelling and prototype culling is evident from these figures. In Figure 7.21, for example, the prototypes are initialized randomly, and many of them become redundant during training and clutter the feature space. Prototype culling greatly improves this situation. In Figure 7.22, the initial bout of unsupervised SOM training clusters the prototypes over the data distribution and prototype clustering further removes those which are discriminatively redundant.

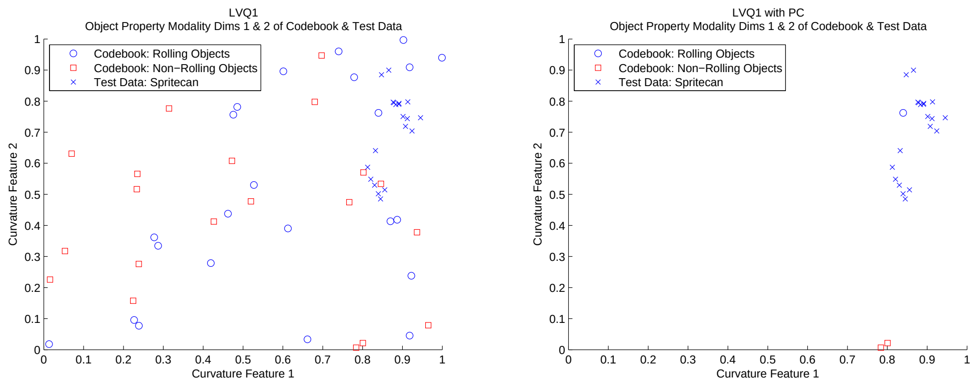


Figure **7.21:** The effect of prototype culling (cf. Section 4.2.2.4) after training with the LVQ1 classifier. Each figure shows a projection of both the codebook prototype vectors as well as test data the 3-D curvature feature dimensions O_3^b (y-axis) and O_4^b (x-axis) in the object feature view. In the left figure, prototype culling has not been applied, whereas in the right figure, it has been.

Another question that arises is, does the addition of LVQ-based self-supervision offer any improvement over unsupervised multi-view discriminative learning? To answer that question, we compare unsupervised multi-view discriminative learning to self-supervised multi-view discriminative learning in isolation in Figures 7.23 and Figures 7.24. Here, it would appear from Figure 7.23(a)

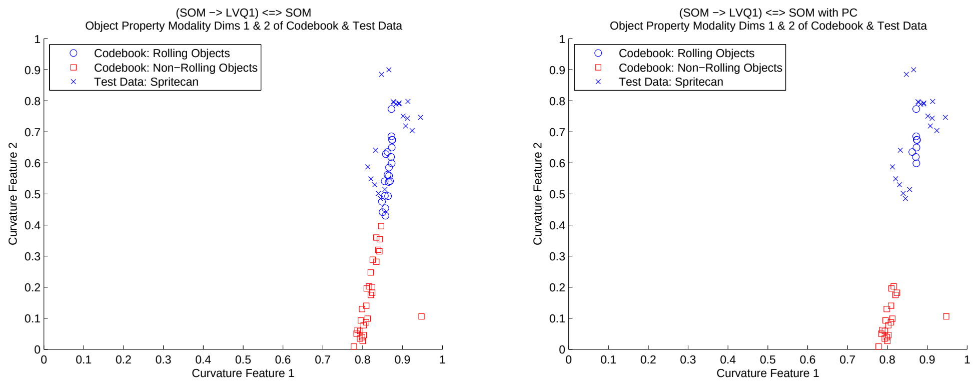
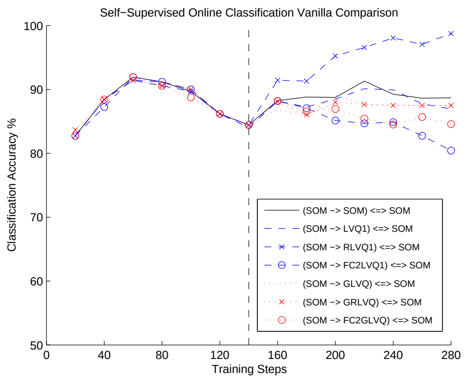
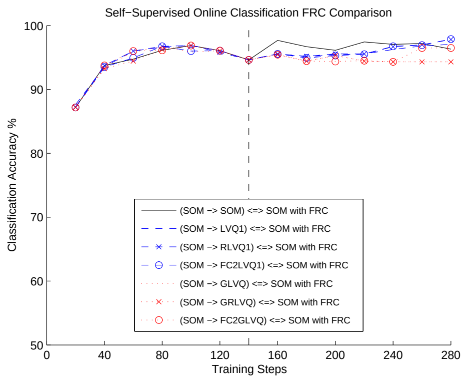


Figure **7.22:** The effect of prototype culling after training with the (SOM → LVQ1) ⇔ SOM self-supervised classifier. Again, in the left figure, prototype culling has not been applied, whereas in the right figure, it has been.



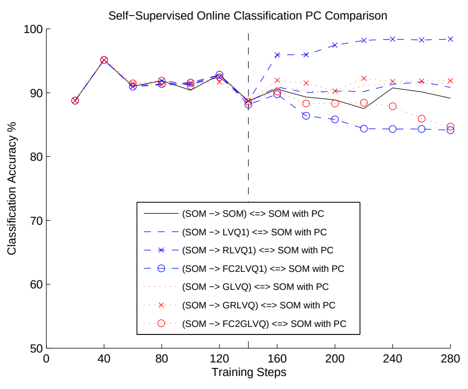
(a) Normal.



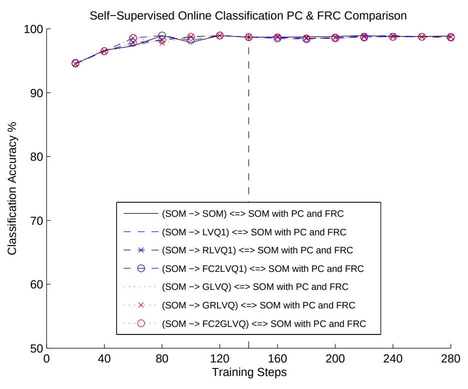
(b) With feature relevance determination at classification time (FRC).

Figure **7.23:** Comparing unsupervised and self-supervised learners, with and without feature relevance determination at classification time.

that discounting prototype culling or any form of feature relevance determination, the unsupervised (SOM → SOM) ⇔ SOM actually does marginally better than the self-supervised (SOM → LVQ1) ⇔ SOM or (SOM → GLVQ) ⇔ SOM. This situation changes when prototype culling is added, boosting self-supervision results slightly beyond those of the unsupervised learner, as can be seen in Figure 7.24(a). When feature relevance determination during training is considered, RLVQ1-based self-supervision appears to offer great improvement over its LVQ1-based counterparts. And once feature relevance is introduced at classification



(a) With prototype culling (PC).



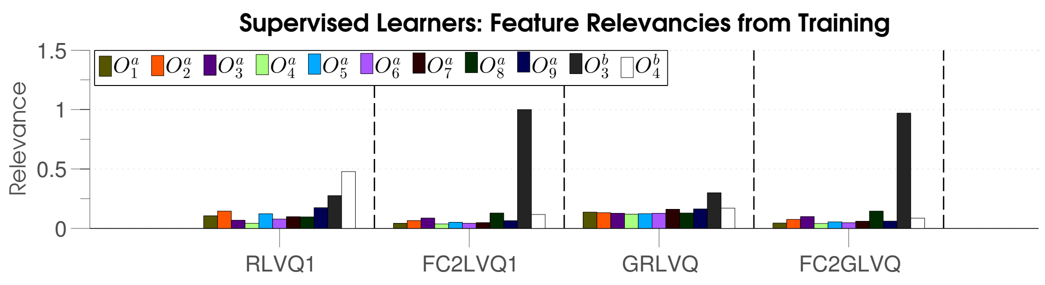
(b) With prototype culling and feature relevance determination at classification time (FRC).

Figure **7.24:** Comparing unsupervised and self-supervised learners with prototype culling, with and without feature relevance determination at classification time.

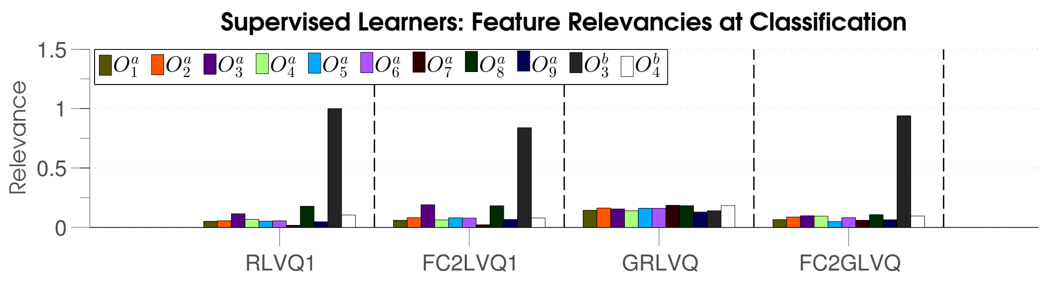
time, all self-supervised learners tend to improve dramatically, as was the case in the previous section with the synthetic multi-view dataset.

7.3.4.3 Feature Relevance Determination

The bar plots of Figures 7.25 and 7.26 show average feature relevancies for various fully-supervised and self-supervised learners both from training and at classification time. In the fully-supervised case, we can observe that both RLVQ1 and GRLVQ do not do as well as the FC2-based methods during training, a fact that can be correlated with their respective prediction results in the graphs of Figure 7.23. From here, we can also see why the RLVQ1-based self-supervised learning does so well at prediction in the results of Figure 7.23(a) when no augmentations are considered compared to the other self-supervised learners. Figure 7.26(a) shows that it finds an appropriate feature weighting during training whereas the FC2-based methods do not do as good a job. However, as seen in Figure 7.26(b), this is resolved by applying feature relevance determination at classification which provides appropriate feature weighting across all of the learners, leading to the uniformly good predictive results shown in Figures 7.23(b) and 7.24(b).



(a) Feature relevance from training.



(b) Feature relevance at classification.

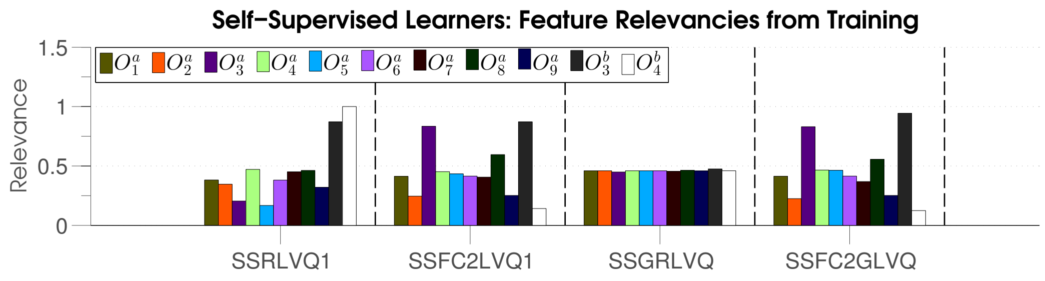
Figure **7.25:** Feature relevance bar plots for fully-supervised learners showing feature weights, both from in-built feature relevance mechanisms, and from the FC1 feature relevance method being applied at classification time, averaged over all folds and trials. Vertical dashed lines separate bar plots for different learners.

7.3.5 Results: 3-D + 2-D Object Features Dataset Follow-Up Study

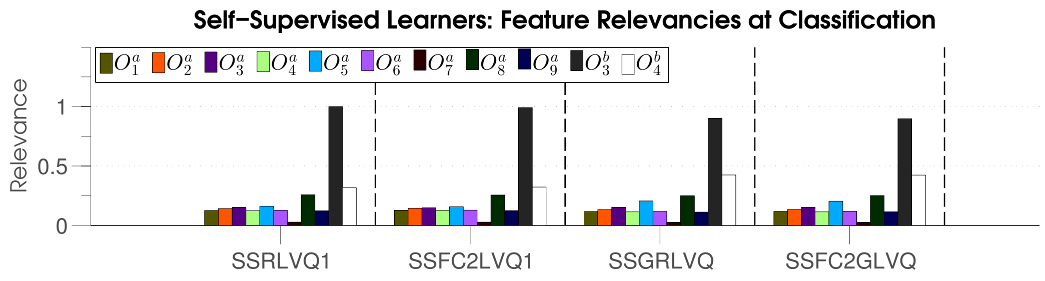
In this experiment, once again, the self-supervised learners were trained using data from $\{X^{a,b}, Y^{a,d}\}$, while the supervised learners were trained with $\{X^{a,b}, L_{GT}(X^{a,b})\}$, where L_{GT} applies ground-truth labels to the input view samples. This time, however, online evaluation and comparison of the learners was performed using LOOOCV over only a single epoch of training, and we focused the results on a more direct comparison between selected supervised and self-supervised learners.

7.3.5.1 Affordance Class Discovery

Optimal performance was achieved very early in training, after ~10 samples, with class-cluster-to-ground-truth match accuracy being maintained at ~100% throughout, meaning that cross-view prediction from input view test samples



(a) Feature relevance from training.



(b) Feature relevance at classification.

Figure **7.26:** Feature relevance bar plots for self-supervised learners showing feature weights, both from in-built feature relevance mechanisms, and from the FC1 feature relevance method being applied at classification time, averaged over all folds and trials. Vertical dashed lines separate bar plots different learners.

should at least have the opportunity to reach optimal ground truth prediction rates within the training period.

7.3.5.2 Affordance Class Prediction

When it comes to predicting these newly discovered classes, Figure 7.27 shows how the various self-supervised learners perform and how they compare to supervised classifiers predicting ground truth labels using input view features. In this test, only a small subset of the object features (curvature features $\{O_3^b, O_4^b\}$) is relevant for class prediction, and the most significant result is that there is a prominent difference between those self-supervised learners with feature relevance mechanisms and those without, most likely due to this reason. Interestingly, all of the self-supervised classifiers out-perform the supervised classifiers throughout training. This initially surprising result is likely due to the fact that, owing to the randomized prototype initialization, the prototype class labels of the supervised

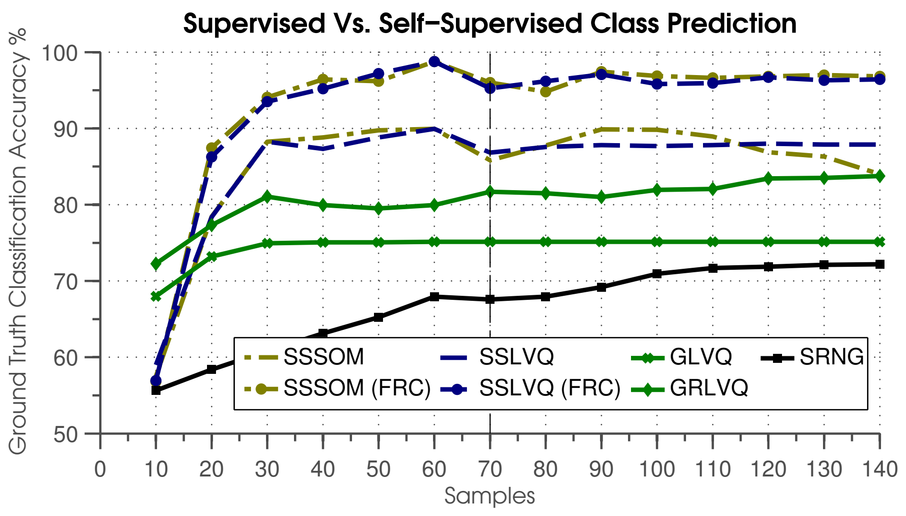


Figure **7.27:** LOOOCV (cf. Section 7.3.2) class prediction results for the Katana/Camera 3-D + 2-D object feature experiment (cf. Section 7.3.5.2). Vertical dashed line indicates transition from unsupervised to self-supervised learning.

classifiers are not optimally distributed with respect to the class distributions at the outset, whereas the self-supervised classifiers undergo dynamic labelling of their input view prototypes as described in Section 4.1.2.2. This means that the supervised classifier prototypes potentially take more time to adjust their positions with respect to the class distributions. It is also worth noting that SS-LVQ appears to maintain more predictive stability over this short training period than SSSOM, which lacks the self-supervised discriminative learning mechanism of SSLVQ (cf. Section 4.3).

7.3.5.3 Feature Relevance Determination

Figure 7.28 illustrates the mean results of feature relevance determination at the end of training for both the input and output views. The input view graph features comparisons with the supervised learners GRLVQ and SRNG that feature feature relevance determination mechanisms. The features $E^a_2, E^a_3, E^a_9, E^d_2$ and E^d_3, those being total distance travelled in y, total Euclidean distance travelled, final y position, the average edge histogram difference and the product of it and its colour counterpart, are highlighted as being the most relevant in the output view for class discovery, which makes sense given their assumed potential for distinguishing rolling versus non-rolling behaviours in the context of this experi-

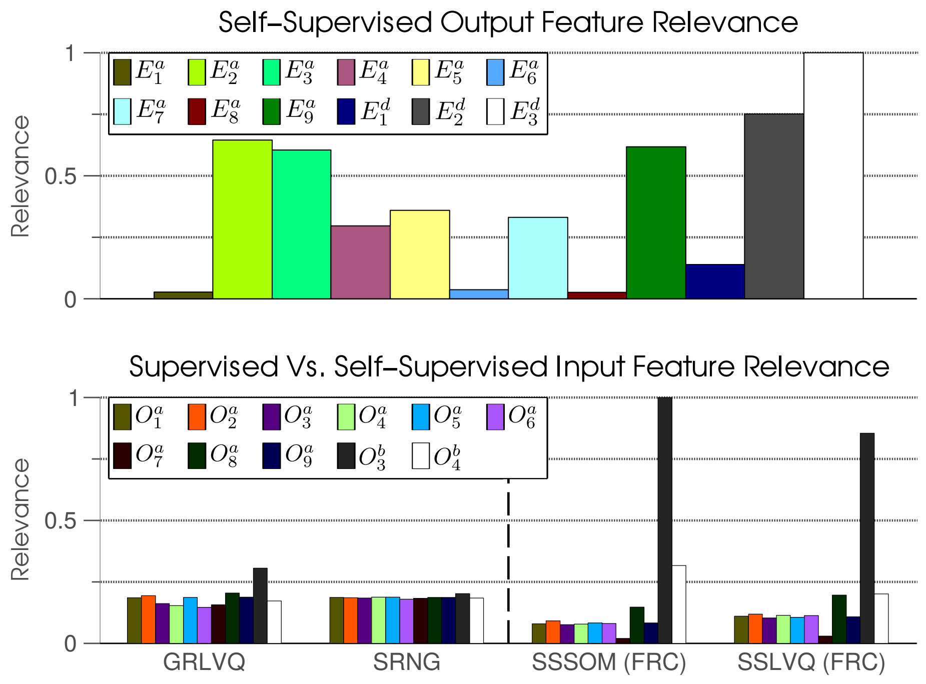


Figure **7.28:** Feature relevance results for the Katana/Camera 3-D + 2-D object features experiment (cf. Section 7.3.5.3). Vertical dashed line in lower figure separates input view feature histograms for two supervised and two self-supervised learners. Upper figure shows output view results which were applicable to self-supervised learners only and were the same in both cases (SSSOM & SSLVQ). Features are colour-coded (see legends).

ment. In the input view, most of the learners correctly select the O_3^b feature, or the object curvature along the x-dimension (along the rolling direction), as being the most relevant for class prediction. Both GRLVQ and SRNG make the least prominent distinction, because, in the short training time tested, their gradient descent-based feature relevance mechanisms most likely do not have sufficient training time in which to function optimally.

7.3.6 Results: 3-D Object Features Dataset

In this experiment, this time, the self-supervised learners were trained using data from $\{X^b, Y^{a,d}\}$, while the supervised learners were trained with $\{X^b, L_{GT}(X^b)\}$, where L_{GT} applies ground-truth labels to the input view samples. Once again, online evaluation was performed using LOOOCV over only a single epoch of training, and we focused the results on a more direct comparison between selected learners.

7.3.6.1 Affordance Class Discovery

In this experiment, both the output view data and learning parameters were the same as in the previous experiment, thus class discovery results were indistinguishable, with optimal performance again being attained early in training and maintained throughout.

7.3.6.2 Affordance Class Prediction

In this experiment, the 2-D visual object features were removed in favour of purely 3-D surface features of feature space X^b, and since many more of these features were relevant for class prediction than in the previous case, there were correlated improvements in the class prediction capabilities of all learners, as illustrated in Figure 7.29. This time all of the self-supervised learners perform comparably well, reaching ~100% performance early in training, as well as beating the supervised classifiers over one epoch of training.

7.3.6.3 Feature Relevance Determination

Since both the features that were used in the output view, as well as the output view learning parameters, were identical to those used in the previous experiment, the results for output feature relevance determination were almost identical also, so we omit those results here. Input view feature relevancies, on the other hand, as shown in Figure 7.30, reveal that many of the features of X^b are relevant for class prediction.

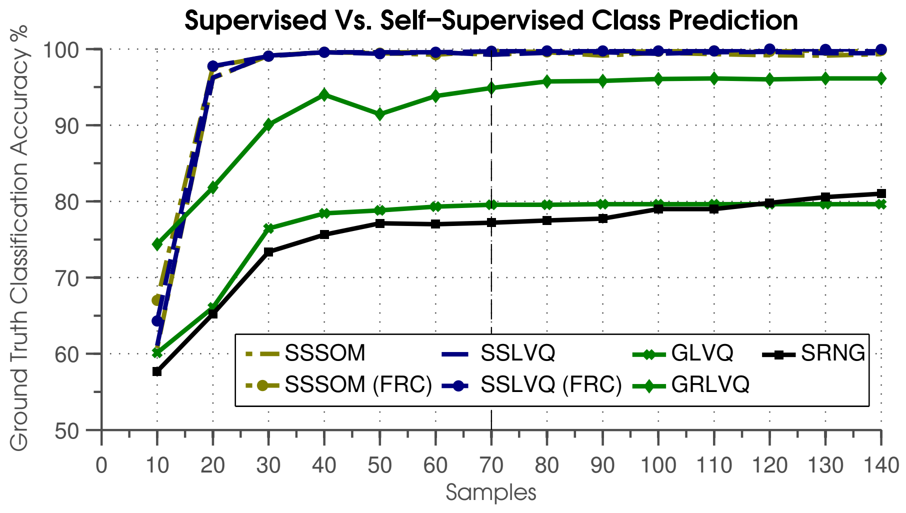


Figure **7.29:** LOOOCV class prediction results for the Katana/Camera 3-D object features experiment (cf. Section 7.3.6).

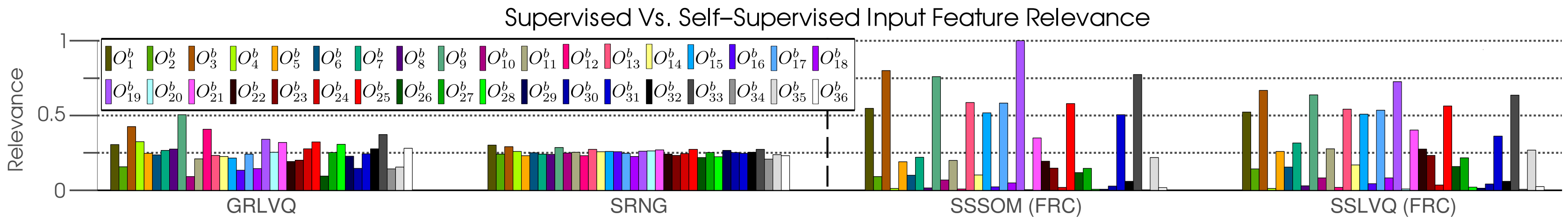


Figure **7.30:** Input view feature relevance results from the Katana/Camera experiment using the 3-D object features (cf. Section 7.3.6). Vertical dashed line separates input view feature histograms for two supervised and two self-supervised learners averaged over 10 trials.

Features O^b_3, O^b_9, O^b_{19}, and O^b_{33} are favoured prominently by both the supervised and self-supervised learners. The selection of O^b_3 matches with the results from the previous experiment, where it was one of two curvature features included in input space $X^{a,b}$, and was selected as the most relevant feature. The right side of the x-axis split that generated the O^b_9 plane normal feature was the closest side to the stereo camera, so produced more 3-D points, thus yielding more reliable surface fits and resulting features. Similarly, O^b_{19} is generated from the back side of the y-axis split, which is the closest object part to the camera along that dimension. Finally, feature O^b_{33} is generated from the part at the intersection between the previous two, and so benefits from the increased reliability of both.

7.3.7 Summary

In summary, the results of applying our self-supervised multi-view learning framework on real-world data from object push affordance learning experiments appear to validate its application at least in this two-class context. The affordance classes are discovered very rapidly by meta-clustering prototypes in the output view. Cross-view class prediction from the input data view works very effectively, reaching near-optimal performance after a short training period for all variants of the self-supervised algorithm when the prototype culling and feature relevance augmentations are applied. Feature relevance determination itself also appears to operate quite effectively, at least when applied at the classification step after dynamic labelling in the input view. Moreover, these results were achieved under the difficult conditions of leave-one-object-out validation, thus demonstrating that the learners can effectively predict the affordances of novel objects from experience learning with similar ones. In the experiments presented in the following section, we sought to further analyse the generalization capabilities of our framework by using a different robotic system, more objects, and an additional affordance class.

7.4 Object Push Affordance Experiments with KUKA-LWR/Kinect Setup

In the KUKA-LWR/Kinect setup experiment, we set out to apply our learning framework to a slightly more complex pushing scenario that would yield more than two affordance classes: rolling, toppling and translating. The environment was set up similarly as in the experiments with the Katana/Camera system (cf. Figure 6.4). This time we selected 10 household objects (cf. Figure 7.31) for the experiments: six flat-surfaced objects and four curved-surfaced objects.

Figure **7.31:** Segmented 3-D object point clouds from the KUKA-LWR/Kinect experiment (not to scale).

7.4.1 Data

During the experiment, objects were again placed at a fixed starting position prior to interaction. This time, however, objects were placed in more varied poses when possible in order to generate more varied affordance effects: flat, sideways or upright, and either with the major axis of the object perpendicular to or parallel to the push direction vector if that was well-defined, i.e. for the non-ball objects. These descriptions are provided here for the readers' benefit and were not used for training, outside of the pose information potentially being encoded in the input feature vectors. During these trials, the flat-surfaced objects would either tend to translate forward or topple over after being pushed, depending on their pose, and the balls would tend to roll, but along much more complex and varied trajectories than in the Katana/Camera experiments, as illustrated in Figure 7.32.

The samples were again hand-labelled with the ground-truth labels: rolling, translating, and toppling. 126 samples were collected in total of a biscuit box, a coffee box, a cookie packet, a contact lens solution box, a marshmallow box, a book, a handball, a small football, and both lightly coloured and darkly coloured larger plastic toy balls. A summary of the dataset is provided in Table 7.4. This time the data was used to generate the 3-D shape features described in Section 6.2.1.3 for the input view space X^c consisting of feature vectors of the

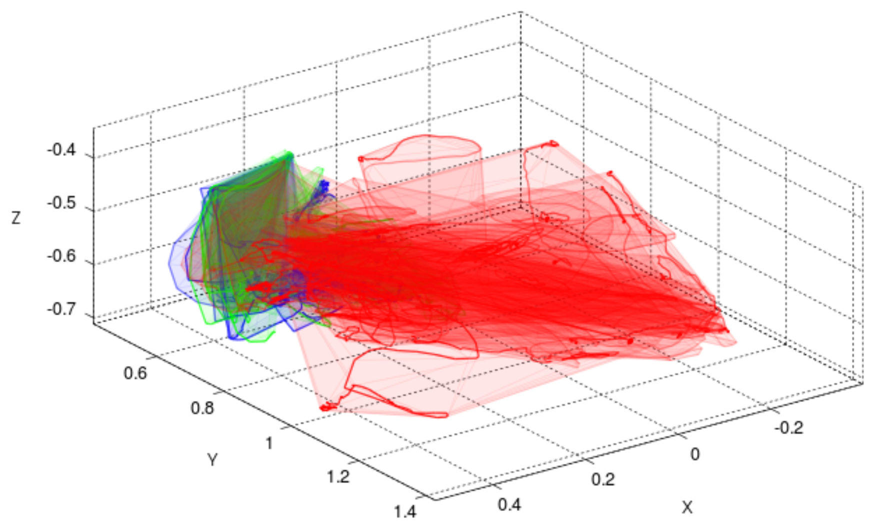


Figure **7.32:** Object trajectories and trajectory convex hulls from the KUKA-LWR/Kinect experiment colour-coded by ground-truth; red: rolling, green: toppling, blue: translating.

Table **7.4:** A summary of the dataset from the KUKA-LWR/Kinect experiment.

Object	# Toppling	# Translating	# Rolling	Total
Biscuit box	8	4	0	12
Coffee box	5	15	0	20
Cookie packet	0	8	0	8
Contact lens box	16	0	0	16
Marshmallow box	9	14	0	23
Book	6	8	0	14
Handball	0	0	8	8
Small football	0	0	6	6
Lightly-coloured ball	0	0	8	8
Darkly-coloured ball	0	0	11	11
Total	44	49	33	126

form $\mathbf{x} = \{O^c_1, \ldots, O^c_{49}\}^T$, as well as the 3-D shape change features and the global motion effect features described in Sections 6.2.2.3 and 6.2.2.4 respectively, to form the output view space $Y^{c,b}$ consisting of feature vectors of the form $\mathbf{y} = \{E^c_1, \ldots, E^c_{49}, E^b_1, \ldots, E^b_{17}\}^T$.

7.4.2 Staged Feature Relevance Determination

Initial learning trials with the full $\{X^c, Y^{a,d}\}$ feature set proved inconsistent, so we opted for a more advanced learning strategy: staged self-supervised learning, in which the most relevant features are determined via thresholding feature relevance at each stage, and the remaining features are discarded before retraining with the reduced feature set at the next stage. This, emulates an aspect of developmental learning, where concepts are refined over multiple learning stages. We thus performed three rounds of self-supervised learning using LOOOCV on the SSLVQ (FRC) learner over 10 epochs for the first two stages, before performing a final run of LOOOCV on all of the learners over 50 epochs using the remaining features. After the first stage, only the features from the output view were reduced, whereas after the second stage, both the input and output view feature sets were reduced. When reducing feature sets, thresholds were set to the mean feature relevance value plus one standard deviation. Output feature relevancies were calculated separately for the shape difference features and for the motion features.

7.4.3 Results

7.4.3.1 Feature Relevance Determination

Input vectors used in the first stage were of the form $\mathbf{x}^{s_1} = \{O^c_1, \ldots, O^c_{49}\}^T$, with output vectors $\mathbf{y}^{s_1} = \{E^c_1, \ldots, E^c_{49}, E^b_1, \ldots, E^b_{17}\}^T$. After the first stage, the input vectors remained as $\mathbf{x}^{s_2} = \mathbf{x}^{s_1}$ for the second stage, while the output vectors were reduced to the form $\mathbf{y}^{s_2} = \{E^c_9, E^c_{11}, E^c_{23}, E^c_{24}, E^c_{38}, E^c_{39}, E^c_{45}, E^b_{14}, E^b_{16}\}^T$, After the second stage, the input vectors were reduced to $\mathbf{x}^{s_3} = \{O^c_4, O^c_{11}, O^c_{18}, O^c_{23}, O^c_{24}, O^c_{25}, O^c_{28}, O^c_{31}, O^c_{32}, O^c_{39}, O^c_{41}, O^c_{45}\}^T$, while the output vectors were reduced to $\mathbf{y}^{s_3} = \{E^c_{45}, E^b_{14}\}^T$. The feature relevance results over the three stages are collated in Figure 7.33. Feature relevance results in Figure 7.35, show output view results at the end of the second stage, and input view results at the end of the third stage respectively.

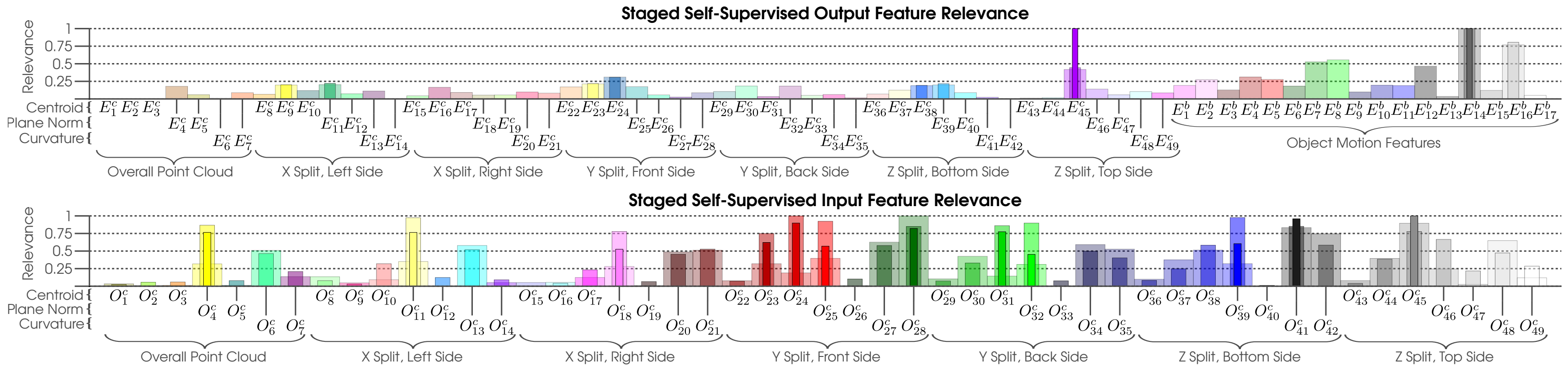


Figure **7.33:** Staged feature relevance determination from the KUKA-LWR/Kinect experiment.

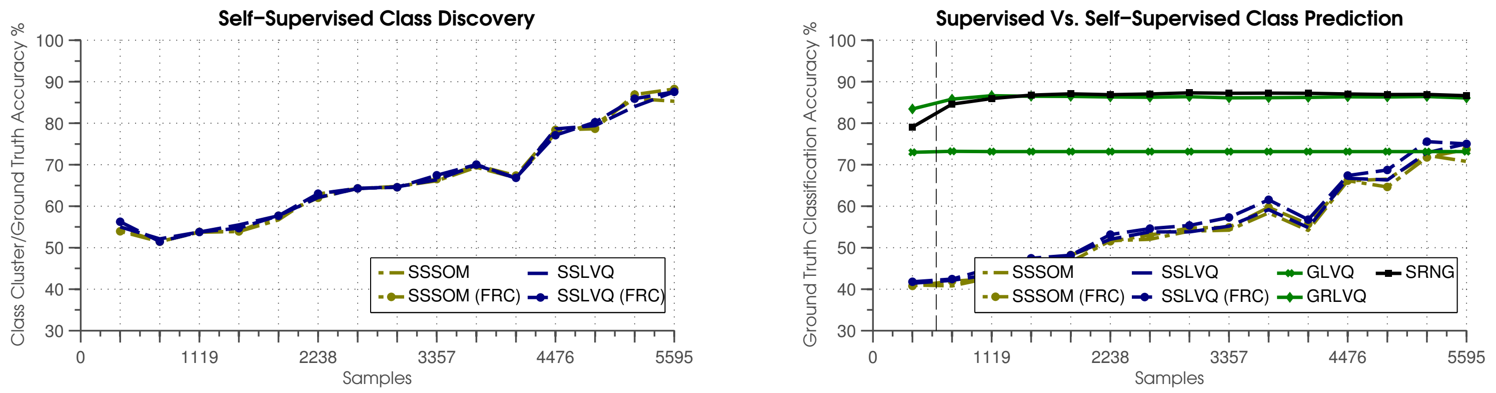


Figure **7.34:** Mean class discovery (left graph) and prediction (right graph) results for the third stage of LOOOCV (cf. Section 7.4.2) run over 50 training epochs in the KUKA-LWR/Kinect experiment.

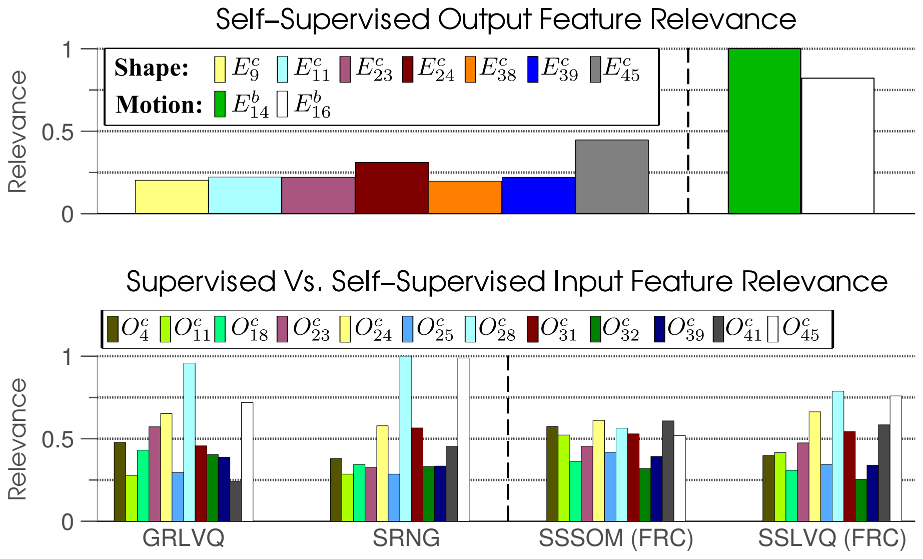


Figure **7.35:** Mean feature relevance results for both input and output views in staged feature relevance learning for the KUKA-LWR/Kinect experiment as described in Section 7.4.2. Upper figure shows relevance histogram of reduced output feature set after second learning stage with a vertical line separating shape and motion features which were clustered and reduced separately at each stage. Lower figure shows reduced feature relevance histograms for input view after third stage where vertical dashed line separates supervised and self-supervised learners.

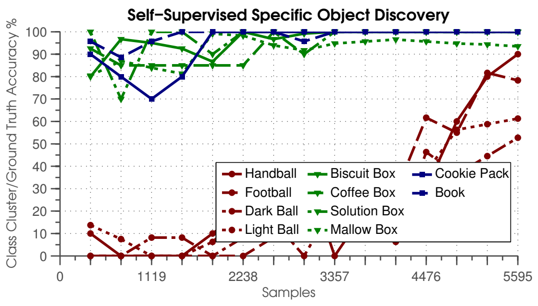
By the end of the final third stage, in the output view, only the E^c_{45} (*z-axis split top side part centroid z-coordinate difference*) and E^b_{14} (*mean trajectory point distance from the start position*) features remain as the most relevant for class discovery, whereas the input view features O^c_{28} (*y-axis split front side part y-curvature*) and O^c_{45} (*z-axis split top side part centroid z-coordinate*) are the most relevant for class prediction. This demonstrates that affordance concepts which associate curvature in the direction of the push (O^c_{28}) and object height (O^c_{45}) with affordance effect classes defined primarily by object height difference (E^c_{45}) and the distance travelled by the object in motion (E^b_{14}), have been formed autonomously using staged self-supervised learning. This appears to match with human intuition for this scenario, though this has not been tested empirically. It is also worth noting that the SSLVQ-based method proposed in this paper proves to be more effective than SSSOM at discerning relevant discriminative features in the input view. This is not too surprising given that the SSSOM lacks a cross-view discriminative learning mechanism.

7.4.3.2 Affordance Class Discovery

The left graph of Figure 7.34 shows average class discovery results for each of the self-supervised learners over 10 trials of the 50-epoch third stage LOOOCV evaluation. Learning parameters were the same in the output view in all cases, so the results are similar for each of the learners. The learners discover the ground truth classes with a mean accuracy of ~90% after 50 epochs of training, which sets a limit on their potential results for class prediction.

7.4.3.3 Affordance Class Prediction

The right graph of Figure 7.34 shows both average class prediction results for all learners and average SSLVQ (FRC) class prediction for specific test objects. While GRLVQ and SRNG reach just over ~85% accuracy on average during most of the 50-epoch training period, the self-supervised learners start out quite poorly, before reaching ~75% accuracy by the end of training. SSLVQ (FRC) appears to show slightly better performance over the other self-supervised learners, but the results are inconclusive. This might be explained by the fact that the feature sets have been significantly refined by this final learning stage. The graphs of Figure 7.36 show that SSLVQ (FRC) appears to struggle most with both class discovery and prediction when either the dark or the light-coloured large balls are left out of the training set. This may be due to the fact that when either of these



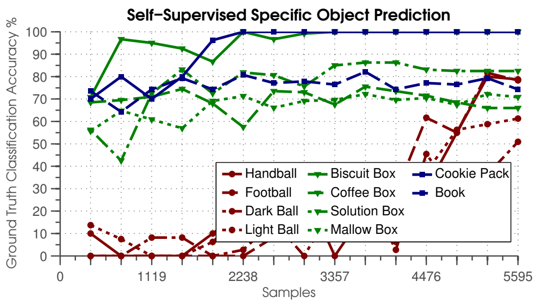


Figure **7.36:** Mean class discovery (left graph) and prediction results (right graph) for all learners & for SSLVQ (FRC) with specific test objects for the third stage of LOOOCV (cf. Section 7.4.2) run over 50 training epochs in the KUKA-LWR/Kinect experiment.

objects are left out, there are far fewer training samples that are both curved in the direction of the push (O^c_{28} feature) *and* are relatively tall (O^c_{45} feature). For example, when the dark ball is left out, there are 115 total training samples, 22 of which are rolling samples, and only 8 of which are samples of relatively tall ball objects rolling. By comparison, when the small football is left out, there are 120 total training samples, 27 of which are rolling samples, and 19 of which are samples of relatively tall rolling balls. Without adequate numbers of samples of tall rolling balls in the training set, the self-supervised learner struggles when it encounters them as novel samples in the test set.

7.4.4 Summary

To summarize this final experimental section, we wished to test our proposed learning framework on a different robotic system, using more objects and more affordance classes, thus the KUKA-LWR/Kinect robotic platform was employed and the additional affordance class of "toppling" was introduced through the use of more varied objects and object poses. When performing learning experiments on the subsequent dataset, it was noted that using the full dataset was yielding poor results, thus we devised a more developmentally staged learning approach in which the most relevant features are determined via thresholding feature relevance at each stage, and the remaining features are discarded before retraining with the reduced feature set. We demonstrated how this staged self-supervised learning process could be successfully applied to the dataset, how the small number of most relevant features could thereby be extracted, and how reasonable class discovery and class prediction results could thereby be achieved.

7.5 Chapter Summary

This chapter was divided into four separate experimental sections. In the first, we evaluated the feature relevance determination algorithms proposed in Chapter 5 in a fully-supervised setting over both a synthetic dataset and various public datasets from the UCI repository. From these experiments we concluded that feature relevance determination can play an important role in boosting the performance of LVQ-based methods, particularly over short-term periods, which was one of our main concerns.

In the second section, we evaluated the self-supervised multi-view learning framework from Chapter 4 over a synthetic multi-view dataset. Online comparisons were performed between various different strains of our self-supervised learning algorithm, using various different update rules, and fully-supervised LVQ-based classifiers using k-folds cross validation. Based on these experiments, we analysed different aspects of the algorithms, including unsupervised class discovery in the output data view, unsupervised multi-view discriminative learning, full-supervision versus self-supervision, feature relevance determination in the self-supervised setting, and LVQ1-based training versus GLVQ-based training. A surprising result to emerge from these experiments was that when comparing fully-supervised and self-supervised LVQ-based learning, the additional information provided by self-supervision from an additional data view can actually provide better performance than fully-supervised learning with ground truth labels, at least over short-term training periods and under hard online learning constraints.

In the last two sections of this chapter, we presented object push affordance learning experiments that were performed using the robotic systems described in Chapter 6. In the first of these sets of experiments, the robotic arm in the Katana/Camera setup was used to push eight different rolling and non-rolling household objects along a work surface and, using the data subsequently gathered, the self-supervised learning framework was tasked with discovering the affordance classes within the data in the output effects data view and learning to predict them from data in the input object features view. In the second set of experiments described in the final section, in order to test our learning framework on another system, the KUKA-LWR/Kinect setup was used, and this time, 10 different household objects were used, resulting in three different affordance classes: rolling, translating and toppling. In each experiment we performed a variation on leave-one-out cross validation, that is, leave-one-object-out cross validation, in which the data for one object was left out of the training set in each fold in order to test the ability of the learners to predict novel objects. The results appeared to validate the effectiveness of our self-supervised multi-view learning approach, showing good performance for object affordance class discovery, object affordance prediction, and differentiating relevant object features for affordance prediction.

Chapter 8

Conclusion

8.1 Thesis Summary

In this thesis we sought to address the problem of basic object affordance learning in a robotic system. Drawing on ideas from a number of different fields, we framed the problem to suit our particular approach and outlined a number of requirements such that our proposed solution would adhere to some ideals of developmental learning, namely online learning and self-supervised learning. Our particular affordance learning scenario involved robotic arms interacting in an environment with a table surface, pushing household objects and observing their behaviour using cameras. The idea was that the system might differentiate between different object behaviours and learn to predict them based on the shape properties of the individual objects.

In developing a solution, we proposed a self-supervised multi-view learning algorithm built on a theoretical foundation of unsupervised clustering, Hebbian learning and learning vector quantization. In this algorithm, codebook layers of prototypes are used to represent separate data views or sensory modalities, and are trained online using competitive learning. The separate data view codebook layers are connected together via a cross-view Hebbian mapping that itself may be trained online using the co-occurrence of data between the separate layers. Class clusters are dynamically formed in a data view by meta-clustering the prototypes and the cross-view mapping can be used to map these classes onto another data view through a process known as Hebbian projection, enabling a form of cross-view classification. We have shown how, given the multi-vew structure of the codebook layers, we may derive learning rules based on both Hebbian projection and the learning vector quantization paradigm that can employ class probabilities instead of actual class labels during training, thus allowing us to bootstrap the self-supervised learning process in an online manner even when the categories are not yet fully known.

We also proposed two new feature relevance determination algorithms for learning vector quantization that use the Fisher criterion score as well as the positioning of prototypes in the input space to form an adaptive Euclidean metric that weighs each feature dimension on their respective discriminative relevance. Both methods may be trained online, thus fitting our learning requirements, and can be applied to any algorithms based on the general learning vector quantization paradigm. They can also be successfully applied to our proposed self-supervised multi-view learning algorithm both during training and at classification time, depending on the particular algorithm, and help to boost learning performance substantially in that regard.

In order to perform actual affordance learning experiments, we also developed a robotic system using a Katana robotic arm and various camera systems such that objects could be interacted with, image data of the interactions could be recorded, and visual features could be extracted from the data. To this end, we presented two computer vision algorithms for extracting object property features from static images of objects and 3-D point cloud data, and for extracting object effect features from videos of objects in motion. The first algorithm used information in the 3-D point cloud data to seed a graph cut segmentation in image data in order to segment objects from the background before extracting shape features. The second algorithm used colour histogram backprojection to refine the output of a particle filter tracker such that local object appearance change features could be extracted. As well as that, we made use of another robotic platform, this one employing a KUKA-LWR robotic arm and the Kinect RGB-D depth sensor, to perform additional experiments and put our learning framework to the test under different conditions.

Finally, we evaluated all of the above in a series of experiments. This was broken into four parts. We wanted to evaluate the proposed feature relevance determination algorithms in isolation so, in the first part, they were tested in a fully-supervised setting using ground-truth labels over both a synthetic dataset and various popular public datasets. These experiments demonstrated that the proposed feature relevance methods could potentially boost learning vector quantization performance significantly, particularly over short-term online training periods, which we were most interested in from an autonomous robotics perspective in which a robot may have to learn rapidly from few examples.

In the second part of our experimental work, we evaluated the proposed self-supervised multi-view learning algorithm, comparing it to equivalent fully-supervised algorithms, over a synthetic multi-view dataset in order to judge its capabilities for potential application to the affordance learning problem. Various different aspects of the algorithm were analysed including its ability to discover categories present in the data from unsupervised clustering, its ability to perform cross-view discriminative category prediction, how it compared to fully-supervised classifiers, and whether it worked well with the proposed feature relevance determination methods. These experiments provided one insightful result in particular, in that it was evident that under the hard online learning conditions we had laid out, LVQ-based self-supervised learning appeared to outperform LVQ-based fully-supervised learning in short-term online learning trials. The reason for this was proposed to be that, in the self-supervised case, the additional information provided by self-supervision from an additional data view allows for dynamic labelling of prototypes under such hard online learning conditions, whereas in the fully-supervised case, lacking such additional information or the opportunity to pre-process the data, the learners must resort to applying ground truth labels to prototypes arbitrarily.

In the final part of our series of experiments we used each of the robotic platforms to gather data from arm interactions with a number of household objects in pushing tests. Some of these objects were round and tended to roll when pushed, while others were flat and tended not to, and in the KUKA-LWR/Kinect setup experiments in particular, still other objects were tall and tended to topple over when pushed. The objective was for the self-supervised learning algorithm to differentiate between these affordance classes and learn to predict them. The results appeared to validate the effectiveness of our learning approach, showing near-optimal performance for object affordance class discovery, object affordance prediction, and determining relevant object features for affordance prediction.

8.2 Summary of Contributions

- **A self-supervised multi-view learning algorithm is proposed that dynamically identifies categories in data while using them to drive online supervised learning.** This part of the work was described in Chapter 4.

- **Two novel feature relevance determination algorithms for learning vector quantization are proposed to augment the self-supervised learning algorithm.** These algorithms were described in Chapter 5.

- **A robotic system with appropriate actuation and visual feature extraction mechanisms has been developed, used to perform real-world experiments on object affordance learning and to test our learning algorithms on the resulting data.** The robotic and vision systems were detailed in Chapter 6.

- **Experimental results demonstrating how when comparing fully-supervised and self-supervised LVQ-based learning under certain conditions, the additional information provided by self-supervision from a separate data view can provide better performance than fully-supervised learning with ground truth labels.** This was discussed in the experimental results of Chapter 7.

8.3 Future Work

With regard to future work on the multi-view learning framework, perhaps one of the most interesting ideas would be to try to replace the vector quantization within data view layers with a generative model of some kind that can also be trained online. One possible candidate for this would be online kernel density estimation [94]. Another potential candidate might be a more advanced form of SOM [104]. Referring back to Figure 3.1 in Chapter 3, it would be very interesting to expand our self-supervised multi-view learning framework to account for alternative models of the affordance learning problem. For instance, parametrising actions and including an action space as a separate data view as depicted in Figure 3.1(a) would make for a more complex, but potentially much more powerful model. To imagine this in more concrete terms, a separate codebook could be used to model structure in the action view which would be cross-view connected to the object effects codebook alongside the object features codebook. The effects view codebook could be used to drive self-supervised learning in both the object features view and the action view. For prediction, the output of both the object and action views could perhaps be combined by ensemble averaging, mixture of experts or similar.

Affordances are, of course, intrinsically action-grounded concepts, and one of the aspects of this thesis we would have liked to have developed further would have been the further development of the action-related components. One way to achieve this would be to increase the number of available actions. For instance, objects might be pushed from different directions and different learners could train on the data for each of the separate actions. This might in turn expand the number of affordances that the system could potentially learn. Some of our current and ongoing work involves using trajectory data from different pushing actions to ground 3-D object point cloud features (similar to those used in Section 6.2.1.3) with respect to a coordinate frame defined by the pushing actions and their contact points on the surfaces of objects [143].

It would also be very interesting to test our framework on other forms of object affordance learning problems, or indeed, affordance learning problems generally. Grasping objects as opposed to pushing them might be one potential line of research here. It would also be very interesting to experiment with different types of visual features, or perhaps features that would come from other sensory systems that could be added to the system. Haptic data from a robotic hand with touch sensors would be such an example.

References

[1] E. Alhoniemi, J. Himberg, and J. Vesanto. Probabilistic measures for responses of self-organizing map units. In *In Proceedings of the International ICSC Congress on Computational Intelligence Methods and Applications*, 1999.

[2] E. Ardizzone, A. Chella, M. Frixione, and S. Gaglio. Integrating subsymbolic and symbolic processing in artificial vision. *Journal of intelligent systems*, 1(4):273–308, 1992.

[3] A. M. Arsenio. Developmental learning on a humanoid robot. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 3167–3172. IEEE, 2004.

[4] R. G. Barker. *Ecological psychology: Concepts and methods for studying the environment of human behavior.* Stanford Univ Pr, 1968.

[5] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 19–26, Washington D.C., USA, November 2004.

[6] C. M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[7] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[8] T. Bojer, B. Hammer, D. Schunk, and K. T. von Toschanowitz. Relevance determination in learning vector quantization. In *European Symposium on Artificial Neural Networks*, pages 271–276, 2001.

[9] N. Bolshakova and F. Azuaje. Cluster validation techniques for genome expression data. *Signal Processing*, 83(4):825–833, April 2003.

[10] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[11] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[12] T. Breuer, M. Ndoundou-Hockemba, and V. Fishlock. First observation of tool use in wild gorillas. *PLoS Biol*, 3(11):e380, October 2005.

[13] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1-3):139–159, 1991.

[14] La casa amarilla — {V}iquipèdia{,} l'enciclopèdia lliure, December 2014. http://ca.wikipedia.org/w/index.php?title=La_Casa_Amarilla, Page Version ID: 14266466.

[15] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[16] G. Card and M. H. Dickinson. Visually mediated motor planning in the escape response of drosophila. *Current Biology*, 2008.

[17] Cephalopod intelligence — {W}ikipedia{,} the free encyclopedia, November 2014. https://en.wikipedia.org/w/index.php?title=Cephalopod_intelligence, Page Version ID: 618746614.

[18] P. Cesari, F. Formenti, and P. Olivato. A common perceptual parameter for stair climbing for children, young and old adults. *Human movement science*, 22(1):111–124, 2003.

[19] C. Chang and C. Lin. LIBSVM: a library for support vector machines. 2001.

[20] Y.-H. Chang, P. R. Cohen, C. T. Morrison, R. S. Amant, and C. R. Beal. Piagetian adaptation meets image schemas: The jean system. In *SAB*, pages 369–380, 2006.

[21] J. Chappell and A. Kacelnik. Tool selectivity in a non-primate, the new caledonian crow (corvus moneduloides). *Animal Cognition*, 5(2):71–78, 2002.

[22] H. H. Chaput, B. Kuipers, and R. Miikkulainen. Constructivist learning: A neural implementation of the schema mechanism. *Proceedings of the Workshop on Self-Organizing Maps (WSOM03)*, 2003.

[23] H. H. Chaput, B. Kuipers, and R. Miikkulainen. *The Constructivist Learning Architecture a Model of Cognitive Development for Robust Autonomous Robots.* PhD thesis, University of Texas at Austin, 2004.

[24] A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision. *Artificial Intelligence*, 89(1-2):73–111, 1997.

[25] A. Chemero, C. Klein, and W. Cordeiro. Events as changes in the layout of affordances. *Ecological Psychology*, 15(1):19–28, January 2003.

[26] A. Chemero. An outline of a theory of affordances. *Ecological Psychology*, 15(2):181–195, 2003.

[27] M. H. Coen. Cross-modal clustering. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 932–937, Pittsburgh, PA, USA, July 2005. AAAI Press.

[28] M. H. Coen. *Multimodal dynamics: self-supervised learning in perceptual and motor systems.* PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2006.

[29] M. H. Coen. Self-supervised acquisition of vowels in american english. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 1451–1456, Boston, MA, USA, July 2006. AAAI Press.

[30] L. B. Cohen, H. H. Chaput, and C. H. Cashon. A constructivist model of infant cognition. *Cognitive Development*, 17(3):1323–1343, 2002.

[31] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[32] I. Cos-Aguilera, L. Canamero, and G. Hayes. Motivation-driven learning of object affordances: First experiments using a simulated khepera robot. In *The*

Logic of Cognitive Systems. Proceedings of the Fifth International Conference on Cognitive Modelling, volume 1001, pages 57–62, 2003.

[33] I. Cos-Aguilera, L. Canamero, and G. Hayes. Using a SOFM to learn object affordances. In *Proceedings of the 5th Workshop of Physical Agents (WAF'04), Girona, Spain*, 2004.

[34] K. Crammer, R. Gilad-bachrach, A. Navot, and N. Tishby. Margin analysis of the LVQ algorithm. In *Advances in Neural Information Processing Systems 15*, pages 462–469, Vancouver, BC, Canada, 2003. MIT Press.

[35] P. Davidsson. Toward a general solution to the symbol grounding problem: combining machine learning and computer vision. In *AAAI Fall Symposium Series, Machine Learning in Computer Vision: What, Why and How*, pages 157–161, 1993.

[36] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979.

[37] V. R. de Sa and D. H. Ballard. Self-teaching through correlated input. *Computation and neural systems*, pages 437–441, 1992.

[38] V. R. de Sa and D. H. Ballard. Category learning through multimodality sensing. *Neural Computation*, 10:1097–1117, 1998.

[39] V. R. de Sa, P. Gallagher, J. Lewis, and V. Malave. Multi-view kernel construction. *Machine Learning*, 79(1):47–71, 2010.

[40] V. R. de Sa. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems 6*, pages 112–119, Denver, CO, USA, 1994. Morgan Kaufmann.

[41] V. R. de Sa. *Unsupervised classification learning from cross-modal environmental structure.* Ph.d. thesis, University of Rochester, 1994.

[42] V. R. de Sa. Sensory modality segregation. In *Advances in Neural Information Processing Systems 16*, pages 913–920, 2003.

[43] V. R. de Sa. Spectral clustering with two views. In *In Proceedings of the ICML 2005 Workshop on Learning With Multiple Views*, Bonn, Germany, 2005.

[44] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning grasp affordance densities. *Paladyn*, 2(1):1–17, 2011.

[45] G. Dorffner, J. Irran, F. Kintzler, and P. Poelz. Robotic learning architecture that can be taught by manually putting the robot to action sequences. Technical report 5.3. 1, The Austrian Research Institute for Artificial Intelligence (OFAI), 2005.

[46] G. L. Drescher. Made-up minds: a constructivist approach to artificial intelligence. *Mit Press Series Of Artificial Intelligence*, page 220, 1991.

[47] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):research0036, June 2002.

[48] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

[49] L. Fe-Fei, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134–1141. IEEE, 2003.

[50] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[51] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–264. IEEE, 2003.

[52] J. K. Finn, T. Tregenza, and M. D. Norman. Defensive tool use in a coconut-carrying octopus. *Current Biology*, 19(23):R1069–R1070, 2009.

[53] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[54] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[55] P. Fitzpatrick and G. Metta. Grounding vision through experimental manipulation. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 361(1811):2165–2185, 2003.

[56] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action-initial steps towards artificial cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, 2003.

[57] P. M. Fitzpatrick. *From First Contact to Close Encounters: A Developmentally Deep Perceptual System for a Humanoid Robot.* PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2003.

[58] A. Frank and A. Asuncion. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences, 2010.

[59] G. Fritz, L. Paletta, R. Breithaupt, E. Rome, and G. Dorffner. Learning predictive features in affordance based robotic perception systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[60] G. Fritz, L. Paletta, M. Kumar, G. Dorffner, R. Breithaupt, and E. Rome. Visual learning of affordance based cues. *From animals to animats*, 9:52–64, 2006.

[61] B. Fritzke. Some competitive learning methods. Technical report, Institute for Neural Computation, Ruhr-Universitat Bochum, 1997.

[62] P. Gärdenfors. Three levels of inductive inference. *Studies in Logic and the Foundations of Mathematics*, 134:427–449, 1995.

[63] A. L. Gibbs and F. E. Su. On choosing and bounding probability metrics. *International Statistical Review*, 70(3):419–435, December 2002.

[64] E. J. Gibson, G. Riccio, M. A. Schmuckler, T. A. Stoffregen, D. Rosenberg, and J. Taormina. Detection of the traversability of surfaces by crawling and walking infants. *Journal of Experimental Psychology: Human Perception and Performance*, 13(4):533, 1987.

[65] J. J. Gibson. *The Ecological Approach to Visual Perception.* Houghton Mifflin, 1979.

[66] J. J. Gibson. *The Ecological Approach to Visual Perception.* Lawrence Erlbaum Associates, 1986.

[67] H. Grabner, J. Gall, and L. Van Gool. What makes a chair a chair? Colorado Springs, USA., 2011.

[68] J. G. Greeno. Gibson's affordances. 1994.

[69] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[70] B. Hammer, M. Strickert, and T. Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters*, 21(1):21–44, 2005.

[71] T. C. Handy, S. T. Grafton, N. M. Shroff, S. Ketay, and M. S. Gazzaniga. Graspable objects grab attention when the potential for action is recognized. *Nature Neuroscience*, 6(4):421–427, 2003.

[72] S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990.

[73] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory.* New York: Wiley, 1949.

[74] X. Huang and J. Weng. Novelty and reinforcement learning in the value system of developmental robots. *Proceedings of the 2nd international workshop on Epigenetic Robotics: Modeling cognitive development in robotic systems*, pages 47–55, 2002.

[75] K. Huebner, S. Ruthotto, and D. Kragic. Minimum volume bounding box decomposition for shape approximation in robot grasping. In *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008*, pages 1628–1633. IEEE, May 2008.

[76] J. Irran, F. Kintzler, and P. Pölz. Grounding affordances. *Cybernetics and Systems. Austrian Society for Cybernetic Studies, Vienna*, 2006.

[77] D. Katz and O. Brock. Manipulating articulated objects with interactive perception. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 272–277, 2008.

[78] D. Katz, Y. Pyuro, and O. Brock. Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In *In Robotics: Science and Systems*. Citeseer, 2008.

[79] B. Kenward, A. A. S. Weir, C. Rutz, and A. Kacelnik. Behavioural ecology: Tool manufacture by naive juvenile crows. *Nature*, 433(7022):121, January 2005.

[80] B. Kenward, C. Rutz, A. A. Weir, and A. Kacelnik. Development of tool use in new caledonian crows: inherited action patterns and social influences. *Animal Behaviour*, 72(6):1329–1343, 2006.

[81] J. M. Kinsella-Shaw, B. Shaw, and M. T. Turvey. Perceiving 'walk-on-able' slopes. *Ecological Psychology*, 4(4):223, December 1992.

[82] S. Kirstein, H. Wersing, and E. Korner. Rapid online learning of objects in a biologically motivated recognition architecture. *Deutschen Arbeitsgemeinschaft fur Mustererkennung*, pages 301–308, 2005.

[83] S. Kirstein, H. Wersing, H.-M. Gross, and E. Körner. A life-long learning vector quantization approach for interactive learning of multiple categories. *Neural Networks*, 28:90–105, April 2012.

[84] D. E. Knuth. *The Art of Computer Programming (Volume 2).* Addison–Wesley, 1981.

[85] W. Kohler and E. Winter. The mentality of apes. 1925.

[86] T. Kohonen. *Self-organizing maps.* Springer, 1997.

[87] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[88] M. Kopicki, A. Sloman, J. Wyatt, and R. Dearden. Learning to predict in robotic manipulation. Technical report, 2008.

[89] M. Kopicki, J. Wyatt, and R. Stolkin. Prediction learning in robotic pushing manipulation. In *International Conference on Advanced Robotics, 2009. ICAR 2009*, pages 1–6. IEEE, June 2009.

[90] M. Kopicki, S. Zurek, R. Stolkin, T. Morwald, and J. Wyatt. Learning to predict how rigid objects behave under simple manipulation. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5722 –5729, May 2011.

[91] M. Kopicki. *Prediction learning in robotic manipulation.* Ph.d. thesis, University of Birmingham, April 2010.

[92] M. Kristan, J. Perš, A. Leonardis, and S. Kovačič. A hierarchical dynamic model for tracking in sports. In *Proceedings of the Sixteenth Electrotechnical and Computer Science Conference*, September 2007.

[93] M. Kristan, J. Perš, M. Perše, and S. Kovačič. Closed-world tracking of multiple interacting targets for indoor-sports applications. *Computer Vision and Image Understanding*, 113(5):598–611, 2009.

[94] M. Kristan, A. Leonardis, and D. Skočaj. Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, 2011.

[95] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, and others. Object–action complexes: Grounded abstractions of sensory–motor processes. *Robotics and Autonomous Systems*, 59(10):740–757, 2011.

[96] M. Krützen, J. Mann, M. R. Heithaus, R. C. Connor, L. Bejder, and W. B. Sherwin. Cultural transmission of tool use in bottlenose dolphins. *Proceedings of the National Academy of Sciences of the United States of America*, 102(25):8939–8943, 2005.

[97] W. J. Krzanowski and Y. T. Lai. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, 44(1):23–34, 1988.

[98] G. Lakoff. *Women, fire, and dangerous things: What categories reveal about the mind.* University of Chicago press, 1987.

[99] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7, 2004.

[100] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[101] S. P. Lloyd. Least squares quantization in PCM's. *Bell Telephone Lab. Paper, Murray Hill. NJ*, 1957.

[102] B. Long, P. S. Yu, and Z. M. Zhang. A general model for multiple view unsupervised learning. In *Proc. 8th SIAM Int. Conf. Data Mining*, pages 822–833, 2008.

[103] M. Lopes and J. Santos-Victor. Visual learning by imitation with motor representations. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 35(3), 2005.

[104] E. López-Rubio. Probabilistic self-organizing maps for continuous data. *Neural Networks, IEEE Transactions on*, 21(10):1543–1554, 2010.

[105] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, 2003.

[106] K. F. MacDorman. Partition nets: An efficient on-line learning algorithm. In *Ninth International Conference on Advanced Robotics*, 1999.

[107] K. MacDorman. Responding to affordances: learning and projecting a sensorimotor mapping. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 4, pages 3253–3259 vol.4, 2000.

[108] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14, 1967.

[109] J. Malcolm, Y. Rathi, and A. Tannenbaum. A graph cut approach to image segmentation in tensor space. In *Workshop on Component Analysis Methods (CVPR)*, pages 18–25, 2007.

[110] L. S. Mark. Eyeheight-scaled information about affordances: A study of sitting and stair climbing. *Journal of Experimental Psychology: Human Perception and Performance*, 13(3):361, 1987.

[111] D. Marr and others. Vision: A computational investigation into the human representation and processing of visual information. 1982.

[112] M. J. Mayo. Symbol grounding and its implications for artificial intelligence. In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 55–60, 2003.

[113] M. E. McCarty, R. K. Clifton, D. H. Ashmead, P. Lee, and N. Goubet. How infants use vision for grasping objects. *Child Development*, 72(4):973–987, 2001.

[114] F. Mechner and L. Guevrekian. Effects of deprivation upon counting and timing in rats. *Journal of the Experimental Analysis of Behavior*, 5(4):463, 1962.

[115] F. Mechner. Probability relations within response sequences under ratio reinforcement. *Journal of the Experimental Analysis of Behavior*, 1(2):109, 1958.

[116] G. Metta and P. Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, 2003.

[117] C. F. Michaels. Affordances: Four points of debate. *Ecological Psychology*, 15(2):135, April 2003.

[118] R. Miikkulainen. Dyslexic and category-specific aphasic impairments in a self-organizing feature map model of the lexicon. *Brain and Language*, 59(2):334–366, 1997.

[119] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1824–1829 vol.2, 2003.

[120] L. Montesano and M. Lopes. Learning grasping affordances from local visual descriptors. In *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, pages 1–6, 2009.

[121] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.

[122] J. Mugan and B. Kuipers. Learning distinctions and rules in a continuous world through active exploration. In *Proc. of the Int. Conf. on Epigenetic Robotics*, 2007.

[123] J. Mugan and B. Kuipers. Towards the application of reinforcement learning to undirected developmental learning. In *Proc. of the Int. Conf. on Epigenetic Robotics*, 2008.

[124] K. M. Newell, D. M. Scully, F. Tenenbaum, and S. Hardiman. Body scale and the development of prehension. *Developmental Psychobiology*, 22(1):1–13, January 1989.

[125] 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 849–856, 2001.

[126] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[127] D. Omrčen, C. Boge, T. Asfour, A. Ude, and R. Dillmann. Autonomous acquisition of pushing actions to support object grasping with a humanoid robot. In *Proceedings of the 9th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 277 –283, December 2009.

[128] R. R. Oudejans, C. F. Michaels, B. van Dort, and E. J. P. Frissen. To cross or not to cross: The effect of locomotion on street-crossing behavior. *Ecological Psychology*, 8(3):259–267, 1996.

[129] E. Oztop, N. S. Bradley, and M. A. Arbib. Infant grasp learning: a computational model. *Experimental Brain Research*, 158(4), June 2004.

[130] L. Paletta and G. Fritz. Reinforcement learning of predictive features. In *Proceedings of 31st Workshop of the Austrian Association for Pattern Recognition (AAPR/OAPR)*, pages 105–112, 2007.

[131] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner. Learning to perceive affordances in a framework of developmental embodied cognition. In *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pages 110–115, 2007.

[132] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner. Perception and developmental learning of affordances in autonomous robots. *LECTURE NOTES IN COMPUTER SCIENCE*, 4667:235, 2007.

[133] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[134] C. E. Pedreira. Learning vector quantization with training data selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):157–162, 2006.

[135] F. S. Perotto, J. C. Buisson, and L. O. Alvares. Constructivist anticipatory learning mechanism (CALM) - dealing with partially deterministic and partially observable environments. In *Proceedings of the Seventh International Conference on Epigenetic Robotics*, 2007.

[136] J. Piaget. *The Origins of Intelligence in Children*. International Universities Press, 1952.

[137] M. Popovic, D. Kraft, L. Bodenhagen, E. Baseski, N. Pugeault, D. Kragic, T. Asfour, and N. Krüger. A strategy for grasping unknown objects based on co-planarity and colour information. *Robotics and Autonomous Systems*, 58(5):551–565, 2010.

[138] M. Pregenzer, G. Pfurtscheller, and D. Flotzinger. Automated feature selection with a distinction sensitive learning vector quantizer. *Neurocomputing*, 11(1):19–29, 1996.

[139] J. D. Pruetz and P. Bertolani. Savanna chimpanzees, pan troglodytes verus, hunt with tools. *Current Biology*, 17(5):412–417, 2007.

[140] A. K. Qin and P. N. Suganthan. Initialization insensitive LVQ algorithm based on cost-function adaptation. *Pattern Recognition*, 38(5):773–776, May 2005.

[141] R. M. Ramstad. *A Constructivist Approach to Artificial Intelligence Reexamined*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1992.

[142] E. S. Reed. *Encountering the world: Toward an ecological psychology*. Oxford University Press, USA, 1996.

[143] B. Ridge and A. Ude. Action-grounded push affordance bootstrapping of unknown objects. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2791–2798, November 2013.

[144] B. Ridge, D. Skočaj, and A. Leonardis. Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5047–5054, Anchorage, USA, May 2010. IEEE.

[145] B. Ridge, A. Leonardis, A. Ude, M. Deniša, and D. Skočaj. Self-supervised online learning of basic object push affordances. *International Journal of Advanced Robotic Systems*, 2014.

[146] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[147] M. T. Rosenstein and P. R. Cohen. Symbol grounding with delay coordinates. In *Working Notes of the AAAI Workshop on The Grounding of Word Meaning*, 1998.

[148] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, November 1987.

[149] A. Roy and S. Marcel. Crossmodal matching of speakers using lip and voice features in temporally non-overlapping audio and video streams. In *2010 International Conference on Pattern Recognition*, pages 4504–4507, 2010.

[150] D. K. Roy and A. P. Pentland. Learning words from sights and sounds: A computational model. *Cognitive science*, 26(1):113–146, 2002.

[151] D. Roy. Grounded speech communication. In *Sixth International Conference on Spoken Language Processing*, 2000.

[152] D. Roy. Learning visually grounded words and syntax of natural spoken language. *Evolution of communication*, 4(1):33–56, 2002.

[153] D. K. Roy. Learning visually grounded words and syntax for a scene description task. *Computer Speech & Language*, 16(3-4):353–385, 2002.

[154] D. Roy. Grounded spoken language acquisition: Experiments in word learning. *IEEE Transactions on Multimedia*, 5(2):197–209, 2003.

[155] D. Roy. Grounding words in perception and action: Insights from computational models. *Trends in Cognitive Science*, 9(8):389–96, 2005.

[156] E. Sahin, M. Cakmak, M. R. Dogar, E. Ugur, and G. Ucoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447, 2007.

[157] J. T. Sanders. An ontology of affordances. *Ecological Psychology*, 9(1):97–112, 1997.

[158] A. Sato and K. Yamada. Generalized learning vector quantization. In *Advances in Neural Information Processing Systems 8*, pages 423–429, Denver, CO, USA, 1996. MIT Press.

[159] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng. Robotic grasping of novel objects. In *In Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems Conference*, Vancouver, Canada, 2006.

[160] A. Saxena, J. Driemeyer, J. Kearns, C. Osondu, and A. Ng. Learning to grasp novel objects using vision. In *Experimental Robotics*, pages 33–42, 2008.

[161] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157, 2008.

[162] A. Saxena, L. Wong, M. Quigley, and A. Ng. A vision-based system for grasping novel objects in cluttered environments. *Robotics Research*, pages 337–348, 2011.

[163] D. Skočaj, B. Ridge, and A. Leonardis. On different modes of continuous learning of visual properties. In *Proceedings of the Fifteenth Electrotechnical and Computer Science Conference*, pages 105–108, Portorož, Slovenia., 2006.

[164] D. Skočaj, G. Berginc, B. Ridge, A. Štimec, M. Jogan, O. Vanek, A. Leonardis, M. Hutter, and N. Hawes. A system for continuous learning of visual concepts. In *Proceedings of Fifth International Conference on Computer Vision Systems (ICVS)*, Bielefeld, Germany, 2007.

[165] D. Skočaj, B. Ridge, G. Berginc, and A. Leonardis. A framework for continuous learning of simple visual concepts. In *Proceedings of the Twelveth Computer Vision Winter Workshop*, pages 99–105, St. Lambrecht, Austria, 2007.

[166] D. Skočaj, M. Kristan, A. Vrečko, B. Ridge, A. Leonardis, S. Roa, and G.-J. Kruijff. DR 5.2 continuous learning of cross-modal concepts. EU FP7 CogX project year 2 deliverable, University of Ljubljana, DFKI Saarbrücken, 2010.

[167] M. Steedman. Plans, affordances, and combinatory grammar. *Linguistics and Philosophy*, 25(5):723–753, 2002.

[168] L. Steels and F. Kaplan. AIBO's first words: The social learning of language and meaning. *Evolution of communication*, 4(1):3–32, 2002.

[169] L. Steels and P. Vogt. Grounding adaptive language games in robotic agents. In *Proceedings of the fourth european conference on artificial life*, pages 474–482, 1997.

[170] T. A. Stoffregen. Affordances as properties of the animal-environment system. *Ecological Psychology*, 15(2):115–134, 2003.

[171] A. Stoytchev. Behavior-grounded representation of tool affordances. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3060–3065, 2005.

[172] A. Stoytchev. Toward learning the binding affordances of objects: A behavior-grounded approach. In *Proceedings of AAAI Symposium on Developmental Robotics*, pages 17–22, 2005.

[173] R. Sun. Symbol grounding: a new look at an old idea. *Philosophical Psychology*, 13(2):149–172, 2000.

[174] S. Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, pages 1–8, 2013.

[175] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[176] J. D. Sweeney and R. Grupen. A model of shared grasp affordances from demonstration. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 27–35, 2007.

[177] A. H. Taylor, G. R. Hunt, J. C. Holzhaider, and R. D. Gray. Spontaneous metatool use by new caledonian crows. *Current Biology*, 17(17):1504–1507, 2007.

[178] C. R. Thouless, J. H. Fanshawe, and B. C. R. Bertram. Egyptian vultures neophron percnopterus and ostrich struthio camelus eggs: the origins of stone-throwing behaviour. *Ibis*, 131(1):9–15, 1989.

[179] M. K. Tsay, K. H. Shyu, and P. C. Chang. Feature transformation with generalized learning vector quantization for hand-written chinese character recognition. *IEICE Transactions on Information and Systems*, E82-D(3):687–692, 1999.

[180] M. Tucker and R. Ellis. On the relations between seen objects and components of potential actions. *Journal of Experimental Psychology: Human perception and performance*, 24(3):830, 1998.

[181] M. T. Turvey, K. Shockley, and C. Carello. Affordance, proper function, and the physical basis of perceived heaviness. *Cognition*, 73(2):B17–B26, December 1999.

[182] M. T. Turvey. Affordances and prospective control: An outline of the ontology. *Ecological Psychology*, 4(3):173–187, 1992.

[183] G. F. Tzortzis and C. L. Likas. Multiple view clustering using a weighted combination of exemplar-based mixture models. *IEEE Transactions on Neural Networks*, 21(12):1925–1938, December 2010.

[184] A. Ude, D. Schiebener, N. Sugimoto, and J. Morimoto. Integrating surface-based hypotheses and manipulation for autonomous segmentation and learning of object representations. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1709 –1715, May 2012.

[185] E. Ugur, M. R. Dogar, M. Cakmak, and E. Sahin. Curiosity-driven learning of traversability affordance on a mobile robot. In *Proceedings of the 6th IEEE International Conference on Development and Learning (ICDL)*, pages 13–18, 2007.

[186] E. Ugur, M. R. Dogar, M. Cakmak, and E. Sahin. The learning and use of traversability affordance using range images on a mobile robot. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 1721–1726, 2007.

[187] E. Ugur, E. Oztop, and E. Sahin. Goal emulation and planning in perceptual space using learned affordances. *Robotics and Autonomous Systems*, 2011.

[188] E. Ugur, E. Sahin, and E. Oztop. Self-discovery of motor primitives and learning grasp affordances. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3260 –3267, October 2012.

[189] E. Ugur, S. Szedmak, and J. Piater. Bootstrapping paired-object affordance learning with learned single-affordance features. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, 2014.

[190] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[191] E. Visalberghi, E. Addessi, V. Truppa, N. Spagnoletti, E. Ottoni, P. Izar, and D. Fragaszy. Selection of effective stone tools by wild bearded capuchin monkeys. *Current Biology*, 19(3):213–217, 2009.

[192] P. Vogt. The physical symbol grounding problem. *Cognitive Systems Research*, 3(3):429–457, 2002.

[193] W. H. Warren and S. Whang. Visual guidance of walking through apertures: Body-scaled information for affordances. *Journal of Experimental Psychology: Human Perception and Performance*, 13(3):371, 1987.

[194] W. H. Warren. Perceiving affordances: Visual guidance of stair climbing. *Journal of Experimental Psychology: Human Perception and Performance*, 10(5):683, 1984.

[195] A. J. Wells. Gibson's affordances and turing's theory of computation. *Ecological Psychology*, 14(3):140–180, July 2002.

[196] J. Weng and W. Hwang. Incremental hierarchical discriminant regression. *IEEE Transactions on Neural Networks*, 18(2):397, 2007.

[197] J. Weng and J. Weng. Developmental robotics: theory and experiments. *International Journal of Humanoid Robotics*, 1(2):199–236, 2004.

[198] J. Weng, W. Hwang, Y. Zhang, C. Yang, and R. Smith. *Developmental Humanoids: Humanoids that Develop Skills Automatically.* 2000.

[199] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. *ARTIFICIAL INTELLIGENCE: Autonomous Mental Development by Robots and Animals*, volume 291. 2001.

[200] J. Weng. The developmental approach to intelligent robots. *1998 AAAI Spring Symposium Series, Integrating Robotic Research: Taking The Next Leap, Stanford University*, 1998.

[201] M. Wertheimer. Laws of organization in perceptual forms. *A source book of Gestalt psychology*, pages 71–88, 1938.

[202] K. Woods, D. Cook, L. Hall, K. Bowyer, and L. Stark. Learning membership functions in a function-based object recognition system. *Arxiv preprint cs.AI/9510103*, 1995.

[203] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, and B. Porr. Cognitive agents — a procedural perspective relying on the predictability of object-action-complexes (OACs). *Robotics and Autonomous Systems*, 57(4):420–432, April 2009.

[204] M. Wünstel and R. Moratz. Automatic object recognition within an office environment. *Canadian Conference on Computer and Robot Vision (CRV2004)*, 2004.

[205] H. Yin and N. M. Allinson. Self-organizing mixture networks for probability density estimation. *IEEE Transactions on Neural Networks*, 12(2):405–411, 2001.

[206] Y. Zhang and J. Weng. Action chaining by a developmental robot with a value system. *Development and Learning, 2002. Proceedings. The 2nd International Conference on*, pages 53–60, 2002.

[207] Y. Zhang and J. Weng. Chained action learning through real-time interactions. *Neural Networks, 2002. IJCNN\'02. Proceedings of the 2002 International Joint Conference on*, 3, 2002.

Appendix A

Derivation of Update Rules for GLVQ that use Probabilities instead of Class Labels

Following the discussion in Section 4.3.2, in the case where we do not have explicit class labels for training samples, but instead have a probability function $0 \leq \phi(\mathbf{w}^j) \leq 1$ indicating the probability of prototype $\mathbf{w}^j$ correctly classifying the current training sample, we may define

$$\mathbf{w}^j = \arg\min_{\mathbf{w}^l} \left\{ d^2(\mathbf{x}, \mathbf{w}^l) \,\middle|\, \forall l : \mathbf{w}^l \in \left\{ \mathbf{w}^i \in \mathcal{W} \,|\, \forall i : \phi(\mathbf{w}^i) \geq \epsilon + \delta \right\} \right\},$$

$$\mathbf{w}^k = \arg\min_{\mathbf{w}^l} \left\{ d^2(\mathbf{x}, \mathbf{w}^l) \,\middle|\, \forall l : \mathbf{w}^l \in \left\{ \mathbf{w}^i \in \mathcal{W} \,|\, \forall i : \phi(\mathbf{w}^i) < \epsilon - \delta \right\} \right\}.$$

to be the nearest matching and non-matching prototypes in the vicinity of sample $\mathbf{x}$ respectively, where ϵ is some threshold value and $\pm\delta$ is a window of uncertainty around ϵ.

The following is adapted from the original GLVQ stochastic gradient descent derivation in [158]. Given an error function

$$E = \sum_{i=1}^{m} f(g(\mathbf{x}^i))$$

recall that the original GLVQ [158] definition for $g(\mathbf{x})$ is

$$\begin{aligned} g(\mathbf{x}) &= \frac{d_j - d_k}{d_j + d_k} \\ &= \frac{d_j}{d_j + d_k} - \frac{d_k}{d_j + d_k} \end{aligned}$$

where d_j and d_k are the distances of x to the nearest correctly labeled prototype and incorrectly labeled prototype respectively.

We seek to re-define $g(\mathbf{x})$ such that it accounts for the uncertainty in the class identity of the prototypes. Recall (Section 4.3.1.2) that under the GLVQ framework, g should be a function that is negative when the sample is classified correctly and positive when the sample is classified incorrectly. We may incorporate the ϕ probability function into g such that when $\phi(\mathbf{w}^j)$ indicates uncertainty in the correctness of $\mathbf{w}^j$, $g(\mathbf{x})$ becomes more positive and, equally, when $\phi(\mathbf{w}^k)$ indicates uncertainty in the incorrectness of $\mathbf{w}^k$, $g(\mathbf{x})$ becomes more negative. We thus re-define $g(\mathbf{x})$ as follows:

$$\begin{aligned} g(\mathbf{x}) &= \left(\frac{d_j}{d_j+d_k} - (1-\phi(\mathbf{w}^j)\left|\frac{d_j}{d_j+d_k} - \frac{1}{2}\right|\right) - \left(\frac{d_k}{d_j+d_k} - \phi(\mathbf{w}^k)\left|\frac{d_k}{d_j+d_k} - \frac{1}{2}\right|\right) \\ &= \frac{2d_j - 2d_k - (1-\phi(\mathbf{w}^j) - \phi(\mathbf{w}^k))|d_j - d_k|}{2(d_j+d_k)}. \end{aligned}$$

Using this altered definition of g, we seek to minimize E via stochastic gradient descent [146], such that the prototypes are updated in the following fashion:

$$\mathbf{w}^i := \mathbf{w}^i - \alpha\frac{\partial E}{\partial \mathbf{w}^i}.$$

Therefore, since at a given timestep t, both $\mathbf{w}^j$ and $\mathbf{w}^k$ have been selected and the terms $\phi(\mathbf{w}^j)$ and $\phi(\mathbf{w}^k)$ are therefore constant, taking the partial derivatives of E with respect to both $\mathbf{w}^j$ and $\mathbf{w}^k$, we have:

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{w}^j} &= \frac{\partial f}{\partial g}\frac{\partial g}{\partial d^j}\frac{\partial d^j}{\partial \mathbf{w}^j} = -\frac{\partial f}{\partial g}\frac{d_k(\phi(\mathbf{w}_t^j) + \phi(\mathbf{w}_t^k) + 1)}{(d_j+d_k)^2}(\mathbf{x} - \mathbf{w}^j), \\ \frac{\partial E}{\partial \mathbf{w}^k} &= \frac{\partial f}{\partial g}\frac{\partial g}{\partial d^k}\frac{\partial d^k}{\partial \mathbf{w}^k} = \frac{\partial f}{\partial g}\frac{d_j(\phi(\mathbf{w}_t^j) + \phi(\mathbf{w}_t^k) + 1)}{(d_j+d_k)^2}(\mathbf{x} - \mathbf{w}^k). \end{aligned}$$

This gives us update rules of the form:

$$\mathbf{w}_{t+1}^j := \mathbf{w}_t^j + \alpha\frac{\partial f}{\partial g}\frac{d_k(\phi(\mathbf{w}_t^j) + \phi(\mathbf{w}_t^k) + 1)}{(d_j+d_k)^2}(\mathbf{x} - \mathbf{w}_t^j),$$

$$\mathbf{w}_{t+1}^k := \mathbf{w}_t^k - \alpha\frac{\partial f}{\partial g}\frac{d_j(\phi(\mathbf{w}_t^j) + \phi(\mathbf{w}_t^k) + 1)}{(d_j+d_k)^2}(\mathbf{x} - \mathbf{w}_t^k),$$

where

$$\frac{\partial f}{\partial g} = f_t(g(\mathbf{x}))(1 - f_t(g(\mathbf{x}))$$

when f is defined as the sigmoidal function

$$f_t(g(\mathbf{x})) = \frac{1}{1+\exp^{-g(\mathbf{x})t}}. \tag{A.1}$$

Appendix B

Extended Summary in Slovenian

B.1 Uvod

B.1.1 Motivacija

Če naj bi roboti bistveno povečali svoje sposobnosti in prežemali naše vsakdanje življenje na vseh področjih, bodo morali biti sposobni avtonomnega učenja in prilagajanja svojemu okolju. Razlog za to je preprost: svet okoli nas je kompleksen! Svet, ki nas obkroža, je tako kompleksen, da, pravzaprav paradoksalno, lahko zaidemo v situacije, ko roboti z lahkoto rešijo najbolj zapletene naloge, navidez enostavne naloge pa predstavljajo zanje zelo zapleten problem. Dandanes se roboti rutinsko uporabljajo za zahtevne, a zelo dobro definirane naloge v nadzorovanih okoljih, kot je proizvodnja avtomobilov. Na tak način, z ročnim programiranjem in definiranjem nalog, je zelo težko oz. skoraj nemogoče upoštevati vse mogoče situacije, ki jih lahko srečamo v bolj nenadzorovanem okolju kot so moderni družinski domovi. Srečni inženir robota, ki opravlja kompleksna a ponavljajoče se dela, kot je npr. izdelava istega tiskanega vezja za avtomobilski satelitski navigacijski sistem, je lahko brezskrben, saj se načrt tiskanega vezja le redko spremeni; po drugi strani pa bo nesrečni snovalec večfunkcijskega hišnega robota obupoval, ko robot ne bo uspel najti belega daljinskega televizijskega upravljavca, ker je bil programiran le za iskanje črnih.

Rešitev tega problema je v učenju; toda razvoj robotov s sposobnostjo učenja je zelo globok in večplasten problem, ki zaobjema veliko različnih raziskovalnih področij, kot so umetna inteligenca, strojno učenje, računalniški vid, psihologija, nevroznanost in druge. Eden izmed teh problemov, in sicer učenje funkcionalnih lastnosti predmetov, je glavna tema tega dela. Ena izmed osnovnih človekovih sposobnosti je zmožnost učenja o predmetih in njihovih lastnostih, tudi funkcionalnih lastnostih; sposobni smo se naučiti kako se predmeti, ki nas obkrožajo, obnašajo, ko na različne načine rokujemo z njimi. Ta sposobnost je bistvena za učenje bolj kompleksnih konceptov, lahko celo rečemo, da je ključna za razvoj naše inteligence. Iz tega lahko sklepamo, da ta sposobnost igra podobno ključno vlogo tudi pri razvoju kompleksnih sposobnosti inteligentnih robotov.

B.1.1.1 Kaj so funkcionalne lastnosti predmetov?

Izraz „funkcionalna lastnost predmeta" (ang. object affordance) je skoval psiholog J.J. Gibson [66] in opisuje interaktivne zmožnosti, ki jih agentu omogoča njegovo okolje. Npr., žoga omogoča, oz. nudi (ang. affords), da se jo zakotali, stikalo pa omogoča prižig luči. Gibson je ta pojem definiral kot:

> „Funkcionalne lastnosti okolja so definirane s tem kar nudijo živalim, kar jim omogočajo, v njihovo dobro ali slabo. ... Nanašajo se tako na okolje kot na žival; implicirajo komplementarnost živali in okolja." [66, str. 127].

Ravno ta komplementarnost živali in okolja igra zelo pomembno vlogo. Ko govorimo o tem, kako bi se spoznavni agent, kot je robot, učil funkcionalnih lastnosti predmetov, moramo upoštevati kar nekaj faktorjev, kot so morfologija ali oblika agenta, morfologija okolja in objektov v njem, kontekst okolja ter kontekst objektov (npr., kako je objekt postavljen v okolje), kot tudi kontekst agenta, in tudi vse možne akcije, ki jih agent lahko izvrši v danem kontekstu. Oz., kot pravi Gibson:

> „Različne postavitve omogočajo različna obnašanja različnim živalim ter različne tipe interakcij. ... Višina kolena za otroka ni enaka višini kolena za odraslega, torej so funkcionalne lastnosti predmetov relativne glede na velikost posameznika." [66, str. 128].

Ravno zato je sposobnost učenja funkcionalnih lastnosti predmetov na dinamičen in razvojen (ang. developmental) način tako ključnega pomena. Za ljudi in tudi vse ostale žive organizme je značilno, da niti dve telesi nista enaki. Še več, telesa se skozi življenje dramatično spreminjajo, iz česar izhaja zahteva po neprestanem prilagajanju in kontinuiranem učenju funkcionalnih lastnosti predmetov. Vsak spoznavni organizem v naravi se je v evolucijski zgodovini razvijal skupaj z okoljem, ravno tako se skupaj s svojim okoljem razvija v teku svojega življenja. In to tako, da sta njegova zaznavni in motorični sistem zgrajena posebej zanj, tako da mu omogočata razpoznavanje in uporabo funkcionalnih lastnosti predmetov, ki jih potrebuje, na način, ki mu omogoča preživetje in razmnoževanje. Oz. kot pravi pregovor: Kar je dobro za Petra morda ni dobro za Pavla: kar ena vrsta organizma zazna kot funkcionalno lastnost, ki jo lahko uporabi, je morda druga vrsta organizma ne more uporabiti ali niti zaznati. Razlog pa je seveda v tem, da imata ta dva organizma različen nabor senzorjev in aktuatorjev za zaznavanje in interakcijo z okoljem.

V naravi najdemo veliko takšnih primerov. Pes se ne bo nikoli naučil prijemanja predmetov z njegovimi šapami, ker mu anatomija njegovih šap to ne omogoča. Krava razpozna travo kot hrano, medtem ko mravlja isto travo razpozna kot visok objekt na katerega lahko spleza. Krava ne bo nikoli zaznala trave kot objekta, ki bi lahko podpiral njeno težo in mravlja se ne bo nikoli namenila prežvečiti celotne travne bilke. Nedavna študija muhe Drosophila [16] zelo lepo ponazarja kako pomanjkanje zmožnosti zaznavanja trenutnega konteksta okolja in akcij, ki jih omogoča, lahko pomeni razliko med življenjem in smrtjo. Mehanizem za nadzor motoričnih akcij, ki pri načrtovanju uporablja vizualno informacijo, ji običajno omogoča pobeg pred grozečo nevarnostjo. Ko nekdo poskuša muho mahniti s časopisom, muhin mehanizem za vizualno zaznavanje zazna časopis, ki se ji hitro približuje, in predno odleti na varno, muha izračuna lokacijo časopisa ter postavi položaj svojega telesa in kril v položaj za vzletanje. Toda, če se muhi s časopisom približamo bolj počasi, le-ta ne zazna tega kot nevarnost in ne izvede običajnega pred-vzletnega protokola, ker je njen zaznavni mehanizem prilagojen hitrim premikom in ne počasnim. Zaradi te neprilagojenosti ima precej manj možnosti za preživetje.

Torej, zakaj je to vprašanje pomembno pri načrtovanju robota sposobnega učenja funkcionalnih lastnosti? Glavni poudarek je na dejstvu, da je zelo velika odvisnost med tem kar sistem lahko zazna, uporabi ali se nauči na eni strani, in funkcionalnimi lastnosti predmetov ter senzorji, aktuatorji in spoznavni mehanizmi, ki so mu na voljo, na drugi strani. Še več, celo zelo majhna sprememba v postavitvi ali majhna razlika med sistemi ima lahko zelo dramatičen vpliv. Npr., zajem dobrega oblaka 3D točk, ki predstavlja majhen predmet, s stereo kamero je zelo odvisen od raznih spremenljivk, kot so razdalja med kamerama, tekstura objekta, osvetlitev, oddaljenost predmeta od kamere, ipd. To lahko povzroči različno zaznavo objekta, kot zaobljenega ali ploskega, grobega ali gladkega, ali celo prisotnega ali neprisotnega. Tem faktorjem moramo torej nameniti posebno pozornost med konstruiranjem takšnega sistema. Čeprav si seveda želimo imeti v sistemu dobre senzorje in aktuatorje, je modro hkrati tudi razviti splošne in plastične mehanizme učenja, ki bodo lahko sistem prilagodili na spremembe v senzorskem in motoričnem sistemu ali na ostale nepredvidene okoliščine brez ponovnega programiranja.

B.1.1.2 Spoznavna in razvojna robotika

Iz naslova doktorske disertacije izhaja, da se v njej predvsem osredotočamo na problem kako naj robot odkriva in se uči osnovnih funkcionalnih lastnosti predmetov v njegovem okolju. Z uporabo svojih senzorjev za opazovanje predmetov, tako statičnih kot med gibanjem, in z uporabo aktuatorjev za interakcijo z njimi, lahko robot izpelje relacijo med morfologijo objektov, različnimi akcijami ter obnašanjem predmetov med interakcijo z njimi. Če je informacija, uporabljena za izpeljavo takšnih relacij, dovolj bogata in se sistem te relacije učinkovito nauči, jih lahko nato posploši in uporabi v novih situacijah. Z drugimi besedami, robot naj bi izvedel akcijo, npr. „potisni center predmeta”, na podobnih predmetih, npr. na nogometni žogi ali na teniški žogici, in opazoval kaj se bo zgodilo; v tem primeru bi se predmeta odkotalila stran, ker sta oba okrogle oblike. Ko bi nato sistem zaznal pomarančo, bi moral biti sposoben sklepati, da se bo tudi odkotalila stran, ker se je tako naučil z opazovanjem obnašanja podobnih predmetov. In, če bi sistem izvedel podobno akcijo potiskanja na škatlah za kosmiče in knjigah, in opazil, da se predmet samo malce podrsa naprej, naj bi bil nato sposoben predvideti, da se bo zaradi podobne oblike podobno gibal tudi paket igralnih kart, če po potisnjen na podoben način.

V zadnjih letih smo bili priča porastu intenzitete aktivnosti na področju razvojne robotike (ang. developmental robotics) [105], s ciljem preiti od trenutnih robotskih sistemov, zasnovanih za konkretne omejene naloge, k bolj robustnim in prilagodljivim platformam in arhitekturam. Zaželene lastnosti takih sistemov vključujejo zmožnost konstruiranja novih konceptov na osnovi predhodno naučenih ali poznanih konceptov, sposobnost aktivnega učenja v interakciji s človeškim mentorjem ali drugim inteligentnim agentom, sposobnost učenja v interakciji z okoljem in predmeti v njem ipd. To so seveda težki problemi za katere (še) ne obstajajo celovite in splošne rešitve, toda med njimi lahko identificiramo mnoge zanimive in rešljive pod-probleme, kot je tudi problem učenja funkcionalnih lastnosti predmetov.

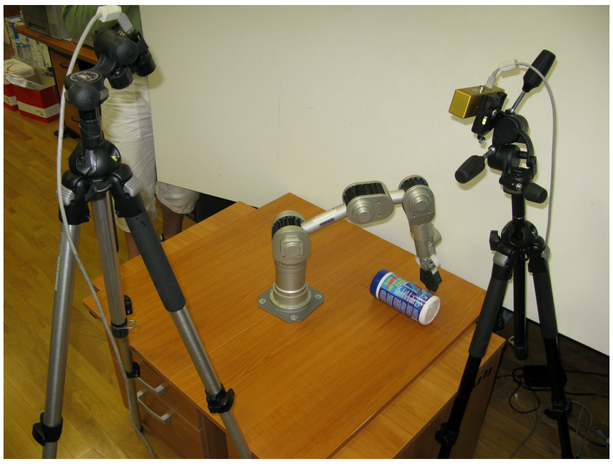
Na področju razvojne robotike je velik poudarek tudi na principu kontinuiranega učenja (ang. continuous, on-line learning). Sistem sposoben kontinuiranega učenja ne zgradi svoje znanje naenkrat, v fazi paketnega učenja, ampak kontinuirano iterativno osvežuje svoje znanje skozi celotni življenjski cikel. Glede na naravo razvojnega robotskega sistema, ki deluje v realnem okolju, kjer se neprestano pojavljajo nove situacije, je realno pričakovati, da sistem ne bo imel venomer dostopa do vseh podatkov iz preteklosti. Takšen scenarij torej izključuje paketno učenje, ki predvideva, da so vsi podatki hkrati na voljo.

Zdi se torej, da je prava pot pri načrtovanju razvojnega robotskega sistema, sposobnega učenja funkcionalnih lastnosti predmetov, takšna, ki bo zgradila sistem zmožen kontinuiranega učenja. Da bi te koncepte prikazali na bolj konkretnem primeru, bomo v naslednjem razdelku podali oris našega scenarija za učenje funkcionalnih lastnosti predmetov, nato pa bomo te raziskovalne teme bolj natančno razdelali v preostanku tega dela.

B.1.2 Scenarij za učenje funkcionalnih lastnosti predmetov

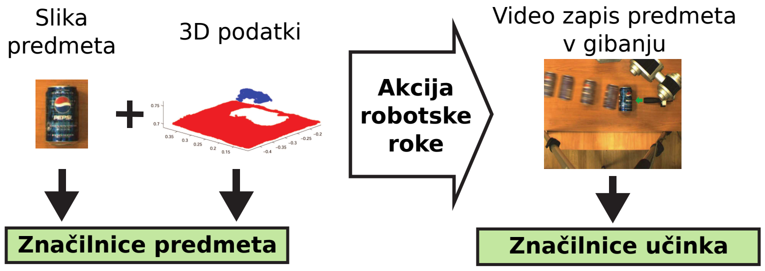
Različni raziskovalci se posvečajo različnim aspektom učenja funkcionalnih lastnosti predmetov v skladu s specifičnimi problemi, ki jih rešujejo in v skladu z opremo, ki jo imajo na voljo. Mi smo se v našem primeru osredotočili na scenarij prikazan na Sliki B.1, ki vključuje robotsko roko, pritrjeno na vodoravno ravno površino, ter kamere različnih tipov, ki opazujejo sceno s fiksnih položajev. Objekte postavljamo na površino, sistem pa z robotsko roko izvaja akcije nad njimi ter s kamerami snema dogajanje.

Osredotočili smo se na bistvo problema učenja funkcionalnih lastnosti predmetov, pri čemer nismo dali poudarka na nekatere detajle s področja računalniškega vida in avtonomne robotike, ki bi še dodatno zapletli celotni problem. Z uporabo takšne postavitve kot je prikazana na Sliki B.1, in ne npr. z uporabo mobilnega robota, smo lahko dodatno omejili naravo interakcij, kar je poenostavilo proces zajemanja podatkov. Naš namen je bil slediti objektom na delovni površini med interakcijo med robotsko roko in njimi. Z vidika računalniškega vida je takšna



Slika **B.1:** Postavitev sistema za učenje funkcionalnih lastnosti predmetov.

statična postavitev kamer zmanjšala verjetnost problematičnih pojavov, kot so zameglitev slike zaradi gibanja, prekrivanje predmetov in odvisnost od velikosti predmetov. Fiksna postavitev robotske roke je tudi olajšala probleme povezane z lokalizacijo predmetov na sceni. Osvetlitev je tudi bila statična in nadzorovana, kar je olajšalo sledenje in segmentacijo predmetov. K temu je pripomogla tudi ravna ter lepo teksturirana delovna površina. Omogoča namreč zanesljivo detekcijo ravne ploskve v oblaku 3D točk, zajetih s stereo kamero, kar posledično olajša zaznavanje ozadja in segmentacijo predmetov postavljenih na mizo.



Slika **B.2:** Glavna ideja našega sistema za učenje funkcionalnih lastnosti predmetov.

Takšen scenarij nam je omogočil implementacijo in raziskovanje naše glavne ideje za učenje funkcionalnih lastnosti predmetov, ki je prikazana na Sliki B.2. Potem, ko postavimo predmet na mizo, sistem zajame vizualne podatke, kot so slike predmeta in oblak 3D točk, ter iz podatkov izloči glavne značilnice predmeta (ang. object features). Robotska roka potem izvede akcijo, sistem pa posname video interakcije med roko in predmetom. Iz tega videa se nato izločijo značilnice učinka akcije (ang. effect features). Naš glavni cilj je bil izgradnja modela ter učnega algoritma za ta scenarij, tako da bi bil robotski sistem sposoben postopoma nadgrajevati znanje o obnašanju predmetov z opazovanjem novih predmetov ter interakcij z njimi. Ko bi torej sistemu predstavili nov predmet, naj bi bil sposoben glede na zajete značilnice predmeta in naučen model predvideti kaj se bo s tem predmetom zgodilo v smislu naučenih učinkov akcij, bodisi kategorij funkcionalnih lastnosti predmeta ali vektorjev značilnic učinka. V idealnem primeru bi si želeli, da bi bil sistem sam zmožen kreirati svoje modele funkcionalnih lastnosti predmetov na vsaj do neke mere razvojen način.

B.1.3 Prispevki

Glavni prispevki k znanosti so naslednji:

- V tem delu predlagamo samo-nadzorovani več-modalni algoritem za učenje, ki dinamično odkrije kategorije v podatkih in jih uporablja za usmerjanje nadzorovanega učenja. Pristop je osnovan na principu kvantizacije vektorjev z uporabo prototipov za predstavitev različnih senzorskih modalnosti (oz. pogledov na podatke), ki so med-modalno povezane s Hebbianovo preslikavo. Algoritem izvaja sprotno gručenje podatkov (ang. clustering) v vsaki izmed modalnosti, medtem ko Hebbianove preslikave hranijo podatke o sopojavnosti med njimi. To omogoča več-modalne preslikave iz ene modalnosti v drugo, ki so gonilna sila samo-nadzorovanosti predlaganega algoritma. Ta je osnovan na paradigmi učenja kvantizacije vektorjev (ang. learning vector quantisation, LVQ). Učenje se odvija brez eksplicitne potrebe po oznakah razredov (ang. labels), saj se namesto tega uporabljajo verjetnosti več-modalnih preslikav.

- Predlagamo tudi dva nova algoritma za določanje ustreznosti posameznih značilnic pri učenju kvantizacije vektorjev, ki dopolnjujeta algoritem za samo-nadzorovano učenje. Algoritma uporabljata Fisherjev kriterij za ocenjevanje pomembnosti posameznih dimenzij v prostoru značilnic. Prvi izmed njiju pri tem uporablja globalne pozicije prototipov, medtem ko drugi uporablja lokalne pozicije najbližjih prototipov. Oba algoritma sta zasnovana tako, da se lahko uporabljata skupaj s kopico metod, ki temeljijo na algoritmu LVQ. V delu tudi pokažemo kako se lahko lokalna metoda uporabi med samim sprotnim samo-nadzorovanim učenjem, ko so na voljo samo verjetnosti za posamezne razrede in ne tudi njihove dejanske oznake.

- Razvili smo robotski sistem z ustreznimi sposobnostmi za izvajanje akcij ter sposobnostmi za zajemanje in procesiranje vizualne informacije, ki izvaja poskuse v realnem svetu ter testira naše algoritme na pridobljenih podatkih. Z nadgradnjo znanih tehnik računalniškega vida smo razvili algoritme za izločanje značilnic oblike predmetov iz statičnih slik ter izločanje značilnic učinkov akcij iz videa. To je zahtevalo razvoj več-modalnega algoritma za segmentacijo objektov z uporabo slik in 3D podatkov. Za sledenje predmetov na videu smo uporabili filter z delci, rezultati pa so bili izboljšani in stabilizirani z uporabo barvnih histogramov, kar je omogočalo analizo sprememb v izgledu predmeta.

- Pokazali bomo tudi eksperimentalne rezultate, ki demonstrirajo, da, ko primerjamo popolnoma nadzorovano in samo-nadzorovano učenje LVQ v določenih pogojih, dodatna informacija, ki jo zagotavlja samo-nadzorovani nadzor iz druge modalnosti lahko zagotovi boljšo delovanje učnega algoritma kot popolnoma nadzorovano učenje s poznanimi oznakami. Poizkusi, ki to demonstrirajo so bili izvedeni v pogojih striktnega sprotnega učenja brez shranjevanja učnih primerov, rezultate pa smo primerjali po kratkih učnih epizodah. V teh pogojih se je boljše izkazal samo-nadzorovani algoritem, ki je izkoristil dinamično označevanje prototipov v nasprotju s popolnoma nadzorovanim učenjem, ki mora zaradi pomanjkanja vnaprejšnje informacije o prototipih, na začetku le-te naključno označiti.

V nadaljnjih razdelkih bomo povzeli glavne komponente doktorske disertacije: predlagani okvir za samo-nadzorovano več-modalno učenje, metodo za odkrivanje pomembnosti posameznih značilnic, robotski sistem za učenje funkcionalnih lastnosti predmetov, ter rezultate poizkusov.

B.2 Samo-nadzorovano več-modalno učenje

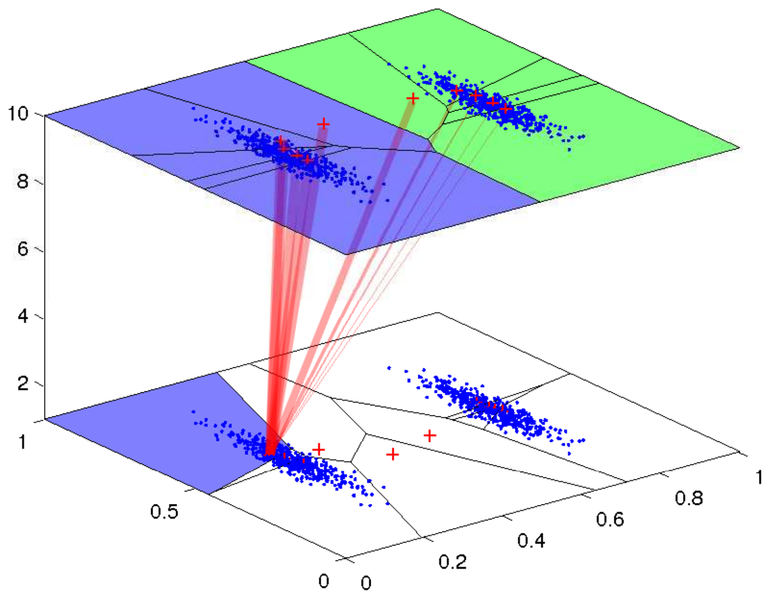
Več-modalno učenje je področje strojnega učenja kjer podatki ne prihajajo iz enega samega prostora značilnic, ampak se podatki sopojavljajo v večih ločenih modalnostih ali pogledih na podatke. Cilji učenja so lahko različni in se razlikujejo glede na določen kontekst v katerem poteka učenje. V zadnjem času je tako precej priljubljeno več-modalno učenje pri katerem se uporablja besedilo na spletni strani kot en pogled na podatke ter povezave na to spletno stran kot drugi pogled [5, 43, 102, 183]. Pogosta aplikacija tega principa v literaturi je tudi poskus povezati sopojavljajoče vizualne in avditorne podatke zajete s snemanjem človeškega govora [38, 126, 149]. V našem scenariju za učenje funkcionalnih lastnosti predmetov obravnavamo lastnosti predmetov (npr. oblika), kot eno modalnost, medtem ko obravnavamo učinke (posledice interakcije med predmetom in robotsko roko) kot drugo modalnost. Modalnost učinkov naj bi usmerjala učenje v modalnosti oblik, npr. s formiranjem kategorij učinkov, ki so vzvratno povezane z modalnostjo oblik. Tak način naj bi omogočal diskriminantno učenje in napovedovanje učinkov akcij zgolj na osnovi oblik predmetov. Te zahteve nas vodijo do ideje samo-nadzorovanega več-modalnega učenja, ki je samo po sebi zelo zahteven problem; še posebej, če dodamo še zahtevo po sprotnem inkrementalnem učenju, kar je pogosto zahteva na področju avtonomne robotike in je tudi ena glavnih zahtev, ki smo si jo postavili pri našem delu.

Naš sistem za samo-nadzorovano več-modalno učenje v veliki meri povzema, kombinira ter nadgrajuje ideje predlagane s strani treh avtorjev. Miikkulainen [118] je predlagal model za simulacijo leksikalnega razvoja, ki več-modalno povezuje šifrante (ang. codebook) vnaprej označenih prototipov vektorjev s Hebbianovimi asociativnimi preslikavami med sopojavljajočimi vektorji. Vsak šifrant predstavlja različne značilnice treh ločenih leksikalnih modalnosti (ortografske, fonološke in semantične). Taki šifranti so zgrajeni na nenadzorovan način z uporabo Samo-organizajočih kart (ang. self-organizing maps, SOM) [86], Hebbianove presikave, ki povezujejo šifrante med seboj, pa z uporabo Hebbianovega učenja. Tako je mogoče, npr., z označevanjem vektorjev prototipov v vsaki modalnosti asocirati fonološke koncepte s semantičnimi. Struktura naše mreže za učenje je zelo podobna strukturi, ki jo je predlagal Miikkulainen, se pa razlikuje način učenja te strukture. V našem delu namreč ne zahtevamo nadzorovano označevanje vhodnih vektorjev, saj smo razvili metodo za nenadzorovano odkrivanje teh oznak.

De Sa [37, 38, 40–42] je razvil algoritem za nenadzorovano učenje nevronskih mrež za učenje ločenih vizualnih in avditornih klasifikatorjev govora z uporabo sopojavljajočih se vzorcev premikanja ustnic in zvočnih signalov. Osnova za učenje te mreže je minimizacija nestrinjanja med modalnostmi z uporabo nenadzorovanega označevanja prototipov in učenja kvantizacije vektorjev za optimizacijo položajev prototipov okrog odločitvenih mej. V našem delu uporabljamo spremenjeno obliko učenja kvantizacije vektorjev, kjer se topološka struktura naše mreže in narava učenja razlikujeta od de Sajeve.

Pred kratkim je Coen [27–29] predlagal algoritem za več-modalno gručenje z mrežo, ki je podobno strukturirana kot Miikkulainenova a brez učnega mehanizma SOM. Coen se osredotoča na iskanje meta-gruč vektorjev prototipov v vsaki modalnosti z uporabo preslikav sopojavljanj za projiciranje verjetnostnih uteži iz prototipov ene modalnosti v drugo. Na ta način se kreira več-modalna metrika razdalje, ki se uporablja za meta-gručenje. Za razliko od Coena, se mi manj osredotočamo na nenadzorovano so-gručenje med modalnostmi in nas bolj zanima iskanje gruč posameznih razredov na nenadzorovan način v eni modalnosti; te gruče nato preko preslikav v druge modalnosti nadzirajo učenje v teh modalnostih.

V tem delu torej predlagamo algoritem za samo-nadzorovano več-modalno učenje, ki temelji na teoretičnih osnovah nenadzorovanega gručenja, Hebbianovega učenja in učenja kvantizacije vektorjev. Šifranti prototipov se uporabljajo za predstavitev posameznih senzorskih modalnosti; naučimo se jih na inkrementalen način v procesu, ki smo ga poimenovali hiper-gručenje. Gruče razredov, ki se uporabljajo za več-modalno klasifikacijo, se oblikujejo v okviru ene modalnosti v procesu meta-gručenja prototipov v tej modalnosti. Posamezne modalnosti so povezane s Hebbianovimi preslikavami, ki se jih naučimo na inkrementalen način z uporabo podatkov o sopojavnosti podatkov med posameznimi modalnostmi. Te več-modalne preslikave se uporabljajo za projiciranje zaznav iz ene modalnosti v drugo s procesom, ki ga poznamo pod imenom Hebbianova projekcija in ki omogoča merjenje ujemanja med podatki iz različnih modalnosti. Ta proces je deloma prikazan na Sliki B.3. V tem delu pokažemo kako lahko ob dani več-modalni strukturi šifranta prototipov izpeljemo učna pravila, ki temeljijo tako na Hebbianovi projekciji kot tudi na paradigmi učenja kvantizacije vektorjev, in ki uporabljajo verjetnosti za posamezne razrede namesto dejanskih oznak. To nam omogoča izvedbo inkrementalnega samo-nadzorovanega učnega procesa, četudi razredi niso vnaprej poznani. To je pomembna lastnost našega pristopa in pogosto zaželena lastnost nižje-nivojskih delov spoznavnih sistemov kot so avtonomni roboti, kjer se nizko-nivojski podatki sopojavljajo na inkrementalen način v različnih modalnostih in kjer se višje-nivojski koncepti dinamično oblikujejo [166].



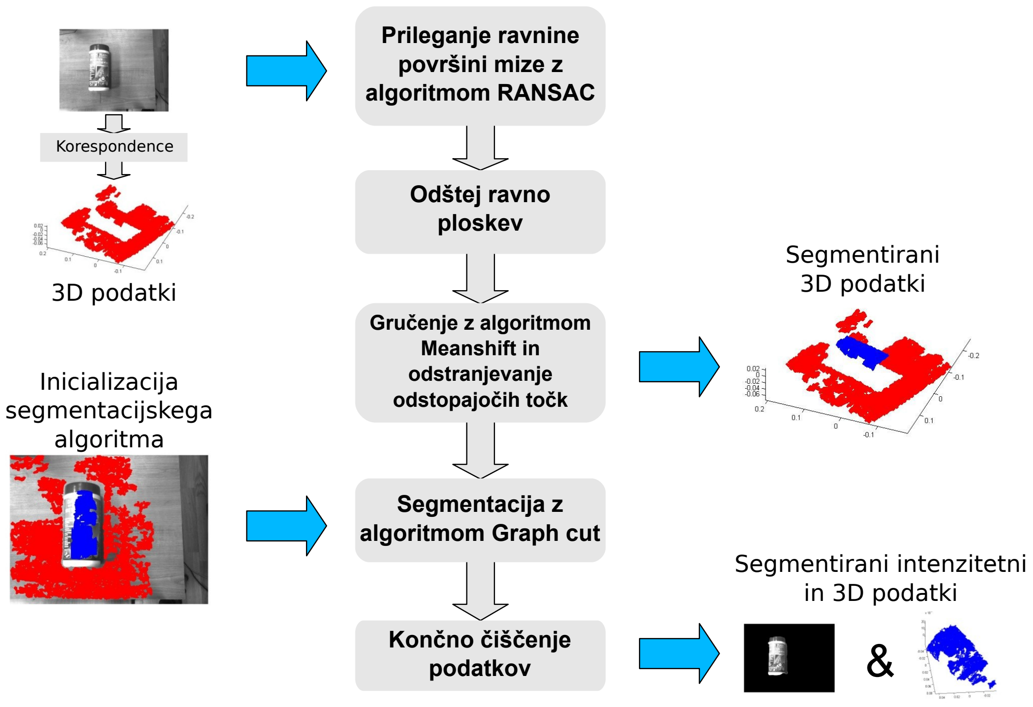
Slika **B.3:** Projekcija meta-gruč iz šifranta ene modalnosti v drugo z namenom več-modalne klasifikacije. Najprej se naučimo šifranta obeh modalnosti (prikazana z Voronoijevo razdelitvijo prostora na zgornjem in spodnjem nivoju) ter preslikav sopojavljanj (utežene rdeče linije). Nato se določijo oznake kategorij z meta-gručenjem zgornjega šifranta z uporabo metode K povprečij (ang. K-means). Te oznake se potem projicirajo na prototipe v spodnjem šifrantu z uporabo Hebbianovih uteži sopojavljanj in se na ta način določijo oznake spodnjih prototipov.

B.3 Določanje ustreznosti značilnic

V naši metodi za samo-nadzorovano več-modalno učenje je učenje realizirano tako, da se najprej s premikanjem prototipov oblikujejo gruče znotraj modalnosti, nato se z uporabo dinamično oblikovanih kategorij označijo prototipi, nakar se nekateri prototipi, najbolj nekoristni v smislu natančnosti napovedovanja, odstranijo iz nadaljnje obravnave. V tem postopku pa se ne upošteva koristnosti posamezne dimenzije v prostoru značilnic, to je koliko dejansko prispevajo k uspešnemu napovedovanju kategorij, niti kako bi se takšna pomembnost oz. ustreznost lahko določila. Klasičen pristop k temu problemu je predprocesiranje podatkov z uporabo ene izmed metod za izbor značilnic ali redukcijo dimenzije prostora značilnic, toda tak pristop je velikokrat neprimeren ali nezadovoljiv, še posebej v scenarijih, ki ne omogočajo shranjevanja podatkov in paketnega učenja, temveč zahtevajo strogo uporabo inkrementalnih algoritmov. V primeru učenja kvantizacije vektorjev lahko določanje ustreznosti značilnic implementiramo na inkrementalen način. V tem delu predlagamo dva algoritma za določanje ustreznosti značilnic za algoritme LVQ, ki uporabljata pozicioniranje prototipov v vhodnem prostoru značilnic za izračun ocen ustreznosti posameznih dimenzij po Fisherjevem kriteriju in s tem oblikujeta adaptivno metriko. Prednost teh metod pred sorodnimi metodami za učenje kvantizacije vektorjev z adaptivnimi metrikami, ki temeljijo na principu spuščanja po gradientu (ang. gradient descent), je, da ne zahtevata nastavitve parametra stopnje učenja niti nastavitve drugih parametrov. Poleg tega tudi vsebujejo pravila za inkrementalno osveževanje, ki jih lahko uporabljamo tudi hkrati s standardnimi pravili učenja LVQ in se tako lahko uporabita na kateremkoli algoritmu, ki temelji na paradigmi LVQ. V disertaciji tudi pokažemo kako lahko metodi uporabimo v primeru našega algoritma za samo-nadzorovano več-modalno učenje, tako med učenjem, kot tudi med klasifikacijo.

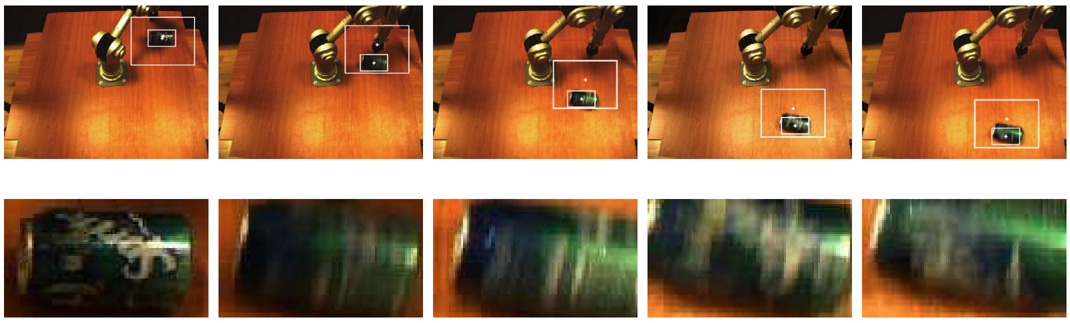
B.4 Robotski sistem za učenje funkcionalnih lastnosti predmetov

Razvoj (in raziskovanje) robotskega sistema za učenje funkcionalnih lastnosti predmetov je težaven proces. Sistem mora namreč podpirati tako zaznavanje okolice kot izvajanje akcij, tako da je potrebno razviti precej različnih podsistemov in jih povezati med seboj. Na področju zaznavanja mora sistem implementirati dovolj robustne module računalniškega vida, tako da lahko „vidi” okolico, ki ga obkroža, in razpozna invariantne značilnice in pomembne vizualne dogodke. Seveda mora biti sistem za učenje funkcionalnih lastnosti sposoben interakcije z okoljem s svojim podsistemom za manipulacijo. Ta zmožnost lahko vključuje vse od relativno enostavne robotske manipulacije, pa do kompleksne kinematike in načrtovanja gibanja. Ena glavnih nalog pri našem delu je bila zasnova robotske platforme, ki združuje dovolj funkcionalnosti iz vsakega izmed teh področij, kar naj bi omogočilo raziskovalno učenje osnovnih funkcionalnih lastnosti predmetov. V tej disertaciji torej obravnavamo strojno opremo, ki smo jo uporabili v sistemu, in vključuje robotsko roko ter različne kamere. Robotska manipulacija, ki smo


Korespondence
3D podatki
Inicializacija segmentacijskega algoritma
Prileganje ravnine površini mize z algoritmom RANSAC
Odštej ravno ploskev
Gručenje z algoritmom Meanshift in odstranjevanje odstopajočih točk
Segmentacija z algoritmom Graph cut
Končno čiščenje podatkov
Segmentirani 3D podatki
Segmentirani intenzitetni in 3D podatki
&

Slika **B.4:** Proces segmentacije predmeta.

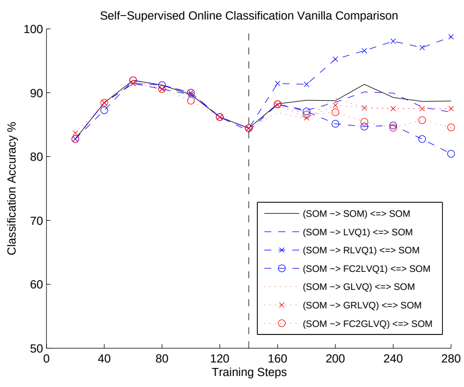
jo preizkusili pri našem delu, implementira potiskanje predmetov, zato še posebej izpostavimo ta scenarij kot tudi delovno okolje sistema in eksperimentalno vrednotenje. Implementirali smo dve glavni kategoriji algoritmov s področja računalniškega vida z namenom izločanja značilnic: značilnice lastnosti objektov izločimo iz statičnih slik in 3-dimenzionalnih podatkov, značilnice učinkov akcij pa izločimo iz video zapisov gibajočih se predmetov. V delu predstavimo dva različna algoritma v obeh kategorijah. Prvi algoritem, prikazan na Sliki B.4, uporablja informacijo iz 3D oblaka točk za postavljanje izhodiščnih točk za inicializacijo segmentacijskega algoritma Graph cut z namenom ločevanja predmeta od ozadja predno se pristopi k računanju značilnic oblike predmeta. Drugi algoritem uporablja vzvratno projekcijo barvnih histogramov za izboljševanje rezultatov sledilnika, ki deluje na principu filtra z delci, tako da zaznava spremembe lokalnih sprememb izgleda predmeta (primer je prikazan na Sliki B.5).


Slika **B.5:** Primer delovanja opisanega mehanizma za sledenje. Slike v prvi vrsti prikazujejo zaporedne slikovne okvirje med sledenjem pločevinki, ki jo potiska robotska roka. Zunanji pravokotnik ponazarja okno verjetja za lokacijo predmeta, ki ga zazna sledilnik delujoč na principu filtra z delci. Ta rezultat je nato izboljšan z vzvratno projekcijo histograma, kar ponazarja notranji pravokotnik. Spodnja vrsta prikazuje kako se spreminja izgled predmeta znotraj notranjega pravokotnika med premikanjem.

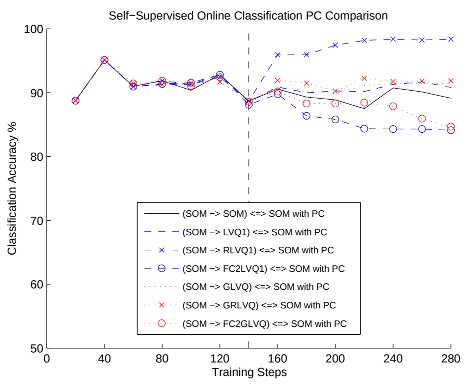
B.5 Rezultati poizkusov

Rezultate poizkusov, ki jih predstavljamo v disertaciji, lahko v grobem razvrstimo v tri razdelke. V prvem smo ovrednotili predlagane algoritme za določanje ustreznosti značilnic v popolnoma nadzorovanih metodah pri čemer smo uporabili tako sintetično zgenerirane podatke kot tudi uveljavljene prosto dostopne množice podatkov iz repozitorija UCI [58]. Na osnovi teh poizkusov lahko zaključimo, da igra določanje ustreznosti značilnic pomembno vlogo pri izboljševanju zmogljivosti metod, ki temeljijo na algoritmu LVQ, še posebej na kratki rok, kar je bil eden naših ciljev.

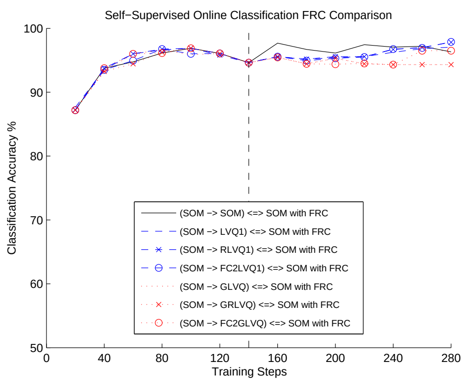
V drugem razdelku smo ovrednotili predlagani algoritem za samo-nadzorovano več-modalno učenje na sintetični množici več-modalnih podatkov. Na inkrementalen način smo primerjali različne izvedbe predlaganega samo-nadzorovanega algoritma, ki so uporabljale različna pravila za osveževanje, ter popolnoma nadzorovani klasifikator LVQ pri čemer smo uporabljali navzkrižno vrednotenje. Na osnovi teh poizkusov smo analizirali različne aspekte algoritmov, kot so nenadzorovano odkrivanje razredov v izhodni modalnosti, nenadzorovano več-modalno diskriminantno učenje, primerjava med popolnoma nadzorovanim in samo-nadzorovanim učenjem, določanje ustreznosti značilnic v samo-nadzorovanem učenju ter primerjava med učenjem temelječim na algoritmoma



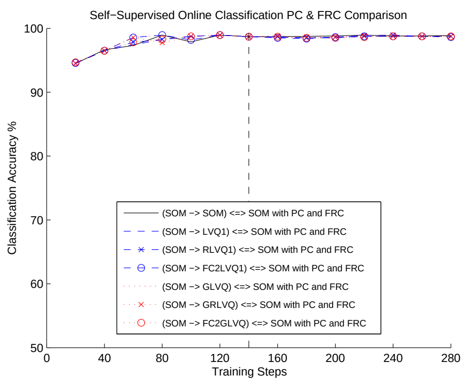
(a)



(b)



(c)



(d)

Slika **B.6:** Sprotno napovedovanje funkcionalnih lastnosti predmetov. Primerjava med različnimi tipi samo-nadzorovanega učenja za dve učni epizodi. Vsaka slika prikazuje rezultate za različne kombinacije odstranjevanja nekoristnih prototipov (ang. prototype culling, PC) ter določanja ustreznosti značilnic ob klasifikaciji (ang. feature relevance at classification time, FRC): (a) običajno, (b) s PC, (c) s FRC ter (d) s PC in FRC.

LVQ1 in GLVQ. Pri analizi rezultatov smo prišli tudi do presenetljivega rezultata, ko smo primerjali popolnoma nadzorovano in samo-nadzorovano učenje LVQ. Izkaže se, da na kratki rok in v strogo inkrementalnih pogojih zaradi dodatne informacije, ki jo zagotovi druga modalnost, samo-nadzorovana metoda prekaša popolnoma nadzorovano metodo z vnaprej poznanimi pravimi oznakami.

V tretjem razdelku smo prikazali rezultate poizkusov učenja funkcionalnih lastnosti predmetov s predstavljenim robotskim sistemom. Sistem uporablja robotsko roko za potiskanje osmih različnih predmetov, ki so se bodisi kotalili ali drsali po površini mize. Na osnovi zbranih podatkov o učinkih akcij sistem odkrije ti dve kategoriji funkcionalnih lastnosti ter se nauči napovedati ti kategoriji na osnovi lastnosti predmetov. Izvedli smo poizkuse pri katerih se je sistem učil na sedmih predmetih, poizkusil pa napovedati funkcionalne lastnosti osmega. Rezultati (nekaj jih je prikazanih na Sliki B.6) potrjujejo učinkovitost predlaganega samo-nadzorovanega več-modalnega algoritma in prikazujejo dobro delovanje v primeru odkrivanja razredov funkcionalnih lastnosti, napovedovanja funkcionalnih lastnosti in razločevanja pomembnih lastnosti objektov z namenov napovedovanja funkcionalnih lastnosti.

B.6 Zaključek

V disertaciji se ukvarjamo s problemom učenja osnovnih funkcionalnih lastnosti predmetov z robotskim sistemom. Upoštevajoč ideje z različnih raziskovalnih področij smo oblikovali problem in postavili zahteve tako, da gre rešitev, ki smo jo razvili, v smeri nekaterih idealov razvojnega učenja kot so sprotno učenje in samo-nadzorovano učenje. Naš scenarij za učenje funkcionalnih lastnosti predmetov vključuje robotsko roko, pritrjeno na vodoravno ravno površino, ki potiska vsakdanje predmete, kamere pa opazujejo dogajanje. Glavni namen je bil zgraditi sistem, ki bi se znal naučiti razlikovati predmete glede na njihovo obnašanje in bi torej znal na osnovi oblike predmeta predvideti, kaj se bo z njim zgodilo, če ga bo robotska roka potisnila v določeni smeri.

Kot del rešitve tega problema smo predlagali algoritem za samo-nadzorovano več-modalno učenje, ki temelji na idejah nenadzorovanega gručenja, Hebbianovega učenja ter učenja vektorske kvantizacije. V tem algoritmu šifranti prototipov predstavljajo posamezne modalnosti, zgradijo pa se na inkrementalen način s konkurenčnim učenjem. Posamezne modalnosti so med seboj povezane z več-modalnimi Hebbianovimi povezavami, ki so ravno tako zgrajene na inkrementalen način z uporabo podatkov o sopojavnosti podatkov iz različnih modalnosti. Gruče, ki predstavljajo posamezne kategorije, se kreirajo v eni modalnosti na dinamičen način z meta-gručenjem prototipov, nakar se več-modalne povezave uporabijo za preslikavo teh kategorij v drugo modalnost z uporabo Hebbianove projekcije, kar omogoča neke vrste več-modalno klasifikacijo. V tem delu smo pokazali kako lahko ob dani več-modalni strukturi šifranta prototipov izpeljemo učna pravila, ki temeljijo tako na Hebbianovi projekciji kot tudi na paradigmi učenja kvantizacije vektorjev, in ki uporabljajo verjetnosti za posamezne razrede namesto dejanskih oznak. To nam omogoča izvedbo inkrementalnega samo-nadzorovanega učnega procesa, četudi razredi niso vnaprej poznani.

Predlagali smo tudi dva nova algoritma za določanje pomembnosti posameznih značilnic pri učenju kvantizacije vektorjev. Uporabljata Fisherjev kriterij za ocenjevanje pomembnosti posameznih dimenzij ter upoštevata položaj prototipov v vhodnem prostoru in na ta način tvorita adaptivno Evklidovo metriko, ki uteži vsako posamezno dimenzijo glede na izračunano pomembnost. Obe metodi sta inkrementalni, in tako izpolnjujeta naše zahteve po inkrementalnem učenju, lahko pa jih apliciramo na katerikoli algoritem, ki temelji na splošni paradigmi učenja vektorske kvantizacije. Lahko se uporabita tudi v algoritmu za sprotno samo-nadzorovano več-modalno učenje, ki ga predlagamo, tako med samim učenjem, kot med klasifikacijo, in pri tem pomembno vplivata na izboljšanje rezultatov.

Za izvajanje poizkusov smo razvili robotski sistem, ki vključuje robotsko roko in sitem kamer. Robotski sistem potiska predmete ter opazuje kaj se z njimi pri tem dogaja. Pri tem posname dogajanje s kamerami ter izloči značilnice iz vizualnih podatkov. S tem namenom smo predstavili dva algoritma računalniškega vida – algoritem za izločanje značilnic lastnosti predmetov iz statičnih slik predmetov in 3-D podatkov ter algoritem za izločanje značilnic učinkov iz video zapisov predmetov v gibanju.

Vse razvite metode smo tudi izdatno ovrednotili v velikem številom poizkusov. Poizkuse lahko razdelimo na tri dele. Najprej smo želeli ovrednotiti samo predlagana algoritma za določanje ustreznosti značilnic, zato smo ju preizkusili v popolnoma nadzorovanem algoritmu z uporabo v naprej poznanih pravih oznak vhodnih vzorcev in to tako na sintetičnih podatkih, kot tudi na podatkih iz različnih priljubljenih javnih množic podatkov. Ti poizkusi so pokazali, da lahko predlagane metode za določanje ustreznosti značilnic bistveno izboljšajo učenje kvantizacije vektorjev, še posebej na kratki rok v pogojih inkrementalnega učenja. Ti pogoji nas v kontekstu inteligentne avtonomne robotike tudi najbolj zanimajo, saj ne zahtevajo učenja v večih epizodah oz. učenja v paketu. V drugem delu našega eksperimentalnega dela smo ovrednotili predlagani algoritem za samo-nadzorovano več-modalno učenje ter ga primerjali z ekvivalentnim popolnoma nadzorovanim algoritmom. Poizkuse smo izvedli na umetno zgeneriranih več-modalnih podatkih, da bi lahko ocenili zmožnosti algoritma za učenje funkcionalnih lastnosti predmetov. Analizirali smo različne vidike algoritma, kot so nenadzorovano odkrivanje razredov, zmožnost več-modalnega diskriminantnega napovedovanja kategorij, primerjava med popolnoma nadzorovanim in samo-nadzorovanim učenjem, ter uporabnost določanja ustreznosti značilnic v samo-nadzorovanem učenju. Ti poizkusi so postregli z zanimivim rezultatom, ki kaže, da na kratki rok in v strogo inkrementalnih pogojih zaradi dodatne informacije, ki jo zagotovi druga modalnost, samo-nadzorovana metoda prekaša popolnoma nadzorovano metodo z vnaprej poznanimi pravimi oznakami. V zadnjem delu naših poizkusov smo uporabili robotsko platformo, ki smo jo razvili, za zajem podatkov o interakciji med robotsko roko in vsakdanjimi predmeti med poizkusi s potiskanjem predmetov. Nekateri objekti so bili okrogle oz. valjaste oblike in so se po površini kotalili, medtem, ko so bili drugi ploščati ter se tako niso zakotalili po površini. Cilj algoritma za samo-nadzorovano učenje je bil ločiti ti dve kategoriji funkcionalnih lastnosti ter se naučiti pravilno napovedati kategorijo. Zdi se, da so rezultati potrdili učinkovitost našega algoritma, saj so se na naši, sicer omejeni, učni domeni približali optimalnim rezultatom tako pri odkrivanju razredov in pri napovedovanju razreda funkcionalnih lastnosti predmetov, kot tudi pri določanju pomembnih značilnosti predmetov za napovedovanje funkcionalnih lastnosti predmetov.

Prevedel: Danijel Skočaj

Published Work

Journal Articles

B. Ridge, A. Leonardis, A. Ude, M. Deniša, and D. Skočaj. Self-supervised online learning of basic object push affordances. *International Journal of Advanced Robotic Systems*, 2014.

Conference Papers

B. Nemec, F. Abu-Dakka, B. Ridge, A. Ude, J. Jorgensen, T. Savarimuthu, J. Jouffroy, H. Petersen, and N. Kruger. Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–7, November 2013.

B. Ridge and A. Ude. Action-grounded push affordance bootstrapping of unknown objects. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2791–2798, November 2013.

B. Ridge, A. Leonardis, and D. Skočaj. Relevance determination for learning vector quantization using the fisher criterion score. In *Proceedings of the Seventeenth Computer Vision Winter Workshop (CVWW)*, Mala Nedelja, Slovenia, February 2012.

B. Ridge, D. Skočaj, and A. Leonardis. Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5047–5054, Anchorage, USA, May 2010. IEEE.

B. Ridge, D. Skočaj, and A. Leonardis. Unsupervised learning of basic object affordances from object properties. In *Proceedings of the Fourteenth Computer Vision Winter Workshop (CVWW)*, pages 21–27, Eibiswald, Austria, 2009.

B. Ridge, D. Skočaj, and A. Leonardis. Towards learning basic object affordances from object properties. In *Proceedings of Eight International Conference on Epigenetic Robotics (EpiRob)*, 2008.

B. Ridge, D. Skočaj, and A. Leonardis. A system for learning basic object affordances using a self-organizing map. In *Proceedings of First International Conference on Cognitive Systems (CogSys))*, 2008.

D. Skočaj, A. Vrečko, M. Kristan, B. Ridge, G. Berginc, and A. Leonardis. Interaktiven sistem za kontinuirano učenje vizualnih konceptov. In *Proceedings of the Sixteenth Electrotechnical and Computer Science Conference*, pages 167–170, Portorož, Slovenia, 2007.

D. Skočaj, G. Berginc, B. Ridge, A. Štimec, M. Jogan, O. Vanek, A. Leonardis, M. Hutter, and N. Hawes. A system for continuous learning of visual concepts. In *Proceedings of Fifth International Conference on Computer Vision Systems (ICVS)*, Bielefeld, Germany, 2007.

D. Skočaj, B. Ridge, G. Berginc, and A. Leonardis. A framework for continuous learning of simple visual concepts. In *Proceedings of Computer Vision Winter Workshop (CVWW)*, St. Lambrecht, Austria, 2007.

D. Skočaj, B. Ridge, and A. Leonardis. On different modes of continuous learning of visual properties. In *Proceedings of Fifteenth International Electrotechnical and Computer Science Conference (ERK)*, Portorož, Slovenia, 2006.

Technical Reports

D. Skočaj, M. Kristan, A. Vrečko, B. Ridge, A. Leonardis, S. Roa, and G.-J. Kruijff. DR 5.2 continuous learning of cross-modal concepts. EU FP7 CogX project year 2 deliverable, University of Ljubljana, DFKI Saarbrücken, 2010.

D. Skočaj, M. Kristan, A. Vrečko, G. Berginc, B. Ridge, A. Leonardis, H. Jacobsson, N. Hawes, G.-J. M. Kruijff, E. Seemann, M. Fritz, and B. Schiele. DR 5.6 framework for continuous learning with different levels of supervision: cognitive systems for cognitive assistants. EU FP6 CoSy project year 3 deliverable, University, Ljubljana, 2007.

Theses

B. Ridge. *Techniques for Computing Exact Hausdorff Measure with Application to a Sierpinski Sponge in* $\mathbb{R}^3$. M.Phil. thesis, School of Mathematics and Statistics, University of St Andrews, 2006.

Declaration

I undertake that all the material presented for examination is my own work, produced independently under the guidance of my supervisors prof. dr. Aleš Leondardis and doc. dr. Danijel Skočaj, and has not been written for me, in whole or in part, by any other person. I also undertake that any quotation or paraphrase from the published or unpublished work of another person has been duly acknowledged in the work which I present for examination.

Ljubljana, December 10, 2014.

Barry Ridge B.Sc. M.Phil.