Linear Regression and Unsupervised Learning for Tracking and Embodied Robot Control

Liam F Ellis

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

August 2010

© Liam F Ellis 2010

ProQuest Number: 27558446

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 27558446

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Summary

Computer vision problems, such as tracking and robot navigation, tend to be solved using models of the objects of interest to the problem. These models are often either hard-coded, or learned in a supervised manner. In either case, an engineer is required to identify the visual information that is important to the task, which is both time consuming and problematic. Issues with these engineered systems relate to the ungrounded nature of the knowledge imparted by the engineer, where the systems have no meaning attached to the representations. This leads to systems that are brittle and are prone to failure when expected to act in environments not envisaged by the engineer. The work presented in this thesis removes the need for hard-coded or engineered models of either visual information representations or behaviour. This is achieved by developing novel approaches for learning from example, in both input (percept) and output (action) spaces. This approach leads to the development of novel feature tracking algorithms, and methods for robot control.

Applying this approach to feature tracking, unsupervised learning is employed, in real time, to build appearance models of the target that represent the input space structure, and this structure is exploited to partition banks of computationally efficient, linear regression based target displacement estimators.

This thesis presents the first application of regression based methods to the problem of simultaneously modeling and tracking a target object. The computationally efficient Linear Predictor (LP) tracker is investigated, along with methods for combining and weighting flocks of LP's. The tracking algorithms developed operate with accuracy comparable to other state of the art online approaches and with a significant gain in computational efficiency. This is achieved as a result of two specific contributions. First, novel online approaches for the unsupervised learning of modes of target appearance that identify *aspects* of the target are introduced. Second, a general tracking framework is developed within which the identified aspects of the target are adaptively associated to subsets of a bank of LP trackers. This results in the partitioning of LP's and the online creation of aspect specific LP flocks that facilitate tracking through significant appearance changes.

Applying the approach to the percept action domain, unsupervised learning is employed to discover the structure of the action space, and this structure is used in the formation of meaningful perceptual categories, and to facilitate the use of localised input-output (percept-action) mappings. This approach provides a realisation of an embodied and embedded agent that organises its perceptual space and hence its cognitive process based on interactions with its environment.

Central to the proposed approach is the technique of clustering an input-output exemplar set, based on output similarity, and using the resultant input exemplar groupings to characterise a perceptual category. All input exemplars that are coupled to a certain class of outputs form a category - the category of a given affordance, action or function. In this sense the formed perceptual categories have meaning and are grounded in the embodiment of the agent. The approach is shown to identify the relative importance of perceptual features and is able to solve percept-action tasks, defined only by demonstration, in previously unseen situations.

Within this percept-action learning framework, two alternative approaches are developed. The first approach employs hierarchical output space clustering of point-to-point mappings, to achieve search efficiency and input and output space generalisation as well as a mechanism for identifying the important variance and invariance in the input space. The *exemplar hierarchy* provides, in a single structure, a mechanism for classifying previously unseen inputs and generating appropriate outputs.

The second approach to a percept-action learning framework integrates the regression mappings used in the feature tracking domain, with the action space clustering and imitation learning techniques developed in the percept-action domain. These components are utilised within a novel percept-action data mining methodology, that is able to discover the visual entities that are important to a specific problem, and to map from these entities onto the action space. Applied to the robot control task, this approach allows for real-time generation of continuous action signals, without the use of any supervision or definition of representations or rules of behaviour.

Key words: Tracking, Adaptive Appearance Models, Embodiment, Robot Control, Unsupervised learning

Email: L.Ellis@surrey.ac.uk

WWW: http://www.eps.surrey.ac.uk/

Acknowledgements

I would like to thank both EPSRC and the European Commission, under both FP7 and Erasmus programme, who supported this work financially.

I would like to thank my supervisors Dr Richard Bowden and Professor Josef Kittler for their encouragement and support and for displaying confidence in my abilities. Richard has acted as supervisor, friend and fellow musician during the course of my studies and has been invaluable as a source of knowledge and inspiration, our interchange of ideas has formed the bedrock of my development in this field of research. Josef has not only provided an excellent working environment, through his management of CVSSP, but his confidence in my abilities has encouraged me to actively contribute to research projects and to develop my research ideas and practices.

I would like to thank my colleagues in the Center for Computer Vision and Speech Signal Processing, at the University of Surrey, and other institutions, for stimulating discussions and both theoretical and practical input during my studies. A special word of thanks for Dr Nicholas Dowson, whose infectious enthusiasm for research provided great motivation during the start of my studies. Also special thanks go to Dr Jiri Matas and Dr Karel Zimmermann, for stimulating discussions and for making my time at Czech Technical University in Prague both productive and interesting.

I'd like to thank all my friends and teachers throughout my life and my family, especially my parents, for always supporting me in the pursuit of my dreams and for loving and understanding me.

Dedicated to my grandfather, Professor Brian Campbell Vickery.
"Gladly would he learn and gladly teach"

Declaration

Elements from this manuscript have appeared in, or are under review for in the following publications.

- *Affordance Mining: Forming Perception Through Action* L. Ellis, R. Bowden, M. Felsberg - To appear in Tenth Asian Conference on Computer Vision
- *Linear Regression and Adaptive Appearance Models for Fast Simultaneous Modelling and Tracking* L. Ellis, N. Dowson, J. Matas, R. Bowden - To appear in International Journal of Computer Vision
- *Problem Solving Through Imitation* E. J. Ong, L. Ellis, R. Bowden - Journal of Image and Vision Computing, 27(11), Oct 2009, pp1715-1728
- *Online Learning and Partitioning of Linear Displacement Predictors for Tracking* L. Ellis, J. Matas, R. Bowden - British Machine Vision Conference 08, Vol 1, pp33-43, Sept 2008
- *Linear Predictors for Fast Simultaneous Modeling and Tracking* L. Ellis, N. Dowson, J. Matas, R. Bowden - Workshop on Non-rigid Registration and Tracking through Learning, IEEE 11th International Conference on Computer Vision, ICCV2007, Rio, Brazil 13-21 Oct 07. pp1-8
- *Learning Responses to Visual Stimuli: A Generic Approach* L. Ellis, R. Bowden - In Proc. 5th International Conference on Computer Vision Systems, ICVS'07, Bielefeld, March 2007
- *Unsupervised Symbol Grounding and Cognitive Bootstrapping in Cognitive Vision* R. Bowden, L. Ellis, J. Kittler, M. Shevchenko, D. Windridge - Proc. International Conference on Image Analysis and Processing, ICIAP05. Lecture Notes in Computer Science, Volume 3617, Nov 2005, Pages 27-36
- *A generalised exemplar approach to modelling perception action couplings* L. Ellis, R. Bowden - International Workshop on Semantic Knowledge in Computer Vision SKCV05 (in Tenth IEEE Intl. Conf. Computer Vision ICCV05), Beijing, China, pp1874

Contents

1 Introduction **1**

1.1 Motivation 4

1.2 Objectives 7

1.3 Contribution 8

1.4 Thesis Overview 10

2 Background **13**

2.1 Classical and Cognitive Computer Vision 13

2.1.1 The Cognitivist paradigm 14

2.1.2 The Emergent Systems paradigm 16

2.1.3 Machine learning 17

2.2 Embodiment 20

2.2.1 Philosophy of embodiment 20

2.2.2 Neuropsychology of embodiment 21

2.2.3 Embodiment in artificial cognitive systems 22

2.3 Imitation Learning 23

2.4 Visual Tracking 24

2.4.1 Tracking via registration 24

2.4.2 Tracking via regression 25

2.4.3 Online appearance model learning 26

2.5 Visual servo control 27

2.6 Robotic systems for autonomous navigation 28

2.7 Motivating discussion 30

3 Linear Regression for Fast Displacement Estimation **33**

3.1 Displacement Estimation: Registration vs. Regression 36

3.1.1 Tracking by registration 37

3.1.2 Tracking by regression: The Linear Predictor 38

3.1.3 Predictor learning 39

3.1.4 The linear predictor flock 43

3.1.5 Inter-frame motion example 45

3.2 Experimental Results 48

3.2.1 Convergence testing 48

3.2.2 Parameter effects 51

3.2.3 Prediction evaluation and flock weighting 53

3.3 Evaluation and Conclusion 55

4 Appearance Models for Tracking **57**

4.1 Simultaneous Modeling and Tracking Approach Overview 61

4.2 Aspect learning for tracking 63

4.2.1 SMAT: Greedy template clustering 66

4.2.2 Medoidshift template clustering 68

4.2.3 Appearance model discussion 70

4.3 Tracking Framework 71

4.3.1 LK-SMAT: Registration based Simultaneous Modeling and Tracking 71

4.3.2 LP-SMAT: Linear Predictors for Simultaneous Modeling and Tracking 72

4.3.3 LP-MED(oidshift): Online partitioning of linear predictors for tracking 75

4.4 Tracking evaluation 79

4.4.1 Car-Surveillance sequence 82

4.4.2 Dudek-Face sequence 83

4.4.3 Runner sequence 87

4.4.4 Head-Motion sequence 87

4.4.5 Camera-Shake sequence 88

4.4.6 Results summary 89

4.5 Evaluation and Conclusion 91

5 Output Space Clustering for Exemplar Learning 97

5.1 Modelling Percept-Action Cycle (PAC) 99

5.1.1 Partitioning the search space 101

5.1.2 Generalising over an exemplar set 102

5.1.3 Output space clustering 103

5.2 The Percept-Action (P-A) Hierarchy 106

5.2.1 Building the hierarchy 106

5.2.2 Searching the hierarchy 109

5.2.3 Discussion of P-A hierarchy 110

5.3 Experiments 111

5.3.1 Experimental setup 111

5.3.2 The exemplar set 113

5.3.3 Evaluating perceptual weighting of output clustering 114

5.3.4 Evaluating generalisation and efficiency 116

5.3.5 Autonomous navigation evaluation 119

5.4 Conclusion and Discussion 120

6 Affordance Mining: Forming Perception Through Action 123

6.1 Embodied Percept-Action Experiences 128

6.1.1 Embodied agent hardware 128

6.1.2 Analysis of embodied training data 131

6.2 Supervised Percept-Action Mapping 134

6.3 Action Space Clustering 135

6.4 Affordance Mining Process Overview 136

6.5 Mining Percept-Action Associations 138

6.5.1 Feature configurations 140

6.5.2 Percept-Action transaction database 141

6.5.3 Mining P-A association rules 143

6.6 Affordance Based Representation 146

6.6.1 Learning action-type specific P-A mappings 146

6.6.2 Responding to novel data 147

6.7 Evaluation 147

6.7.1 Behaviour evaluation 149

6.8 Conclusion and Discussion 152

7 Concluding Discussion **155**

7.1 Summary 155

7.2 Future Work 159

7.2.1 Future work: tracking 160

7.2.2 Future work: robot control 162

Bibliography **165**

List of Figures

1.1 *Thesis Problem Overview* 2

3.1 *Regression Approach to input-output mapping* 34
3.2 *Intensity difference images for eight translations* 39
3.3 *LP response for three training ranges* 41
3.4 *LP prediction examples* 42
3.5 *Inter-frame motion example part I* 46
3.6 *Inter-frame motion example part II* 47
3.7 *Convergence error* 48
3.8 *Convergence success rate* 49
3.9 *Parameter effects* 52
3.10 *Flock weighting results* 53
3.11 *Multi regressor approach to input-output mapping* 56

4.1 *Unsupervised learning in input space for input-output mappings* 58
4.2 *Simultanious Modeling and Tracking methodology* 63
4.3 *Generic system architecture* 64
4.4 *Appearance model medians for head tracking sequence* 67
4.5 *Medoidshift clustering for head tracking sequence* 69
4.6 *LK-SMAT system architecture* 72
4.7 *LP-SMAT system architecture* 74
4.8 *LP-MED system architecture* 77
4.9 *PETS Car-Surveillance sequence results* 83
4.10 *Dudek-Face sequence results* 84
4.11 *Frame rate comparison on Dudek-Face sequence* 85

4.12 *SMAT model medians for Dudek-Face sequence* 86
4.13 *Running sequence results* 89
4.14 *Positional error plots for Runner sequence* 90
4.15 *LP replacement result* 91
4.16 *Camera-Shake sequence results* 92
4.17 *Positional error plots for Camera-Shake sequence* 93

5.1 *Unsupervised learning in output space for input-output mappings* 98
5.2 *NN-PA mapping* 100
5.3 *KD-tree partitioning* 101
5.4 *Output space vs. input space clustering* 105
5.5 *Percept vs action space clustering* 107
5.6 *Action space KD-tree partitioning* 108
5.7 *P-A hierarchy construction* 109
5.8 *P-A hierarchy as percept-action mapping* 112
5.9 *Autonomous navigation hardware and control interface* 113
5.10 *Percept representation for autonomous navigation* 114
5.11 *Action exemplars* 115
5.12 *Perceptual weighting through output space clustering* 116
5.13 *Prediction error as a function of* τ 117
5.14 *Recall and efficiency as a function of* τ 118
5.15 *Frames of video demonstrating autonomous navigation* 119
5.16 *Demonstration of RC car experiment on two sequences* 122

6.1 *Unsupervised learning in output space and percep-action mining for input-output mappings* 127
6.2 *Mobile agent hardware* 129
6.3 *Embodied training data* 130
6.4 *Lead vehicle tracking* 131
6.5 *Supervised vehicle following behaviour dataset* 132
6.6 *Selected pose parameter to control relationship* 133
6.7 *Supervised percept-action mapping* 135
6.8 *Action clustering* 136

6.9 *System overview* 137
6.10 *P-A mining process* 139
6.11 *Encoding configurations* 140
6.12 *Feature configurations* 142
6.13 *Transaction database* 143
6.14 *Association rules* 145
6.15 *Generated action signals* 148
6.16 *Action generation results* 150
6.17 *Behaviour imitation* 151

List of Algorithms

1 LK-SMAT tracking procedure 73
2 LP-SMAT tracking procedure 76
3 LP-MED tracking procedure 80

Chapter 1

Introduction

Everyone takes the limits of their own vision for the limits of the world.

- Arthur Schopenhauer

The work in this thesis in general lies within the research field of computer vision; the problem of interpreting images with a computer in order to achieve some goal. The field of computer vision research has received significant levels of interest from scientists, funding bodies and industry alike, reflecting the perceived advantages and scope of applications of such technology.

Despite the significant successes of many computer vision techniques there are major challenges still remaining especially concerning robust adaptivity and higher *cognitive* processes such as scene interpretation. Having historically diverged from Artificial Intelligence (AI), the image processing, pattern recognition and computer vision communities are once again converging with the AI and Machine Leaning (ML) communities and forming the research field of *cognitive vision* which aims to tackle these issues of robust adaptivity and *cognitive* processes. Of particular interest is the move from engineered, hard-coded or rule based systems to more adaptive systems that learn from experience and improve in performance over time as well as adapt to and deal with unanticipated events or scenarios. The techniques presented in this thesis are developed with the objectives of generality and adaptivity as well as efficiency and robustness in mind; a central mechanism being learning from experience.

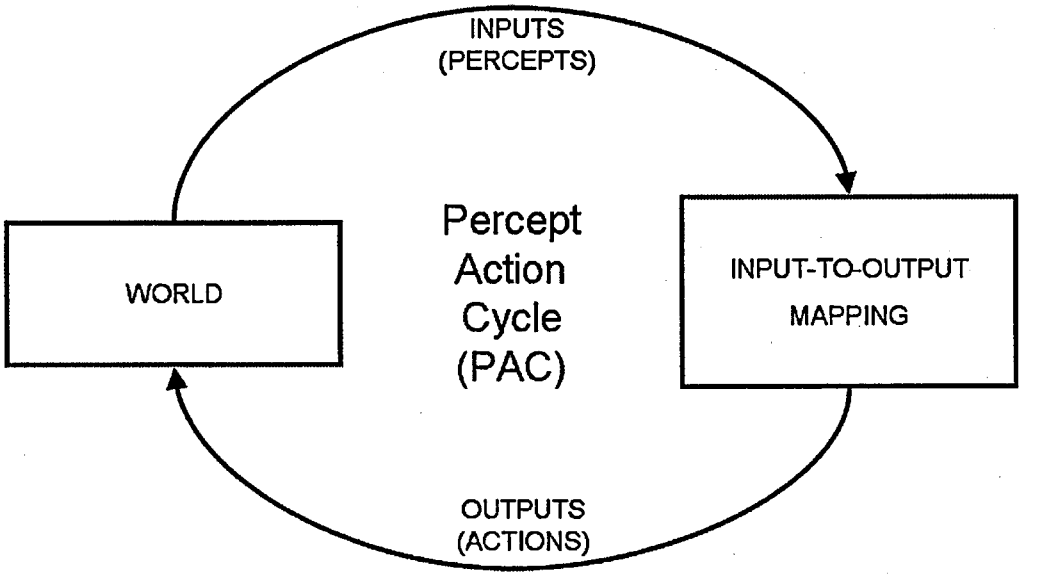


Figure 1.1: *Problem Overview*: This thesis is, in general, concerned with learning mappings from inputs (or percepts) to outputs (or actions). The causal relationships that make up the Percept Action Cycle (PAC) must be learnt and exploited in order to facilitate appropriate behaviour.

This thesis is concerned with learning mappings that map directly from input (image or percept) space to output (action) space without referring to intermediate, high-level, symbolic or top-down representations imposed by an engineer. An overview of this problem is illustrated in figure 1.1. The Percept-Action Cycle (PAC) models the bi-directional causality of an agent interacting with the world. This interaction can entail either the focusing of attention of the perceptual system, the manipulation of objects in the world or the navigation of the agent within the world. In each of these interactions, actions cause a change of percepts and percepts stimulate actions. Within this work the PAC is represented by examples of inputs and outputs. The task then, is for the system to learn the relationships implicit in a set of input-output (percept-action) training examples. Formally, given a set of coupled percept-action training exemplars $E = \{(P_1, A_1), (P_2, A_2), ...(P_n, A_n)\}$, where $\{P_1...P_n\}$ represent percepts and are drawn from vector space $\Re^i$ (real numbered vectors with i dimensions) and $\{A_1...A_n\}$ represent actions and are drawn from vector space $\Re^j$ (j dimensions), the task is to obtain a

mapping $\Re^i \rightarrow \Re^j$ that best approximates the relationships inherent in the examples.

The *vision as regression* paradigm adopted by many researchers provides a tangible mechanism for learning mappings between the input and output parameters. Linear regression functions are utilised within this work to map from input to output space and efficient methods for learning and adapting the regression functions are exploited to achieve real-time, robust performance at both tracking and servo control tasks.

In general, the dimensionality of the input vector space $\Re^i$ will be higher than the dimensionality of the output space $\Re^j$, $i >> j$. This observation motivates a central theme of the proposed approaches, that the organisation of experiences should be based on the structure of the output space. In fact, an important postulate from the emergent paradigm, that has considerable influence on this work, is that of Granlund where he states:

> *"Related points in the response domain exhibit a much larger continuity, simplicity and closeness than related points in the input domain. For that reason, the organisation process has to be driven by the response domain signals."* [33]

Unsupervised learning methods are employed to discover structure, first in the input space as in chapter 4, and later in output space as in chapters 5 and 6. The discovered structure is exploited to both facilitate the formation of perceptual categories and as part of a novel percept-action data mining approach that is able to discover visual entities that are important to a given task, and to associate these entities to action generation models, resulting in the unsupervised formation of an affordance based representation of the world.

The application domains used to illustrate the developed approaches are visual feature tracking and visual servo control for autonomous navigation. These problems share, like many computer vision problems, the need to generate a set of output parameters (i.e. target pose update parameters or servo control signals) from a set of input parameters (e.g. image pixel intensities) and hence are potential applications for the proposed generic input-to-output mapping architecture.

1.1 Motivation

The development of systems or agents capable of learning to respond appropriately to their environment in multiple problem domains is a crucial task to Artificial Intelligence (AI), cognitive sciences, machine learning and computer vision. For many applications, it is not practical or even possible for an engineer to explicitly define the relationships between the input or perceptual space of the agent - it's sensors e.g. cameras or proprioceptive capacities - and it's response or action space. Approaches that allow systems to learn by example and identify and utilise the relationships implicit in examples of appropriate behaviour will widen the scope of applicability of such systems in real world applications. Autonomous systems such as these need to learn via a combination of supervision - which should be minimal and ideally be provided by demonstration rather than hard-coding - exploration and exploitation.

Many conventional approaches to mapping from images to actions require extensive camera calibration and geometric modeling of both objects of interest and the image formation process. These requirements, as well as the requirement for the explicit definition of desired behaviour in a formal language, are often prohibitive. Approaches that require no camera calibration, parameter tuning or explicit definition are therefore desirable. It can also be argued that the very process of *over engineering* a solution to act effectively in real, complex and dynamic environments will tend to lead to *brittle* systems that break as soon as reality strays too far from the idealised world envisaged by the engineer. This argument is typical of the case for *emergent* systems over *cognitivist* systems and is central to the design philosophy adopted throughout this thesis. Chapter 2 gives an overview of the contemporary arguments for the emergent paradigm for the design of cognitive agents.

The emergent systems paradigm is closely coupled with the the philosophy of embodiment and the philosophical theoretical position in cognitive science of Embodied Embedded Cognition (EEC). The philosophy of embodiment, supported by evidence from cognitive science and neuropsychology, is a radical departure from centuries of western philosophical thought and has considerable implications to our conception of the nature of cognition and so it follows that a reappraisal of cognitive systems engi-

neering is necessary. This provides motivation for the approaches developed in chapters 5 and 6. These approaches require an embodied agent that uses interaction with its environment to organise its cognitive process and shape its perception. This is characteristic of an embodied and embedded (situated in an environment) agent. A review of the philosophy of embodiment and its implications to the development of autonomous agents is presented as part of chapter 2.

Visual feature tracking is a widely researched subject that, while remaining a challenging topic, is a prerequisite to computer vision applications that need to model or respond to the dynamics of the observed environment. Given the significant range of tracking approaches developed, each with different strengths and weaknesses, each developed with different user requirements, the development of generic feature trackers that can be applicable to a wide variety of problems is desirable, especially from the end users point of view. For a tracker to be *generic* it should be capable of learning the appearance of the target object on-the-fly as well as being robust to changes in appearance over time. Furthermore, efficiency or low computational cost is of particular importance to trackers expected to operate in real time as high computational cost, meaning more time spent processing each frame, results in larger inter-frame displacements and significant inter-frame appearance changes. This is especially problematic where the target changes pose quickly or where appearance changes are rapid. The desire for computationally efficient visual feature trackers, that require no offline learning, parameter tuning or hard-coding motivates the work presented in chapters 3 and 4. Chapter 3 focuses on efficient displacement estimation functions and the techniques used to learn these functions online.

Besides finding appropriate mappings from input image space to some output parameter space, computer vision is concerned with building representations of the two spaces that are suitable for discriminating inputs and generating outputs. In general, the output space representation is less of an issue, partly due to the fact that the output space generally has fewer dimensions but also because it's structure is more easily interpreted. The higher dimensional image space has a less obvious structure, often with highly uncorrelated and unrelated regions. Whilst this thesis is in part concerned with exploiting the structure discovered in a particular problem output space in order to learn/impose

some structure on the input (image) space, this is not seen as the complete solution. A great deal of computer vision research focuses on the learning, with varying degrees of supervision, and modeling (discriminatory or generative) of perceptual categories. Many of these approaches implicitly employ output space supervision - the categories are specified - as well as varying degrees of input space supervision i.e. labeling of input exemplars (e.g. hand labeling the location of cars for training a car detector or, less supervised, simply stating the presence/absence of the car in a set of images). The development of methods for the unsupervised learning of appearance models to facilitate efficient, robust model free tracking is the focus of chapter 4.

Developing cognitive architectures capable of associating noisy, high dimensional input spaces to appropriate responses in an agents action space is a crucially important task for researches in cognitive sciences, machine learning, AI and computer vision. Central to this problem is the problem of what features of the input space are important to a given task. Systems that are able to autonomously identify the relative importance of different features of the input space, and to use this information to respond appropriately to their environments, will not only remove the need for the explicit definition of perceptual representations, but are likely to be less brittle and more adaptive to changes in the environment not foreseen by the engineer. The work presented in chapter 5 develops an approach that uses input-output exemplars to build generalised input-to-response mappings, allowing the system to learn to respond to previously unseen inputs by identifying the important variance and invariance of the input space.

The formation of perceptual representations that are meaningful to a given task is in fact a fundamental characteristic of all cognitive systems. As mentioned above, the term *embodiment*, used within psychology, philosophy, robotics and artificial intelligence, is based on the premise that the nature of the mind, and in particular the categories of perception, are determined by the embodiment of the cognitive agent [48] [13]. Related to this is *affordance* theory, that states that the world is perceived not only in terms of object shapes and spatial relationships but also in terms of object possibilities for action [29]. The work presented in chapter 6 demonstrates an embodied approach to constructing, in an entirely unsupervised manner, an affordance based representation of the world.

Both chapters 5 and 6 focus on agents that physically interact with their environments. This is achieved through novel visual servo control strategies. Visual servo control is a fundamental problem for all vision systems expected to physically act in the world. Conventional visual servo control techniques rely on strict geometric and projective models and tend to involve extensive and highly sensitive calibration procedures. Furthermore, highly calibrated visual servo control systems have conventionally been driven by hard coded event sequencers or in some cases logical deductive and inductive processes. The desirability of an accurate visual servo control strategy that requires no camera calibration or modeling of scene objects and requires no explicit definition of desired behaviour, provides additional motivation for the work presented in chapter 6.

1.2 Objectives

There are a number of objectives of the research presented in this thesis. Some are broad objectives that specify design methodology and others are more specific and based on the performance requirements of a given task. The objectives are:

1. Develop efficient grouping and mapping strategies.
2. Avoid the explicit definition of representations or rules determining system behaviour.
3. Avoid the reliance on parameter tuning to get algorithms to work on different data sets.
4. Minimise the level of supervision required by learning approaches - i.e. favour unsupervised over supervised.
5. Minimise computationally complexity.

The first objective is the principle objective of the thesis. Efficient mapping strategies provide a means of generating outputs/actions given inputs/percepts. The development of efficient grouping strategies is important for discovering structure in the problem

domain input and output spaces. This structure can be exploited to improve the mapping strategies in terms of the range of inputs and outputs for which the mappings will be effective.

There are two reasons for objectives 2-4. Firstly, each of these objectives reduces the work required of an engineer in order to apply the approaches to a particular problem domain. Secondly it is proposed that approaches that meet these objectives will be more flexible, adaptive and robust and be applicable to a broader set of problem domains.

The fifth objective, although widely adopted in most computer science endeavors, has particular relevance to this work. Both the feature tracking and servo control tasks need to respond to changes in appearance caused by the motion of objects in the world, which may move very quickly. Therefore the response time of the system, proportional to the computational complexity, is an important factor in determining the suitability of the approach to a given problem.

Two key application areas are selected to demonstrate the proposed methodologies. The first is a simultaneous modeling and tracking application that achieves fast, accurate feature tracking with no prior modeling or offline learning. The second application is a computationally efficient visual servo control strategy that requires minimal calibration and no explicit definition of behaviours, but instead learns by example.

1.3 Contribution

This thesis makes a number of contributions to the research fields of computer vision, cognitive vision and robotics.

An example of a linear regression based input-output mapping, the Linear Predictor (LP) tracker, is investigated in order to characterise it's prediction performance and how this performance is affected by various parameterisations (e.g. displacement training range) and conditions (e.g. test displacement ranges). Furthermore the explicit trade off between computational complexity and positional accuracy is characterised.

An approach to combining constellations of LPs into an LP *flock* is introduced and the improvements in accuracy and stability gained are quantified. Mechanisms for

evaluating LP performance and controlling the contribution each LP makes to the overall flock output are tested and show improvements over regular unweighted flocks.

The relationship between regression and registration techniques is explored and the relative strengths of the two approaches are evaluated with illustrative results highlighting the pitfalls of the registration techniques.

This thesis presents the first application of regression based methods to the problem of simultaneously modeling and tracking a target object. The resulting tracking algorithms operate with accuracy comparable to other state of the art online approaches and with a significant gain in computational efficiency. This is achieved as a result of two specific contributions. First, novel online approaches for the unsupervised learning of modes of target appearance that identify *aspects* of the target are introduced. Second, a general tracking framework is developed within which the identified aspects of the target are adaptively associated to subsets of a bank of LP trackers. This results in the partitioning of LP's and the online creation of aspect specific LP flocks that facilitate tracking through significant appearance changes.

Another contribution of this work is the development of a generic approach for learning from experience in the perception-action domain. This work introduces a novel realisation of an embodied and embedded agent that organises its perceptual space and hence its cognitive process based on interactions with its environment. This is achieved by the use of an imitation training framework and by the development of a novel hierarchical representation of a set of input-output exemplars. The *exemplar hierarchy* facilitates fast searching, generalises over the exemplars, identifies important variance and invariance in the input space and provides, in a single structure, a mechanism for classifying unseen inputs and generating appropriate outputs. The organisation of the hierarchy is driven by the structure of the output space and supports Granlunds postulate quoted in the introduction to this chapter.

The novel percept-action data mining methodology proposed in chapter 6 combines the mechanism of organising the percept space using the structure of the action space with efficient data mining algorithms and feature configuration encoding schemes. The approach is able to discover the visual entities that are important to a specific problem,

and to map from these entities onto the action space. This is achieved by finding strong associations between modes of the action space and configurations of features in the percept space. The system requires no explicit definition of behaviour, uses no prior model of the objects of interest to the task and no supervision, other than the provision of input-output exemplars in the form of images and actions i.e. recorded experiences that exhibit the desired behaviour.

This thesis also contributes a strengthened case for embodiment. The self organising, action driven processes of perceptual category generation and exploitation developed in chapters 5 and 6 demonstrate an applicability for a number of the tenets of the theory of embodied cognition.

1.4 Thesis Overview

In chapter 2, the historically changing objectives and scope of the fields of *Computer Vision* are reviewed. Motivations behind a paradigmatic shift from *Cognitivism* to *Emergent Systems* are presented and related to the emergence of the field of *Cognitive Vision*. Relevant subfields of *Machine Learning*: reinforcement, supervised, semi-supervised, weakly supervised and unsupervised learning as well as case based reasoning are reviewed in section 2.1.3. In section 2.2 the philosophy and neuropsychology of *embodiment* are briefly reviewed, along with reviews of artificial intelligence approaches adopting an embodied paradigm. A literature review of *visual tracking* approaches is presented in section 2.4, with particular focus on registration and regression approaches and appearance modeling techniques. Chapter 2 ends with a review of *visual servo control* strategies in section 2.5.

Chapters 3 and 4 are concerned with the development of fast visual feature tracking algorithms that utilise no prior model (hard coded or learned) of the target appearance. Chapter 3 focuses on efficient displacement estimation functions and the techniques used to learn these functions online. In section 3.1 the registration and regression approaches to displacement estimation are detailed and compared and a *Linear Predictor* is developed that provides an efficient and powerful learning mechanism. Also the behaviour of *flocks* of LPs are investigated. Experimental results that characterise and

evaluate the proposed displacement estimator are presented in section 3.2. Section 3.3 concludes the chapter.

The development of methods for the unsupervised learning of appearance models to facilitate efficient, robust model free tracking is the focus of chapter 4. An overview of the proposed *Simultaneous Modeling and Tracking* framework is presented in section 4.1. Various configurations of a generic tracking framework are presented that incorporate methods for identifying *aspects* of the target and associating them to the displacement estimators for which they work best. The proposed trackers are evaluated, in terms of positional accuracy, efficiency and robustness, on a number of challenging video sequences. Comparisons are made with other state of the art simultaneous modeling and tracking approaches.

Chapter 5 introduces a generic problem solving architecture for the percept-action domain. An overview of the proposed approach to modeling the Percept Action Cycle (PAC) is given in section 5.1. The PAC model essentially organises experiences - examples of instances of percept-action interaction - in such a way as to reuse a previous experience in order to solve the current problem. This is achieved by hierarchically partitioning the search space, as detailed in section 5.1.1, and generalising over the exemplars, as in section 5.1.2. This constitutes a mapping from the percept to the action space. The *Exemplar Mapping* approach is developed in section 5.2. Experimental setup and results are presented in section 5.3.

Chapter 6 presents a novel percept-action data mining methodology for discovering the visual entities that are important to a given robotics task and utilising these perceptual representations to imitate the behaviour that is demonstrated by a teacher. Section 6.1 provides details of the robotic platform used to collect the training data and test the system and then presents an analysis of the collected training data. Section 6.3 describes the central mechanism of action space clustering and how this identifies classes of actions and percept groupings. Section 6.4 presents a complete overview of the proposed system, identifying the key processing stages involved, which are presented in detail in sections 6.5 and 6.6. Section 6.5.1 details the approach used to encode visual information as feature configurations and sections 6.5.2 and 6.5.3 detail the method of

finding associations between classes of actions and these feature configurations. Section 6.6 details how mappings are learnt between associated percept and action data and how these mappings are exploited to generate responses to novel image data. Section 6.7 presents the experimental evaluation of the system and section 6.8 contains a discussion and conclusions for this chapter.

Chapter 7 presents discussion and conclusions from the thesis and various avenues of future work are discussed in section 7.2.

Chapter 2

Background

This chapter provides the relevant background and context to the work carried out in this thesis. The field of computer vision is introduced and some of the perceived shortcomings of the earlier problem definitions and approaches are detailed, in particular focusing on problems associated with the cognitivist paradigm, such as the symbol grounding problem. The shift towards the emergent systems paradigm is then detailed with a review of the research field. Machine learning algorithms have been widely adopted by computer vision researchers and so a number of relevant subfields are reviewed including reinforcement learning, supervised and unsupervised learning, case based reasoning and also association rule mining. Concepts relating to embodiment are reviewed in section 2.2. Other related areas such as visual tracking and visual servo control are given in sections 2.4 and 2.5 respectively.

2.1 Classical and Cognitive Computer Vision

Computer vision has always been strongly related to, and influenced by, Artificial Intelligence (AI) and Cognitive Science. Early definitions of computer vision, therefore, adhered to the prevalent paradigm of cognitive science, *cognitivism*. Over the past couple of decades alternative approaches to cognition have appeared leading to a paradigmatic shift from cognitivism to *emergent systems*. This shift has been reflected by a

shift in the definition of the problem of computer vision and so motivates new research directions.

2.1.1 The Cognitivist paradigm

In the introduction to this thesis, computer vision was broadly defined as 'the problem of interpreting images with a computer in order to achieve some goal'. Almost 30 years ago Ballard and Brown presented a widely accepted definition of computer vision:

> *"Computer Vision is the construction of explicit, meaningful descriptions of physical objects from images. Image processing, which studies image-to-image transformations, is the basis for explicit description building. The challenge of Computer Vision is one of explicitness. Explicit descriptions are a prerequisite for recognising, manipulating, and thinking about objects."* [6]

Ten years after this definition was given - and after the computer vision community had been heavily influenced by research into Artificial Intelligence advocating an approach to reasoning, based on *explicit* representation of common sense and application-specific knowledge, Haralick and Shapiro put forward the following definition:

> *"Computer Vision is the combination of image processing, pattern recognition, and artificial intelligence technologies which focuses on the computer analysis of one or more images, taken with a singleband/multiband sensor, or taken in time sequence. The analysis recognises and locates the position and orientation, and provides a sufficiently detailed symbolic description or recognition of those image objects deemed to be of interest in the three-dimensional environment. The Computer Vision process often uses geometric modeling and complex knowledge representations in an expectation or model-based matching or searching methodology. The searching can include bottom-up, top-down, blackboard, hierarchical, and heterarchical control strategies."* [36]

Within each of these definitions there are the requirements for the construction of **meaningful** descriptions of objects **deemed to be of interest**. The questions then arises, to whom must these descriptions be meaningful, and who decides what is of interest? For many practical applications it is possible for the engineer to define a symbol set representing the objects that are of interest and to select the features that describe those objects satisfactorily for the task at hand - this approach to computer vision is consistent with the *Symbol System Hypothesis* [12].

The symbol system hypotheses states that intelligence operates on a system of symbols [12]. In a symbolic system the reasoning engine operates domain independently on sets of symbols representing inputs (perception) and outputs (actions). In such a system, where the symbolic representations, and hence semantics, are defined by the engineer, the agent itself will not necessarily (or perhaps, will never) *understand* the symbol set - this is commonly referred to as the *symbol grounding problem* [37].

In [72] Taddeo and Floridi review fifteen years of research concerned with solving the symbol grounding problem (SGP). They introduce the *zero semantical commitment condition* - that the semantic interpretation of the symbols must be *intrinsic* to the system i.e. do not rely on symbols having meaning to an observer - as a necessary requirement for any hypothesis seeking to solve the SGP. Three broad approaches to solving the SGP are identified: representationalism, semi-representationalism and non-representationalism. In conclusion, the authors use the zero semantical commitment condition to argue that no solution to the SGP currently exists. However, when discussing the non-representationalist approaches such as the *Physical Grounding Hypothesis* proposed by Brooks [12], the authors indicate that the SGP is avoided rather than solved. The argument being that physical grounding systems do not initially need to solve the SGP in order to deal with their environment; but if it is to develop any form of language or logical reasoning capabilities, then symbol manipulation will become necessary and the question of the symbols semantic grounding presents itself anew. This argument is countered by the emergent systems paradigm by a restating of what cognition entails. From the cognitivist perspective, cognition comprises computational operations defined over symbolic representations. Cognition from the emergent systems perspective is the process leading to an autonomous system operating effectively within

its environment [77].

Whilst much of this discussion may seem esoteric or even irrelevant to an engineer attempting to construct a cognitive agent, there is a practical reason for it's inclusion here.

> *"It comes down to a simple choice of axioms upon which to build a cognitive vision system. Is the role of cognition to abstract objective structure and meaning through perception and reasoning? Or, is it to uncover unspecified regularity and order that can then be construed as meaningful because they facilitate the continuing operation and evolution of the cognitive system?"* [77]

2.1.2 The Emergent Systems paradigm

In philosophical terms, emergence is the way complex systems and patterns arise out of a multiplicity of relatively simple interactions. That is to say the system is more than the sum of its parts and is therefore in contrast with the reductionist perspective. There are a number of psychological theories that adhere to the emergent systems view, that complex behaviours emerge out of interactions with their environment, these include enactivism [76] and embodied cognition [48]. Within artificial intelligence, the emergent paradigm encompasses, amongst others, connectionist systems [43], dynamical systems [75, 73] and enactive systems [76]. A common characteristic of emergent systems is their ability both to model the world and to organize themselves by interacting with their environment.

Sloman and Chappell argue that not all behaviours must be emergent, that in biological systems there exists a continuum from precocial (innate) to altrical (learnt) skills, and relating to artificial systems, that it may be useful to combine hard-coded, innate skills with mechanisms to develop new behaviours by interaction with the environment when designing cognitive systems [69]. In a general treatise on cognitive vision and the cognitivist/emergent debate, Vernon talks of the phylogeny/ontogeny trade-off [77]. Ontogenic development (learning through co-determination with environment) being

allied with the emergent, altrical behaviours and phylogenic configuration (hard-wired knowledge or capabilities) describing precocial skills and associated with cognitivist or knowledge based AI approaches. The work in this thesis tends towards ontogeny rather than phylogeny as a design philosophy.

A potential downside to the emergent paradigm is the need for continuous exploratory activity that can result in unpredictable or unwanted behaviour. It should also be noted that in cognitivist approaches, the knowledge with which the agent makes its decisions can be easily understood by the engineer, whereas, in emergent systems it can be difficult to interpret the learnt knowledge and/or mappings as they are grounded in the agents own experiences as opposed to the experiences of the engineer.

2.1.3 Machine learning

A key component of any cognitive approach is that of learning. Dictionary definitions of *learning* include "A change in behaviour as a result of experience" and "the cognitive process of acquiring skill or knowledge". For a definition of machine learning, Nilsson broadly states that "*a machine learns whenever it changes its structure, program, or data (based on its inputs or in response to external information) in such a manner that its expected future performance improves.*"[58].

Nilsson puts forward a strong case for the usefulness of machine learning in which it is stated:

> *"Some tasks cannot be defined well except by example; that is, we might be able to specify input/output pairs but not a concise relationship between inputs and desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and thus suitably constrain their input/output function to approximate the relationship implicit in the examples"* [58]

The approach of using input/output exemplars to aid approximation of input/output mappings is a central theme of this thesis and can be achieved in a variety of ways.

Given some general relationship is already known (phylogeny) that delineates the space of possible mappings, the best mapping may be learnt by some form of parameter tuning or optimisation procedure. To achieve this, some form of reward (or cost) function must be available to the system. In reinforcement learning approaches, the aim is to determine the best actions to take so as to maximize some long-term reward. Reinforcement learning algorithms attempt to find a policy that maps inputs (states of the world) to actions. Jodogne and Piater inroduced a method for learning from visual percepts to actions that combined a reinforcement learning strategy with a method of discretizing the perceptual space by isolating aliased perceptual classes [42]. Perceptual aliasing ([80]) occurs when percepts requiring different actions are classified as being the same.

Machine learning approaches can be sub-categorised based on the nature and level of supervision to the learning process. For reinforcement learning strategies, the supervision is in the design of the reward function. Unlike reinforcement learning, other common machine learning approaches provide a number of training examples. In a *supervised learning* approach, the learning algorithm aims to reason from input-output examples supplied to the system in order to produce general hypotheses, which can be used to generate outputs given unseen inputs [47]. For classification tasks, the output space is a class label. If the output is a continuous value or vector space, the supervised learning task is called regression.

For *unsupervised learning* or *clustering* approaches, no output examples are given in the training data and the task is to discover the structure of the input space i.e. to form clusters that identify "natural groupings of the input data" [22]. Unsupervised learning tasks tend to make some assumptions about the nature of the distribution density of the input data, for example, assuming a Gaussian distribution of the data means that the task can be reduced to that of identifying the mean and variance of the data. Other examples are found in some clustering algorithms where the number of modes in the data is assumed to be known [52].

The exemplar approach to machine learning as suggested by Nilsson in the above quote [58], is analogous to the Case Based Reasoning (CBR) approach where the expertise

and knowledge base of the system are provided by a memory of *cases* recording specific prior episodes. In [49], Leake provides a good introduction to the key principles of CBR. The basic general algorithm for CBR methods is:

1. Retrieve the most similar case(s) comparing current case to past cases.

2. Reuse the retrieved case to try to solve the current problem.

3. Revise and adapt the proposed solution if necessary.

4. Retain the final solution as part of a new case.

When CBR is used for problem solving [46], the goal is to use a prior solution to a previous problem (a stored case) in order to generate a new solution to the current problem (current case). A case is "a contextualised piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goal of the reasoner" [46]. How a case is actually represented in a CBR system can depend on the problem context, though there are generally three main parts: A *problem description* (the state of the world and the associated problem), a *solution* (the response or action taken to overcome problem) and an outcome (the world state, post application of the solution) [1]. Aamodt and Plaza describe the above algorithm as the CBR cycle and define the five core problem areas that must be resolved in order to develop a CBR method [1]. These are *knowledge representation, retrieval methods, reuse methods, revise methods* and *retain methods.*

Data mining - the process of extracting patterns from data - is a term used to describe a number of data analysis techniques and has a large overlap with the fields of pattern recognition and machine learning. Typical data mining tasks include clustering (unsupervised learning), classification, regression and association rule mining. Association rule learning is a method for discovering interesting relations between variables in large databases.

2.2 Embodiment

Embodiment, a term used within psychology, philosophy, computer science, robotics and artificial intelligence, is based on the premise that the nature of the mind is determined by the physical body of the cognitive agent. The theory of embodied cognition (or the embodied mind theory) states that all aspects of cognition, from perceptual and motor control systems to categorisation, thought and imagination, are shaped by the body.

2.2.1 Philosophy of embodiment

In 'Philosophy in the Flesh: The Embodied Mind and its Challenge to Western Thought', Lakoff and Johnson present a complete philosophy of embodiment based on the findings of cognitive science [48]. The argument being, that the nature of all human cognitive processes are determined by the form of the human body. These cognitive processes include categorisation and it is argued that aspects of the human body such as the perceptual system and the motor control system directly determine the nature of the categories, thoughts and ideas that a human being is capable of producing [48].

The implications of this argument are that thoughts and categories can no longer be thought of as as a purely intellectual matter, occurring after the fact of experience. Rather that the formation and use of categories is what constitutes experience. This brings into question many assumptions made by mainstream western philosophy. Three of these challenged assumptions are of immediate interest to this work: Firstly, that the categorisation of reality as perceived by a human is a reflection of a *true external reality* in which those categories exist, independently of the human. Secondly that there is a *universal reason* or structure to reality that defines the relationships between the human-independent categories. Finally that human reason is defined by universal reason and therefore the structure of human reason is disembodied. These three assumptions are consistent with *Cartesian dualism* that claim that mental phenomena are non-physical.

This dualistic stance is reflected in the cognitivist approach to AI in which reasoning

is carried out in some abstracted, disembodied symbolic domain. The embodied mind theory refutes the existence (or at least human capacity to conceptualize) of a true external reality. Instead it proposes that categories and the relationships between them are all dependent on the human body and that cognition is the process of forming and using those embodied categories.

2.2.2 Neuropsychology of embodiment

The subject of how actions are executed and represented has been the focus of much recent experimental research from within the fields of cognitive psychology and cognitive neuroscience. Experimental cognitive psychology has developed experimental methods that involve measuring response times for perceptual motor tasks in order to analyse the underlying cognitive processes and has applied these to both overt (executed) and covert (not executed) actions [10]. These experiments demonstrate that action and perception are coded in the same cortical system. Cognitive neuroscience has introduce functional brain imaging techniques, such as functional Magnetic Resonance Imaging (fMRI) as well as brain simulation techniques that both yield descriptions of the neural structures involved in representing actions [15]. This experimental paradigm has resulted in the discovery of mirror neurons [19, 63] which appear to be involved in both the execution of actions and perception of actions performed by others.

An important result drawn from these experimental paradigms, pointed out by Marc Jeanerod in 'Motor Cognition: What Actions Tell the Self', is that 'the observation of actions performed by other agents generates in the brain of the observer representations similar to those of the agents' [40]. This is termed in some literature the 'direct-matching hypothesis' and is facilitated by mirror neurons [26]. This result is important to this thesis in two respects. Firstly it motivates an imitation based behaviour learning process. If it is possible to map the actions of others onto ones own action representation, then mimicking actions becomes possible and may provide a foundation for general imitation and eventually autonomous behaviour. The second reason why this result is relevant concerns the process of perception itself. When discussing the perception of biological motion, Jeanerod points out that 'not only executing, but also

perceiving biological movements is influenced by in-built features of motion generation. In other words, perception seems to obey motor rules'.

2.2.3 Embodiment in artificial cognitive systems

In [14] Brooks presents a number of papers introducing what he terms *behaviour-based* robotics in which he argues the case for the *symbol grounding hypothesis* over the *symbol system hypothesis.* Brooks promotes the study of intelligence from the bottom up, concentrating on physical systems (e.g., mobile robots), situated in the world, autonomously carrying out tasks of various sorts. Embodiment is critical, he argues, because only through a physical grounding can any internal symbolic or other system give 'meaning' to the processing going on within the system [13]. In [12], he puts forward an argument for embodied, bottom up intelligence - the physical grounding hypothesis - based on observations from evolutionary biology. He states that evolution has devoted the vast majority of time in developing the ability to move around in a dynamic environment, sensing the surroundings to a degree sufficient to achieve the necessary maintenance of life and reproduction. By contrast, language, expert knowledge and reason should, in terms of the time spent in evolving such capabilities, be an easier task. A number of physically grounded agents have been developed using the subsumption architecture, an architecture constructed from layers of finite state machines augmented with timing elements that connect perception to action. Many of these are reviewed in [12].

Related to embodiment and physical grounding is *affordance theory* - introduced by Gibson [29]. Affordance theory states that the world is perceived not only in terms of object shapes and spatial relationships but also in terms of object possibilities for action (affordances). Relating this to embodiment, it can be argued that agents with different physical embodiments will perceive the world differently, as the affordances are determined by the agents possibilities for action - it's embodiment. Amongst the many researchers to explore affordance-based robotic systems, Paletta & Fritz have recently demonstrated an affordance-based reinforcement learning approach for learning of causal relationships between visual cues and associated anticipated interactions [60].

Another concept being explored by AI researches, that relates to the theory of affordances, is that of learning functional object-categories. That is developing systems that learn to recognise objects based on the role they play in carrying out activities. Work based on object recognition is almost exclusively based on learning categories based on perceptual features. Sridhar et al. recently presented a method for learning categories from function. Object-categories are learnt from relational spatio-temporal activity graphs extracted from video sequences displaying activities involving a toy kitchen set. In this work, object discovery is performed by clustering in a candidate object space, where the similarity function is based on similarity of the objects interactions [71].

2.3 Imitation Learning

Imitation plays a strong role in the development of cognitive systems. As mentioned above, recent neurophysiological research has shown strong evidence supporting the existence of a mechanism, in both primate and human brains, known in the literature as the direct-matching hypothesis. The mirror-neuron system essentially provides the system (human/primate brain) with the capability "to recognise actions performed by others by mapping the observed action on his/her own motor representation of the observed action" [16]. An agent that can recreate, in its own action representation, physical tasks carried out by another agent can mimic those tasks simply by overtly carrying out actions that have been obtained through the direct-matching process. Mimicry provides a fundamental starting point to achieving more complex imitation, for which some comprehension of purpose or goal state is required.

In the work of Siskind, an attempt is made to analyse from visual data the force dynamics of a sequence and hence deduce the action performed [68]. Alternatively, Fitzpatrick have shown that it is possible for an agent to learn to mimic a human supervisor by first observing simple tasks and then, through experimentation, learning to perform the actions that make up the tasks [25]. Both these approaches deal only with exact mimicry.

The machine learning approach of learning from a set of training examples can also be interpreted as imitative. Given a set of training examples of inputs and associ-

ated decisions/outputs the objective is to imitate the process behind generating those decisions/outputs.

Related to learning by imitation, is Learning from Demonstration (LfD). In [4], Argall et al. present a comprehensive survey of robot LfD, a technique that develops policies from example state-to-action mappings. This work included an analysis and categorisation of the multiple ways in which examples are gathered, ranging from teleoperation to imitation, as well as the various techniques for policy derivation, including classification, regression and reward function. Within this framework, the work presented in this thesis uses teleoperation - where the robot is operated by the teacher while recording both the control signals and the sensor readings - to gather the examples. Both classification and regression approaches are employed for policy derivation, in chapters 5 and 6 respectively.

2.4 Visual Tracking

Visual tracking is the process of consistently locating a desired feature in each of a sequence of images. The problem is typically complicated by noise from the sensor, motion in the scene, motion on the part of the observer, (partial) occlusions and appearance changes of the target feature and real-time constraints. Yilmaz *et. al* introduced a taxonomy of tracking methods [82]. Within this taxonomy, the tracking approach proposed in this thesis falls into the class of methods identified as multi-view kernel methods. However, unlike the methods reviewed, this approach learns the views of the target online.

2.4.1 Tracking via registration

Lucas and Kanade made one of the earliest practical attempts to efficiently align a template image to a reference image [51], minimising the Sum of Squared Difference similarity function. Efficiency was achieved by using a Newton-Raphson method to traverse the space of warp parameters. In Newton-Raphson optimisation, iterative parameter updates to alignment parameters are obtained by multiplying the Jacobian

by the inverse Hessian of the similarity function. Lucas and Kanade mainly considered translations, but they demonstrated that any linear transformation could be used.

Later research considered more complex transformations and attempted to reformulate the similarity function allowing pre-computation of some terms. In particular, Hager & Belhumeur proposed inverting the roles of the reference and template at a strategic point in the derivation [34], and Shum et al. constructed the warp as a composition of two nested warps [67]. In a general treatise on Lucas-Kanade (LK) techniques, Baker & Matthews combined these methods to formulate the inverse-compositional method [5] . Dowson & Bowden derived an inverse compositional formulation for aligning a template and a reference image using mutual information and Levenberg-Marquardt optimisation [21].

2.4.2 Tracking via regression

Cootes et al. [18] proposed a method for pre-learning a linear mapping between the image intensity difference vector and the error (or required correction) in AAM model parameters. Jurie & Dhome employed similar Linear Predictor (LP) functions to track rigid objects [44]. The work of Matas et al. again uses linear regression for displacement prediction [55], similar to the LP functions in [44] and [18]. They extend the approach by introducing the Sequential Linear Predictor (SLP) [83]. Williams et al. [81] presented a sparse probabilistic tracker for real-time tracking that uses an Relevance Vector Machine (RVM) to classify motion directly from a vectorised image patch. The RVM extends the method of forming a regression between image intensity difference vectors and the error/correction to non-linear regression.

A key issue for LP trackers is the selection of its reference point, i.e. its location in the image. In the work of Marchand et al. [53], predictors are placed at regions of high intensity gradient but Matas et al. have shown that a low predictor error does not necessarily coincide with high image intensity gradients [55]. In order to increase efficiency of the predictors, a subset of pixels from the template can be selected as support pixels used for prediction. Matas et al. present a comparison of various methods for learning predictor support, including randomised sampling and normalised reprojection, and

found that randomised sampling is efficient with minimal and controllable tradeoff in terms of accuracy [55].

Ong & Bowden recently proposed a real-time facial feature tracker based on the biased linear predictor, that includes a bias term in the regression function that improves performance especially when using multiple training images [59]. Multiple linear predictors are grouped into a rigid flock to increase robustness and a probabilistic selection method is used to identify tracking reference points. The model in [59] also extends the LP tracker to a multi-resolution model.

2.4.3 Online appearance model learning

Tracking approaches typically employ appearance models in order to optimise warp parameters (e.g. translation or affine) according to some criterion function. Linear predictor trackers typically rely upon hard coded models of object geometry [55, 53]. This requires significant effort in hand crafting the models and like simple template models [51, 5, 56], are susceptible to drift and failure if the target appearance changes sufficiently. Systems that use a priori data to build the model [18] or train the tracker offline [81] can be more robust to appearance changes but still suffer when confronted with appearance changes not represented in the training data. Incremental appearance models built online such as the WSL tracker of [41] have shown increased robustness by adapting the model to variations encountered during tracking, but the overhead of maintaining and updating the model can prevent real-time operation.

Two recent approaches that achieve real-time tracking, and have adopted an entirely online learning paradigm, are the discriminative tracker of [31] that uses an online boosting algorithm to learn a discriminative appearance model on the fly and Dowson & Bowden's SMAT algorithm. Dowson & Bowden present the Simultaneous Modeling And Tracking algorithm, SMAT, and show the benefits of online learning of a multiple component appearance model when employing alignment-based tracking [20].

2.5 Visual servo control

Sanderson and Weiss have introduced a taxonomy of visual servo control architectures [65]. From this taxonomy we can draw four major categories formed by two modes of distinction. The first distinction is between *dynamic look-and-move* and *direct visual servo* control mechanisms. This distinguishes between systems that use joint feedback to help with stability (dynamic look-and-move) and systems that directly compute servo positions (direct visual servo) using vision. The second distinction is between *position-based* and *image-based* methods. In position-based control strategies the pose of the target is estimated by matching extracted image features to a geometric model of the target using a known (calibrated) camera model. The error between desired and estimated pose drives the control signals. In image-based servo control, the control signal is computed directly from the image features. A third distinction between visual servo control approaches concerns the placement of the camera. Generally the camera is either mounted on the end-effector - *eye-in-hand* - or with both the end-effector and the target in view [38].

Whilst much of the visual servo control literature focuses on manipulating a robot arm or *end effector positioning*, much of the theory relates equally to other applications, such as autonomous navigation. The visual servo control approach developed in chapter 6 is applied to the task of autonomous navigation of a robotic vehicle. The strategy can be classified as an eye-in-hand image-based visual servo (IBVS) control strategy. That is to say the signals that control the vehicles motion are computed directly from features extracted from images captured by a camera mounted on the vehicle.

Commonly eye-in-hand position-based (PBVS) and image-based visual servo control (IBVS) strategies utilise a priori knowledge of the 3D model of the target. ([79]; [38]). This is transparent in position-based methods that by definition require a perfect geometric 3D model of the target. For image-based methods this knowledge is generally employed in the estimation image Jacobian, sometimes called the interaction matrix. This matrix defines the relationship between motion of 2D image features and the 3D motion of the camera. Generally pose estimation algorithms that require knowledge of the 3D geometry of the target are used to estimate the interaction matrix [24].

The term target has been used throughout this background to visual servo control. For many applications the target is a single object to which the end-effector should be in some way aligned. For example a robotic arm moves to grasp/inspect/deflect a cup/unknown-object/ball. Another example would be a visual servo controlled auto-pilot undergoing a landing. In this case the target is a more abstract combination of features e.g. the runway and the line of the horizon.

2.6 Robotic systems for autonomous navigation

The development of robotic systems for autonomous navigation has become a major research area in recent years. Interest in the topic has been stimulated, in particular, by the well publicised DARPA Grand Challenge. The DARPA Grand Challenge is a competition for autonomous vehicles, funded by the Defense Advanced Research Projects Agency. To date there have been three competitions, in 2004, 2005 and 2007. Many of the competition entries use multiple sensor inputs, often relying heavily on laser perception and GPS navigation. One of the entries that is of particular interest to this work is the winner of the 2005 challenge - Stanley [74].

Stanley (the winner of the 2005 DARPA grand challenge) was designed to achieve high-speed desert driving without manual intervention. Interestingly, the robot relied heavily on state-of-the-art machine learning and probabilistic reasoning technologies. Of particular relevance to this work is the use of labeled training data, obtained from a human driver, for learning a laser terrain mapping. The vehicle is equipped with five single-scan laser range finders. This sensor data is then used to label the terrain in front of the vehicle as either drivable, occupied or unknown. The mapping from laser data to a terrain labeling is achieved using a labeled training data, provided by a human driver. A human driver is instructed to only drive over obstacle-free terrain. Regions traversed by the vehicle are then labeled as drivable, and regions either side of the traversed regions are labeled as containing obstacles. Although not all the regions labeled as containing obstacles by this method are actually occupied by obstacles, training with this approximate labeling was found to be achieve sufficient accuracy [74]. A similar approach of using human driver data is also used for learning the velocity control. To

further extend the range of the drivable surface classification, an adaptive computer vision algorithm is employed. This algorithm extends the range from approximately 22m in front of the vehicle (from the laser data) to around 70m. Drivable surfaces are found by projecting the drivable area from the laser analysis into the camera image. Pixels in the camera image that are labeled as drivable by the laser analysis are used to train a Gaussian Mixture Model (GMM) in the red/green/blue colour space, to classify the surface in front of the vehicle as either drivable or non-drivable. The GMM adapts to the data online, by both adjusting the parameters of the Gaussian models and by introducing new Gaussian models. This allows for adaptation to slowly changing lighting conditions as well as rapid changes to surface colour (e.g. when the vehicle moves from paved to unpaved road).

Another autonomous navigation system worth mentioning is Trinity Colleges ALVIN robot, an autonomous ground vehicle that has participated in the Association for Unmanned Vehicle Systems International Intelligent Ground Vehicle Competition (IGVC) since 2000 [9]. The vehicle was developed to be able to navigate along a path (two white lines marking the path boundaries) and to avoid obstacles. Essentially this was achieved by using image thresholding to identify the white lines immediately in front of the vehicle. The lines were then projected to the horizon. Moving parallel to both lines meant the centerline of the robot had to meet both lines at the horizon, i.e., the intersection of two lines had to be at the center of camera view. Obstacles were detected using an ultrasonic sensor array. The design evolution of ALVIN is interesting, not so much due to successes, but more as it highlights the many and varied difficulties with developing autonomous navigation systems. Over the five generations of ALVIN, significant failures have occurred due to: short circuit in the motor control system, lack of ability to adapt to varying lighting conditions, insufficient obstacle representations, overheating computers and programming bugs. In developing the autonomous navigation system proposed in this thesis, many of these problems were also encountered, and effective solutions are developed.

2.7 Motivating discussion

This chapter provides a background to the approaches presented within this thesis. This concluding discussion highlights aspects of this background discussion that motivate particular aspects of the proposed approaches.

In the section reviewing machine learning, various strategies for supervising the learning tasks are reviewed. In chapter 3, a supervised machine learning technique is used to form regression functions from image pixel intensities to target pose parameters, however the supervision to the learning process is provided autonomously by the system.

In chapter 4, another machine learning approach, incremental unsupervised learning, is applied to the problem of discovering aspects of a target object during tracking. In chapters 5 and 6 unsupervised learning approaches are applied to the response space and the resulting clustering is used to organise the perceptual space.

Although not initially motivated by the CBR approach, the system presented in chapter 5 has many characteristics in common with a CBR system. Also, the problems faced when attempting to construct the system can be viewed as direct analogies to those stated by Aamodt and Plaza above.

Chapter 6 uses association rule mining in order to discover feature configurations that frequently co-occur with a particular action-type, and not all other action-types.

It is the objective of chapters 5 and 6 to develop mechanisms that allow the systems to organise the process of cognition - demonstrated by operating effectively within its environment. This is consistent with the emergent systems definition of cognition. Chapters 5 and 6 also demonstrate novel ways in which the *experience* - process of forming and using categories - of the agent is determined by the embodiment of the agent and by the interactions the agent has with its environment. This is consistent with the theory of embodiment. The neuropsychological findings reviewed highlight the concept of perception being determined to a certain extent by the structure of the motor space and this further motivates the action space clustering approach developed in chapters 5 and 6.

The perceptual categorising and response generation systems developed in chapters 5

and 6 can be related to affordance theory and the functional object category learning approach reviewed. The use of a similarity function based on similarity of the objects interactions rather than appearance being similar to the use of a similarity function applied in the action rather the the percept domain.

The final two sections of this chapter (sections 2.4 and 2.5) review related work concerning the two application domains for the proposed approaches. As shall be shown, the proposed visual tracking approach avoids the need for costly reference point and support selection strategies employed by the linear predictor approaches reviewed. This is achieved by evaluating the performance of a predictor online and allowing poor performers to be replaced as opposed to minimising a learning error offline. Each of the displacement prediction trackers detailed in [55, 53, 81, 59] require either an offline learning stage or the construction of a hard coded model or both. The work presented in this thesis does not require either hard coded models or offline learning.

The approach presented in chapter 6 differs significantly from conventional visual servo control systems, even from most eye-in-hand IBVS approaches. As shall be shown no 3D representation of the target is used and the interaction matrix is not computed. Instead, visual feature configurations that are strongly associated to modes of the control space are discovered and linear functions are learnt for each of these discovered configurations, that map from feature space to control space.

Chapter 3

Linear Regression for Fast Displacement Estimation

The work in this chapter and in the next chapter of this thesis is concerned with the development of fast visual feature tracking algorithms that utilise no prior model (hard coded or learned) of the target appearance. The approach presented operates at high frame rates, tracks fast moving objects and is adaptable to variations in appearance brought about by occlusions or changes in pose and lighting. This is achieved by employing a novel, flexible and adaptive object representation comprised of sets of spatially localised linear displacement predictors associated to various modes of a multi modal template based appearance model learnt on-the-fly. This chapter focuses on the displacement prediction methods.

Many problems in computer vision have been expressed as the task of learning a mapping between image space and a parameter space [57]. The *vision as regression* paradigm has been widely adopted by a number of researchers [2, 81], as regression naturally lends itself to the task of learning input-to-output mappings. This chapter explores the use of linear regression models for tracking image features i.e. mapping from image space to target pose parameter space. The linear regression tracker, or Linear Predictor (LP), is an example of an input-output mapping that forms part of the Percept Action Cycle (PAC) framework introduced in chapter 1. Figure 3.1 illustrates this regression based approach. The outputs in this instance are target pose parameters

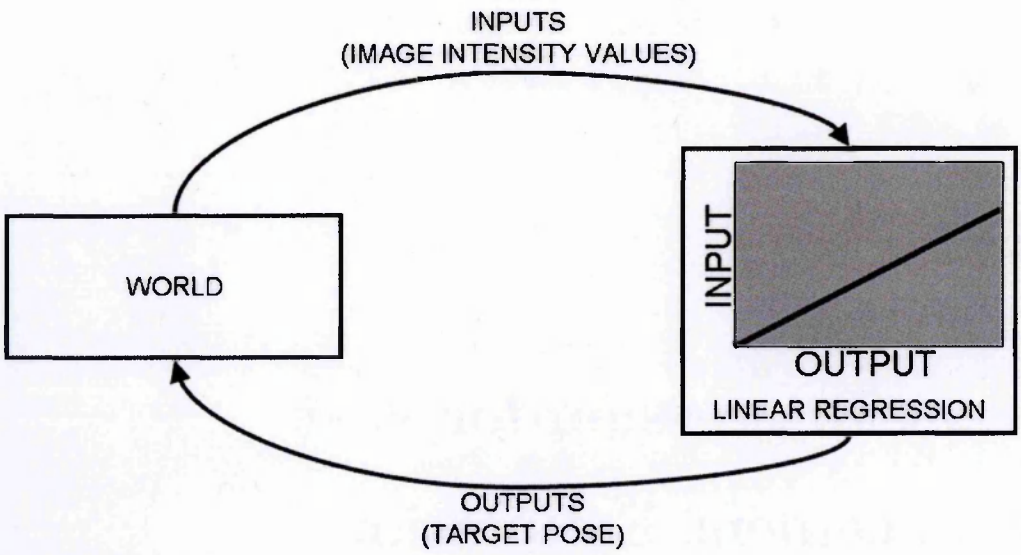


Figure 3.1: *Regression Approach to input-output mapping*: This figure represents the tracking approach proposed in this chapter within the PAC framework introduced in chapter 1.

which essentially shift the attention of the tracker i.e. determine where in the input image the pixel intensities will be sampled in the next frame.

Conventional alignment based tracking approaches aim to estimate the position of the target in each frame by aligning an image template of the target with the new frame; the template (or input frame) is *warped* in order to obtain an optimal alignment. The warp parameters are obtained by optimising the registration between the appearance model and a region of the input image according to some similarity function (*e.g.* L_2 norm, Normalised Correlation, Mutual Information). Optimisation is often carried out using gradient descent or Newton methods and hence assumes the presence of a locally convex similarity function with a minima at the true position. The *basin of convergence* of such methods is the locally convex region of the cost surface within which a gradient descent approach will converge. The size of the basin of convergence determines the *range* of the tracker *i.e.* the maximum magnitude of inter-frame displacements for which the approach will work. Trackers with small range require low inter-frame displacements to operate effectively and hence must either operate at high frame rates (with high computational cost) or only track slow moving objects. If the target moves a distance

greater than the range between two consecutive frames then the method will fail. While multiscale approaches can be used to address this in registration approaches, regression based tracking allows the user to select the optimal range as a trade-off against accuracy and will be experimentally shown to have a greater range (not limited by the range of convexity or the presence of local minima in the cost surface) than registration methods and due to their simplicity are computationally efficient. The computational efficiency of the method is a result of learning a simple and general mapping directly from patterns of image intensity differences to desired displacements, and applying this mapping at each displacement prediction step, rather than performing an optimisation process for each prediction step.

The computational efficiency of the LP regression based tracking methods developed allows for the use of multiple LPs. The behaviour of constellations or *flocks* of these LPs is investigated along with methods for weighting each LP's contribution to the overall flock output based on it's performance. Approaches to evaluating the performance of a LP within a flock are introduced.

The rest of the chapter is organised as follows: Section 3.1 formulates the tracking problem and compares regression and registration approaches to predicting inter-frame displacement of a target object for tracking. The LP regression tracker is introduced in section 3.1.2 and the method used for learning the LP regression function is detailed in section 3.1.3. This is followed in section 3.1.4 by a description of methods of combining multiple LPs into LP-flocks. In section 3.1.5 some illustrative experimental results are presented that compare regression and registration techniques on an example inter-frame displacement prediction. In section 3.2 experiments that characterise the proposed regression method are presented along with experiments evaluating the effect of various parameters on the positional accuracy, range and computational complexity of the approach. The predictor evaluation and flock weighting mechanisms are also evaluated.

3.1 Displacement Estimation: Registration vs. Regression

The tracking problem is defined as the task of estimating the change of pose or warp parameter, $\delta\mathbf{x}$, such that:

$$I_R(W(\mathbf{x}, \delta\mathbf{x})) \approx I_T \tag{3.1}$$

where I_R is the new input image, W is a warping function (*e.g.* translation, affine), $\mathbf{x}$ is the state vector parameterising the pose of the target and I_T is a template representing the appearance of the target.

For the LK or registration based method this is treated as a minimisation problem such that we wish to find $\delta\mathbf{x}$ that minimises the dissimilarity between I_R and I_T.

$$\delta\mathbf{x} = \underset{\delta\mathbf{x}}{\operatorname{argmin}} \, ||I_R(W(\mathbf{x}, \delta\mathbf{x})) - I_T|| \tag{3.2}$$

For the regression or Linear Prediction (LP) method, the prediction directly estimates $\delta\mathbf{x}$.

$$\delta\mathbf{x} = \mathbf{P}(I_R(W(\mathbf{x}, \mathbf{dx})) - I_T) \tag{3.3}$$

Every tracking approach has some representation of the target; tracking output is a function of both this representation and new image data. For registration methods the representation is a template of pixel intensities, I_T, drawn from the input image at the location of the target. Tracking is then the process of aligning template, I_T, with the new input reference image, I_R *i.e.* finding the warp, W, with parameters $\delta\mathbf{x}$ that minimises (maximises) some distance (similarity) function between I_T and I_R.

For the linear regression method presented in section 3.1.2 the target representation is a vector of image intensities. Additionally, the regression function, $\mathbf{P}$, encodes information about the target appearance. Tracking is then the process of multiplying $\mathbf{P}$ with

the difference between target representation vector and an intensity vector sampled from the input image at the current position.

Looking at equations 3.2 and 3.3 it is apparent that both approaches involve some operation on the difference between the target representation and the input image information. Whilst the registration method explicitly minimises the cost surface to obtain an optimal alignment, the regression method directly maps from image intensity difference patterns to required displacements. In fact, as detailed in the following section, the iterative optimisation methods used in the registration approaches involve, at each iteration, a linear operation on the intensity difference. The difference between the two methods is that the parameters of the linear function used in the iterative optimisation methods are based on cost surface gradient information, whereas for the regression methods, the parameters are learnt from examples of displacement and intensity difference patterns.

3.1.1 Tracking by registration

The registration process aims to locate the region in I_R (reference image) that most resembles I_T (template image) by minimizing a distance function, f, which measures the similarity of the two regions. The position of I_T relative to I_R is specified by a warp function W with parameters $\delta\mathbf{x}$.

$$\delta\mathbf{x} = \underset{\delta\mathbf{x}}{\operatorname{argmin}}\ f[I_R(W(\mathbf{x}, \delta\mathbf{x})), I_T(\mathbf{x})] \tag{3.4}$$

Distance function, f, can be any similarity measure, *e.g.*, L_2 norm or MI. For comparisons of the relative merits of different similarity measures see [21]. The position of greatest similarity is found using an optimisation method. LK methods use a group of optimization methods, the so-called Newton-type methods, *i.e.* methods which assume locally parabolic shape and proceed with an update as follows:

$$\delta\mathbf{x}^{(k+1)} \leftarrow \delta\mathbf{x}^{(k)} - H^{-1}(\delta\mathbf{x}^{(k)})G(\delta\mathbf{x}^{(k)}) \tag{3.5}$$

Where H, $\frac{\partial^2 f}{\partial \delta \mathbf{x}^2}$, is the Hessian of f, and G, $\frac{\partial f}{\partial \delta \mathbf{x}}$, is the Jacobian, while k indexes the iteration number. However, minima in tracking and registration problems are frequent which results in erroneous alignment of the template with the target. Multiple initializations can improve performance but at an obvious computational cost.

Generally, LK type methods apply Quasi-Newton optimisation, *i.e.* an approximation to the Hessian, $\tilde{H}$, is used. In general, Newton and Quasi-Newton only perform well when near to the minimum. Steepest Descent methods, which ignore local curvature and instead multiply G by a scalar *step-size* value λ, perform better when further from the minimum. The Levenberg-Marquardt [54] method combines these two methods. In this work a formulation similar to that presented in [21] (using Levenberg-Marquardt and L_2 norm) of this registration based tracking is used in comparisons with regression based techniques. The C++ (or Matlab) warthog library is used as an efficient implementation[1].

3.1.2 Tracking by regression: The Linear Predictor

Feature tracking by regression is achieved by predicting inter-frame displacement of the target. The displacement predictors explored here use linear models to predict. These predictors compute motion at a reference point from a set of pixels sub-sampled from its neighbourhood called the support set $\mathbf{S} = \{\mathbf{s}_1, ..., \mathbf{s}_k\}$. The intensities observed at the support set $\mathbf{S}$ are collected in an observation vector $\mathbf{I}(\mathbf{S})$. The $\mathbf{I}_0(\mathbf{S})$ vector contains the intensities observed in the initial training image. Here the motion is a 2D translation $\mathbf{t}$, we use $(\mathbf{S} \circ \mathbf{t}) = \{(\mathbf{s}_1 + \mathbf{t}), ..., (\mathbf{s}_k + \mathbf{t})\}$ to denote the support set transformed by $\mathbf{t}$. Translation is sufficient as the multi-modal appearance models developed in chapter 4 cope with affine deformations of the image templates, also shown in [20].

Predictions are computed according to the expression in Eq. (3.6) where $\mathbf{P}$ is a (2 $\times$ k) matrix that forms a linear mapping $\Re^k \rightarrow \Re^2$ from image intensity differences, $\mathbf{d} = \mathbf{I}_0(\mathbf{S}) - \mathbf{I}(\mathbf{S} \circ \mathbf{x})$, to changes in warp parameters, $\delta\mathbf{x}$. The state vector, $\mathbf{x}$, is the 2D position of the predictor after prediction in the preceding frame.

[1] Link to code found at www.cvl.isy.liu.se/research/adaptive-regression-tracking

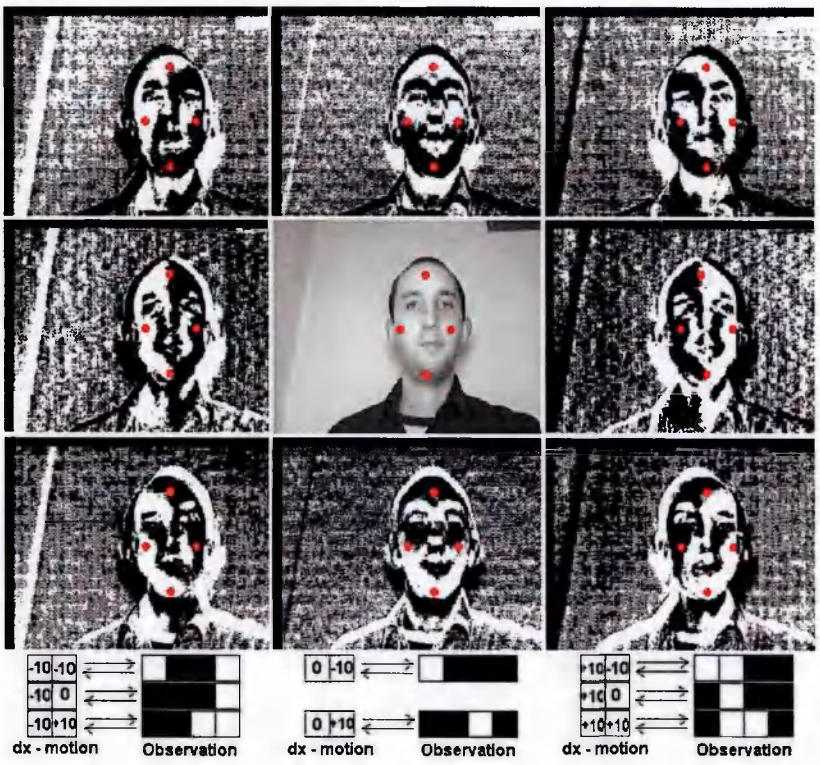


Figure 3.2: *Intensity difference images for eight translations*: Four support pixel locations illustrate the predictive potential of the difference image. The input image is in the center. All images to the left/right of the input have been translated left/right by 10 pixels. Those images above/below the input have been translated by 10 pixels up/down. Under the images, the motion and support vectors are illustrated. Note that observations at support locations are grey values.

$$\delta \mathbf{x} = \mathbf{P}\mathbf{d} = \mathbf{P}(\mathbf{l}_0(\mathbf{S}) - \mathbf{l}(\mathbf{S} \circ \mathbf{x})) \tag{3.6}$$

This efficient prediction only requires k subtractions and a single matrix multiplication, the cost of which is proportional to k.

3.1.3 Predictor learning

In order to learn $\mathbf{P}$, the linear regressor or projection matrix, N training examples of $\{\delta \mathbf{x}_i, \mathbf{d}_i\}$ pairs, $(i \in [1, N])$ are required. These are obtained from a single training im-

age by applying synthetic warps and subtracting the deformed image from the original. For efficiency, the warp and difference computation is only performed at the support pixel locations but, for illustration, the result of performing this operation on the entire image is shown in figure 3.2 for eight different translation warps. Also marked on the figure are four possible locations for support pixels and the unique observation patterns they produce.

In the proposed approach, support pixels are randomly selected from within a range, r_{sp}, of the predictors reference point. This is in contrast to other LP learning strategies [84, 59] where the objective is to select an optimal support set. The next step in learning the linear mapping $\mathbf{P}$ is to collect the training data, $\{\delta\mathbf{x}_i, \mathbf{d}_i\}$ into matrices $\mathbf{X}$, $(2 \times N)$, and $\mathbf{D}$ $(k \times N)$ where N is the number of training examples. The Least SQuares (LSQ) solution, denoted P, is then:

$$\mathbf{P} = \mathbf{X}\mathbf{D}^{+} = \mathbf{X}\mathbf{D}^{T}(\mathbf{D}\mathbf{D}^{T})^{-1} \tag{3.7}$$

Where $\mathbf{D}^{+}$ is the pseudo inverse of $\mathbf{D}$.

Clearly there are more sophisticated learning methods, both in the selection of support pixels and in the method used to solve the regression problem. However, the methods selected provide a computationally efficient solution. As shall be shown here and in the next chapter, the use of LPs with low computational cost combined with methods to rate the performance (and hence weight the contribution) of each LP allows the replacement of poorly performing LPs during tracking. This essentially spreads the cost of learning appropriate mappings over a period of time and allows incremental learning as opposed to batch (offline) learning.

The LPs have a number of tunable parameters. The parameter, r_{sp}, defines the range from the reference point within which support pixels are selected. Parameter r_{tr} defines the range of synthetic displacements used for training the predictor. Figure 3.3 illustrates the displacement prediction errors of LPs with $r_{tr} = 10$, $r_{tr} = 50$ and $r_{tr} = 80$. The predictor complexity, k, specifies the number of support pixels used and hence the dimension of $\mathbf{P}$. The number of synthetic translations used in training is denoted N. In section 3.2.2, experimental results are presented to illustrate the effect each of these

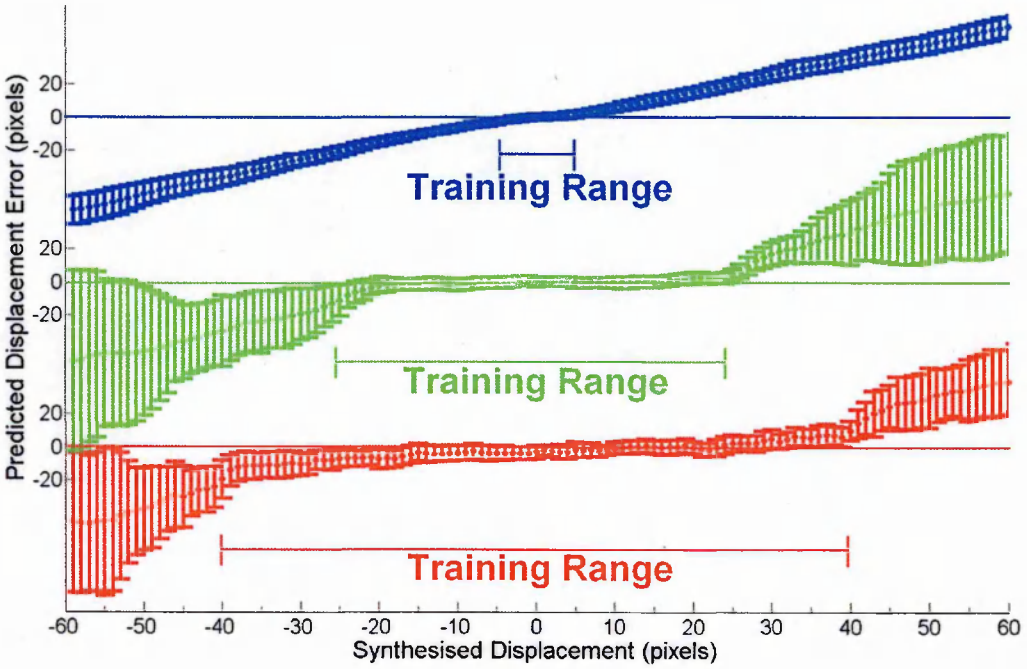
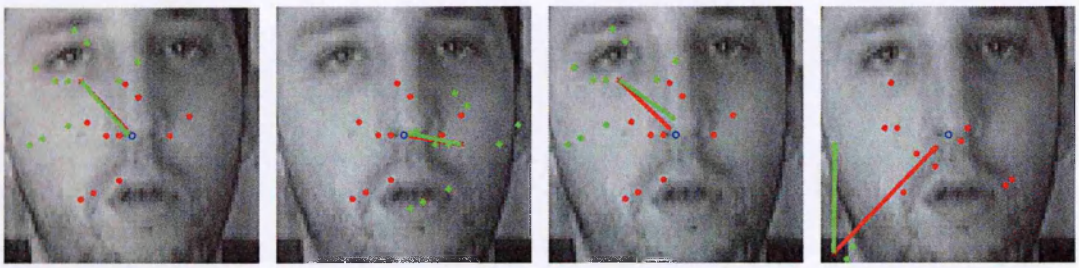


Figure 3.3: *LP response for three training ranges*: The predicted displacement error (vertical axis) versus the true displacement (horizontal axis) of three LPs is shown. The response shown in red (or dark grey in black and white) at the bottom is of a predictor trained on displacements in the range -40 to 40 pixels. The response shown in green (light grey) in the middle is of a predictor trained on displacements in the range -25 to 25 pixels and the response shown in blue (black) at the top is of a predictor trained on displacements in the range -5 to 5 pixels. It can be seen that, within the range of displacements used for training, each of the LPs achieve relatively low errors. It can also be seen, from the error bars, that whilst increasing the range of displacements used for training extends the operational range of the LP, it does so at the cost of consistency.

parameters has on tracker performance. It is sufficient to say, increasing r_{tr} increases the maximum inter frame displacement at the expense of alignment accuracy; k models the trade off between speed of prediction and accuracy/stability. N does not affect prediction speeds but instead parameterises a trade off between predictor learning speeds and accuracy/stability.



(a) Displacement (up and to the left) is accurately predicted.

(b) Displacement (to the right) is accurately predicted.

(c) Displacement (up and to the left) is less accurately predicted.

(d) Displacement (down and to the left) is larger than training range and is not predicted well.

Figure 3.4: *LP predictions*: Four examples of a LP predicting displacement on a test image are shown. In each image the reference point is marked with a blue circle, ten support pixel from the initial support set are marked with red circles, the test displacement vector is indicated by a red line, the support pixels at the displaced position are marked with green crosses and the prediction is indicated by the green line.

Figure 3.4 presents four examples of a LP predicting displacement on a single test image. The point being 'tracked' - the reference point, is marked with a blue circle. Ten of the support pixel locations that make up the initial support set, $\mathbf{l}_0(\mathbf{S})$, are marked with small red circles. For each of the four images the LP is randomly displaced, the displacement vector indicated by the red line. The support pixels at the displaced position are marked with green crosses and the prediction is indicated by the green line. The predictions generally match the displacement with reasonable accuracy, as shown in figures 3.4(a), 3.4(b) and 3.4(c), although, as shown in figure 3.4(c), the accuracy does vary. Figure 3.4(d) shows a prediction with a displacement of -36 pixels horizontally and 37 pixels vertically. As the LP is trained for a range of 20 pixels (in

both horizontal and vertical directions), the prediction error in this case is large.

3.1.4 The linear predictor flock

The displacement predictions made by LPs have limited accuracy; this is especially the case where no attempt is made to optimise support pixel selection. A simple approach to handling the noise introduced by this inaccuracy is to take the mean prediction from a collection of LPs as in equation 3.8.

$$\delta\bar{\mathbf{x}} = \frac{\sum_{l=1}^{L} \delta\mathbf{x}^l}{L} \tag{3.8}$$

The state vector, x for each of the collection of L LPs is then updated with this mean prediction, as in equation 3.9, causing the LPs to *flock* together.

$$\mathbf{x}_t^l = \mathbf{x}_{t-1}^l + \delta\bar{\mathbf{x}}, l = 1...L \tag{3.9}$$

The increase in prediction accuracy, as shown by the experiments in section 3.2, is due to the noise averaging characteristics of the mean. Similar results are/would be obtained using the median but this would complicate the weighting of LP contribution to flock output as described below. Another approach is to use the RANSAC algorithm to select the subset of LPs who's prediction gains most consensus within the flock. Although the outlier rejection of RANSAC may be better than the mean value, RANSAC has a higher computational cost and again is less well suited to weighting LP contributions to flock output.

Within a LP flock, it is desirable to down weight poor predictions or even remove/replace poorly performing LPs. This is especially the case when using the simple learning strategies detailed above. To weight the contribution a single prediction makes to the overall flock output, some way of assessing the reliability of the prediction is required - a prediction error. As no ground truth displacement is available whilst tracking, this error function could rely on observation differences at the support pixels, the assumption being that when a predictor performs well, the observations at the support pixels

- after the trackers state vector, x, has been updated - should be similar to those observed in the initial frame. Alternatively, we can consider flock output to be 'truth' and evaluate predictions based on flock agreement, *i.e.* the error is the difference between 'true' flock output and the prediction being evaluated. If a LP 'strays from the flock' it can be relied on less. This approach benefits from its computational simplicity as it requires only difference computations in the low dimensional pose space, $\|\bar{\delta\mathbf{x}} - \delta\mathbf{x}^l\|$ (t is the current frame) as opposed to in the higher dimensional observation space $\|\mathbf{l}_0^l - \mathbf{l}_t^l\|$. The observation difference error also requires additional computation for image bounds checking.

There is considerable scope for different LP flock contribution weighting strategies using either of the above prediction errors. A simple and cost effective approach is linear weighting (see equation 3.11) with normalised errors (see equation 3.10). The weighting can be based on the errors computed in the current frame, the previous frame or, as investigated in the next chapter, the history of the LP's performance. In chapter 4, the weightings are computed in such a way as to control the contribution of predictors dependent on its usefulness given the current appearance of the target. Equation 3.10 shows how a weight is computed and equation 3.11 illustrates the linearly weighted LP flock.

$$w^l = 1 - \frac{\|\bar{\delta\mathbf{x}} - \delta\mathbf{x}^l\|}{\max \|\bar{\delta\mathbf{x}} - \delta\mathbf{x}^l\|} \tag{3.10}$$

$$\bar{\delta\mathbf{x}} = \frac{\sum_{l=1}^{L} (w^l \cdot \delta\mathbf{x}^l)}{\sum_{l=1}^{L} w^l} \tag{3.11}$$

The experiments in section 3.2.3 show how this weighting strategy improves the accuracy of the LP flock.

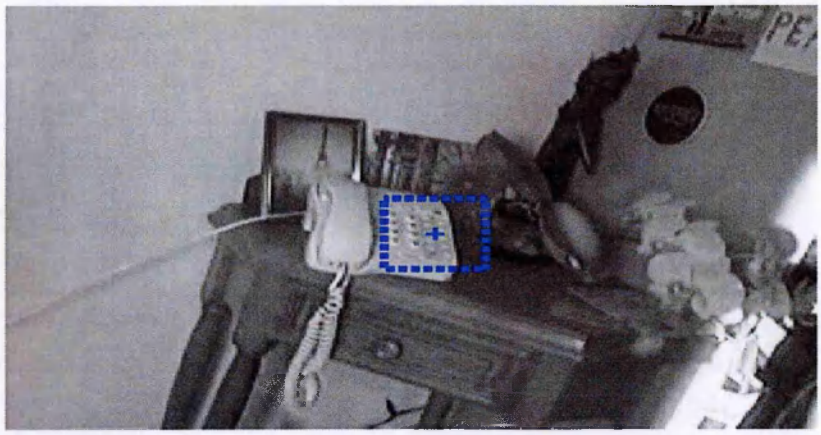
The low computational complexity of the LP learning methods results in lower accuracy of LPs than highly optimised offline learning procedures. This motivates the need for mechanisms to manage the flocks of LP's, such as this linear weighting mechanism. These weighting mechanisms are developed further in chapter 4.

3.1.5 Inter-frame motion example

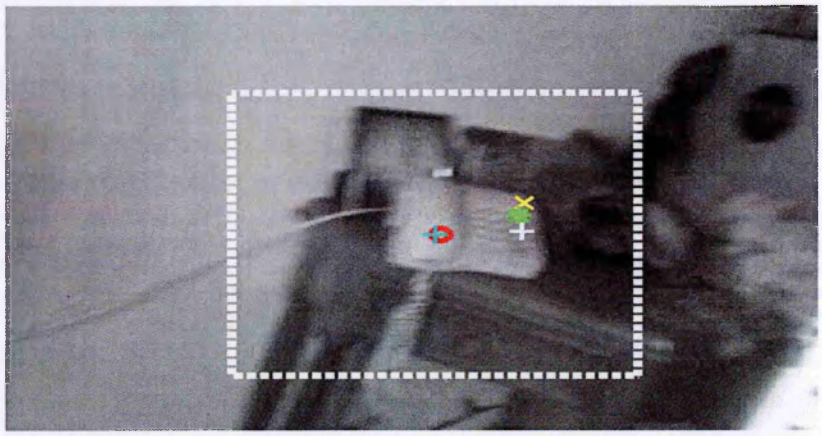
Each of the three trackers under investigation (Lucas-Kanade, LP and LP flock) was applied to an image sequence, captured from a moving web camera, containing considerable motion blur and large inter-frame displacements caused by vigorous shaking of the camera. Figures 3.5(a) and 3.5(b) show frames 374 and 375 respectively.

On figure 3.5(a) the reference point being tracked is indicated by the cross. On figure 3.5(b), which shows frame 375 as suffering considerable motion blur, the same coordinate is marked in light blue (grey in black and white) cross. Also marked on figure 3.5(b) is the position each of the trackers believes to be the target. The Lucas-Kanade tracker (red circle) has moved a short distance from the position in the previous frame and has failed to track the target. The single LP tracker (yellow X) has done better and the LP flock (green star) has done better still. The 'true' point (white cross) is obtained by taking a template of the target and finding the global minimum in the cost surface as shown in figure 3.6.

Figure 3.6 is informative as it illustrates the difference between the regression and registration processes, specifically highlighting the problems of using gradient descent or Newton methods that assume the presence of a locally convex similarity function with a minima at the true warp position. Although the global minimum of the similarity function, or cost surface, is at the true warp position, the Lucas-Kanade tracker is 'caught' in a local minimum. The inter-frame displacement was larger than the basin of convergence of the tracker *i.e.* it fell outside the area of convexity of the surface around the true point. On the other hand, both the regression techniques are able to 'leap' across the cost surface and track successfully despite motion blur and the large 37 pixel inter-frame displacement. This is because the regression approach learns how patterns of image intensity differences relate to displacements. In chapter 4 various trackers, including more advanced registration based approaches, are tested on the entire 1000 frame video sequence and, due mainly to severe camera shake and hence large inter-frame displacement, only the regression methods are successful.



(a) Frame 374 from video with camera motion. Location of reference point of feature being tracked is marked with a cross. The template size (40 by 40 pixels) is also marked with a rectangle.



(b) Frame 375. Location of reference point in last frame marked with a light blue (grey in black and white) cross. LK point of convergence marked with a red (dark grey) circle, single LP with a yellow (white) X, LP flock with a green (black) star and the true position marked with a white cross. The search area used to produce the cost surface below is marked with a white rectangle.

Figure 3.5: *Inter-frame motion example part I*: The registration (circle), regression (X) and flock (star) displacement estimators are tested on a image sequence featuring vigorous camera shake. The regression methods are shown to accurately estimate the large (37 pixel) inter-frame displacement while the registration method fails due to a local minimum in the cost surface (shown in figure 3.6).

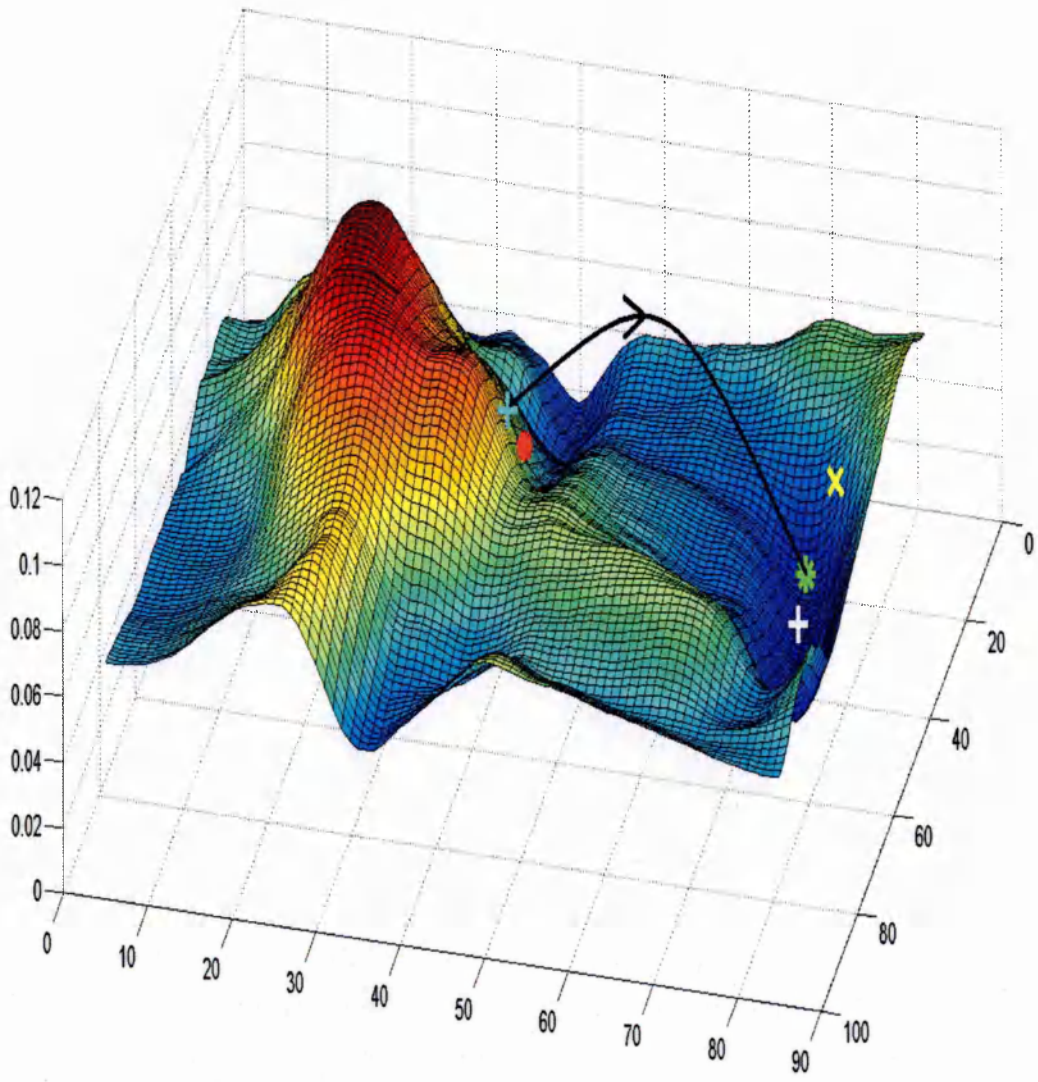


Figure 3.6: *Inter-frame motion example part II*: Cost surface of L_2 norm distance between template drawn from frame 374 and an 80 by 80 pixel region of frame 375 around reference position in frame 374 (shown in figure 3.5). Light blue (grey in black and white) cross indicates position in frame 374, red (black) circle is the location the Lucas-Kanade algorithm converges to, yellow (grey) X is the single LP result, green (grey) star is the LP flock result and the white cross is the global minimum of the cost surface that corresponds to the true position in frame 375.

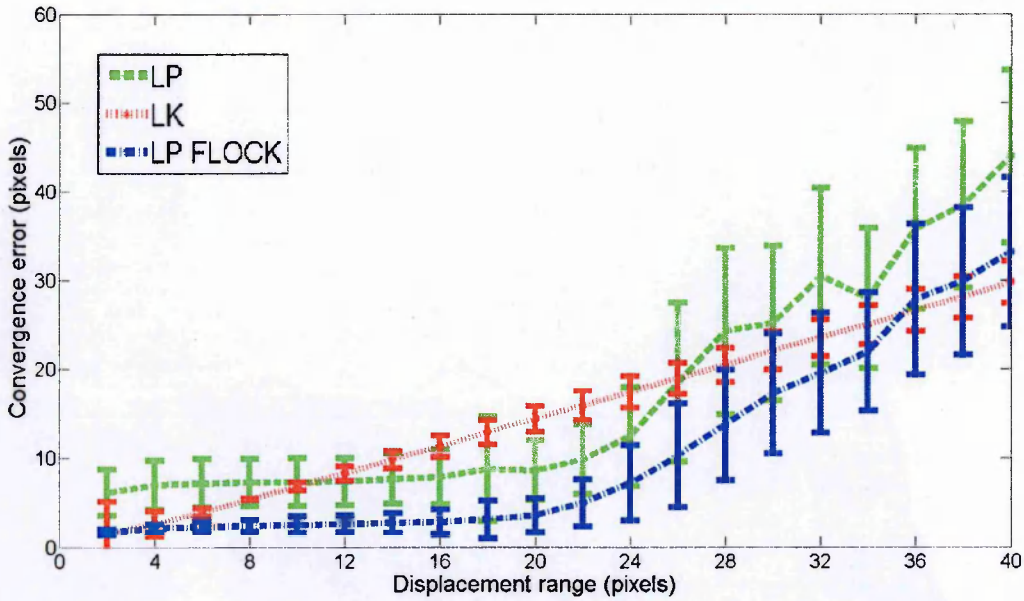


Figure 3.7: *Convergence error*: Convergence error (in pixels) for three tracking approaches over a range of test displacements. The error bars represent the log of the variance of the pixel error over the 3000 tests at each range.

3.2 Experimental Results

This section details a set of experiments used to characterise, compare and evaluate the various displacement estimation approaches. First a convergence test is introduced and used to characterise and compare registration and regression approaches for displacement estimation as well as to investigate the effects of some of the parameters for these methods. This is followed by an experiment illustrating the benefits of the flock weighting strategy.

3.2.1 Convergence testing

A convergence test is used to test and compare various configurations of the regression and registration tracking approaches. For registration, the test involves extracting a template at a given point, $x_{true} = \{xpos, ypos\}$, then starting the registration process at various displacements $x_{true} + d1, x_{true} + d2, ..., x_{true} + dn$, where $d = \Delta x$ and n is the

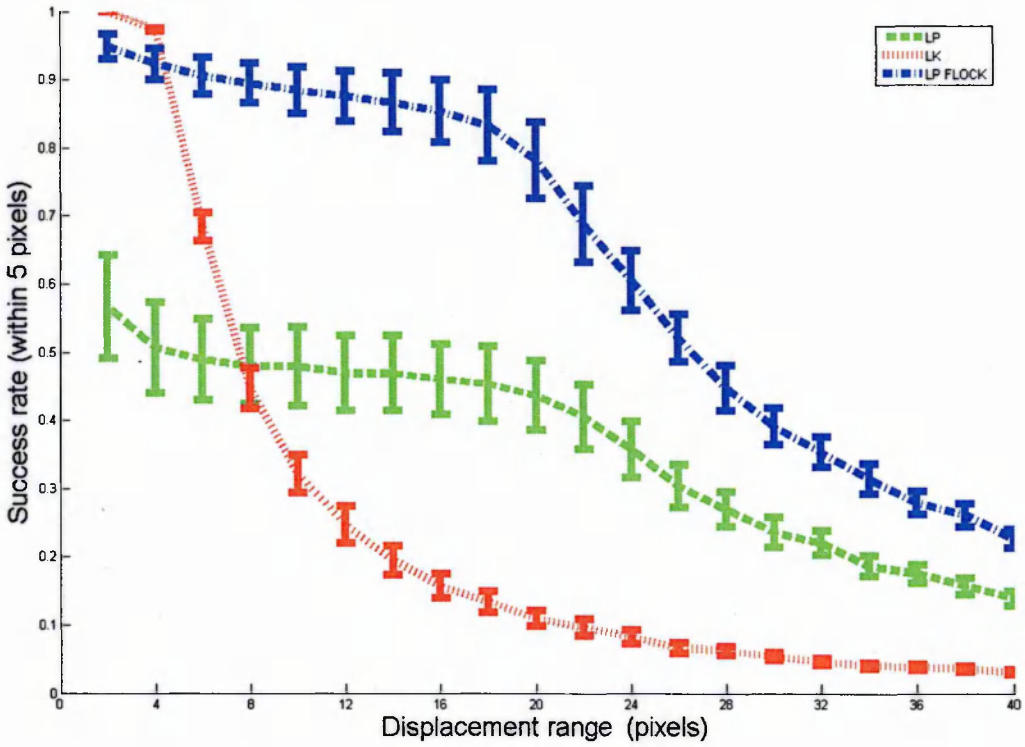


Figure 3.8: *Convergence success rate*: Success rate of the Lucas-Kanade, LP and LP flock tracker. A test is treated as successful if the tracker converges to within 5 pixels of the true point. The error bars represent the the variance of success score over the 3000 tests at each range.

number of tests carried out. The displacements can be thought of as simulated inter-frame displacements in the tracking scenario. For the regression tracking approach the test is similar - the model is learnt at P_{true} and predictions are made given observations at displacements. The convergence test evaluates the accuracy (how close to x_{true} does the tracker get), success rate (how many tests fall within a given accuracy) and range (maximum magnitude of displacements for which tracker performs well) of the approaches.

The results represented in figures 3.7 and 3.8 are obtained by performing convergence tests using three tracking algorithms (a single LP, a flock of 60 LPs and the Lucas-Kanade registration algorithm) on a dataset of three hundred image patches (fifteen points selected on a grid from twenty images of different content, qualities and from

different sources). The displacements (the horizontal axis) range from zero to forty with twenty equal steps. At each of the three hundred points, and for each of the twenty range steps, the convergence test is performed ten times giving a total of sixty thousand tests.

For the results presented in figures 3.7 and 3.8 the LP parameters are: $k = 100$ (number of support pixels), $N = 150$ (number of training examples), $r_{sp} = 20$ (support pixel range) and $r_{tr} = 20$ (training range). The LP flock is made up of 60 unweighted LPs with the same parameters. The Lucas-Kanade tracker uses the L_2 norm distance metric with a template of 20-by-20 pixels, zero order nearest neighbour interpolation and employs the Levenberg-Marquardt optimisation method.

It can be seen in figure 3.7 that, up to a certain range of displacements - that used in training the LP - the accuracy of both the regression methods remains fairly constant after which it degrades rapidly and linearly. The accuracy gained by the LP flock of sixty LPs is around four pixels and can be seen in figure 3.8 to increase the success rate by ten percent. The success rate is the proportion of tests at a given range that converge to within five pixels of the target. It is shown by the error bars in 3.7 and 3.8 that, along with accuracy, the stability of the predictions made by the LP flock is increased over the single linear prediction.

Figure 3.8 shows that the registration method has a greater success rate up to displacements of around five pixels, after which it degrades rapidly. This suggests the registration method has greater alignment accuracy within a certain range, the range of the basin of convergence of the alignment cost surface, than the LP flock regression approach.

It is evident in figures 3.7 and 3.8 that the regression approaches have a greater range than the registration approach. There are methods for increasing the range of registration approaches such as image blurring and multiscale image registration [35, 61]. These methods essentially work by smoothing the registration cost surface thus increasing the range over which alignment can be achieved but at the cost of alignment accuracy. Performing these operations hierarchically, from coarse to fine, can achieve greater range and increased accuracy but with an obvious increase in computational

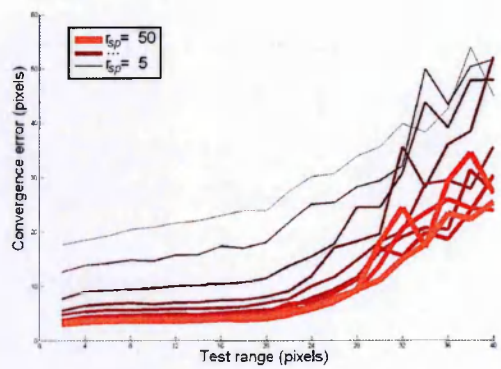
cost. An equivalent course to fine approach has been developed for regression methods by Zimmermann et al. [83] and also Ong and Bowden [59]. The Sequential Linear Predictor (SLP) first predicts displacement using a linear regression function trained on a larger range of displacements (and hence with lower accuracy) and then with another function trained on a smaller range and so on until the required level of accuracy is obtained. The real advantage of regression techniques over registration techniques is that the range is defined by the training process as opposed to being dependent purely on the shape of the alignment cost surface *i.e.* it is possible to specify a priori the desired operating range as is explored in the following section.

3.2.2 Parameter effects

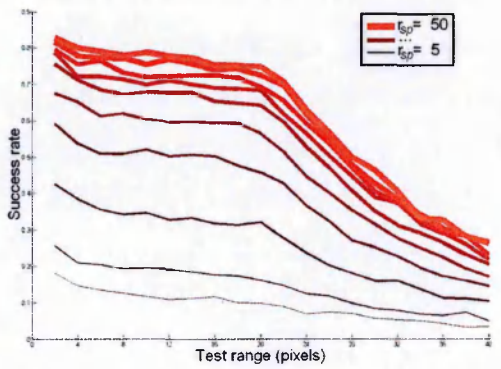
In order to evaluate the effect of various parameters on the accuracy, stability and computational cost of LP trackers, convergence tests are performed with a range of parameter configurations. The parameters explored are r_{sp} (range from reference point within which support pixels are selected), r_{tr} (range of synthetic displacements used in training), k (complexity of LP *i.e.* number of support pixels) and N (learning cost *i.e.* number of synthetic displacements used in training the LP). Rather than performing a global optimisation of these parameters (over the image dataset) these tests illustrate how the convergence characteristics of the trackers changes with varying parameters.

Figure 3.9(a) and 3.9(b) show how varying r_{sp} (the range from the reference point within which support pixels are selected) effects the LP's convergence test performance. As the support range increases, the accuracy increases. There is little or no effect on the range of displacements for which the prediction accuracy remains constant (the same as r_{tr}). Given the nature of the convergence tests (the image is static so there is no discrepancy between foreground and background) it should be noted that, in a real tracking scenario, if r_{sp} is too large it may result in the use of background pixels which would result in poor displacement predictions.

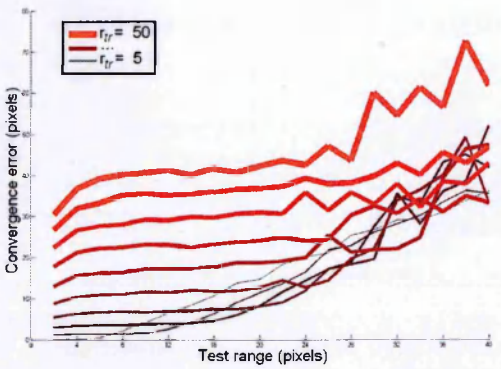
Figures 3.9(c) and 3.9(d) show how varying r_{tr} (range of synthetic displacements used in training) effects the convergence test performance. As the training range increases, the range of displacements for which the prediction accuracy remains constant also



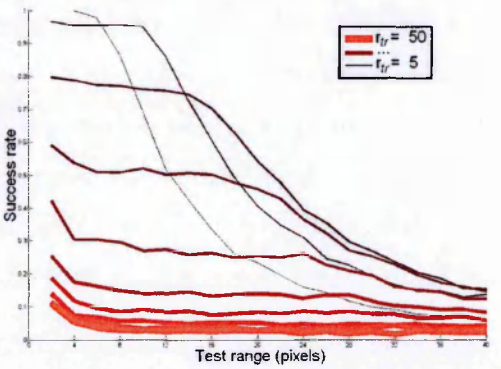
(a) Convergence error (vertical axis) with varying test ranges (horizontal axis) with different support pixel ranges, r_{sp}.



(b) Success (converges to within 5 pixels) rate (vertical axis) with varying test ranges (horizontal axis) with different support pixel range, r_{sp}.



(c) Convergence error (vertical axis) with varying test ranges (horizontal axis) with different training range, r_{tr}.



(d) Success (converges to within 5 pixels) rate (vertical axis) with varying test ranges (horizontal axis) with different training range, r_{tr}.

Figure 3.9: *Parameter effects*: The effect of varying the support pixel range, r_{sp}, and training range, r_{tr}, on pixel errors and success rates. Larger (redder in colour) lines indicate greater r_{sp}/r_{tr}. Values for r_{sp} and r_{tr} start at 5 pixels, increasing to 50 pixels in steps of 5.

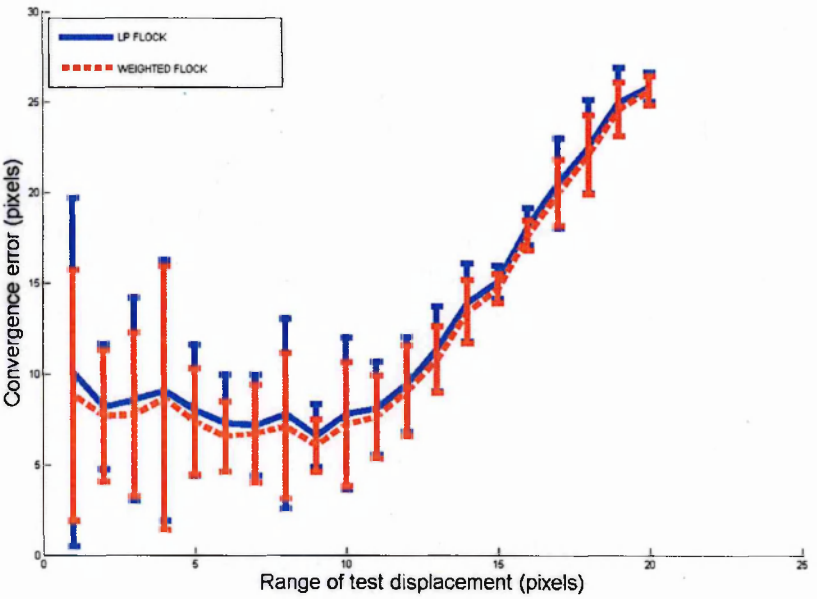


Figure 3.10: *Flock weighting results*: Average result of sixty thousand convergence tests on weighted flocks of LPs using images with synthetic occlusions. The weighted flock is more accurate than the unweighted flock across all displacement ranges. The stability of the flock also improves slightly as can be seen by the shorter error bars.

increases. This is as expected - a LP trained for displacements of up to 20 pixels will perform consistently for test ranges of 20 pixels or less and poorly for displacements greater than 20. The trade-off for this increase in operating range is lower prediction accuracy (this result is also illustrated in figure 3.3).

3.2.3 Prediction evaluation and flock weighting

The LP flock weighting strategy introduced in section 3.1.4 provides a mechanism for controlling each individual LP's contribution to the overall flock output. The level of contribution a LP makes can be influenced by two factors. Firstly, due to the random selection of reference point and support pixels and the inherent weakness of the LSQ method, a LP may be consistently poor at predicting displacements. This would imply the LP should make a small contribution and that it should have a low weight. Secondly a LP's ability to predict may also be affected by changes to pixel

intensities on the target. Such intensity changes may be brought about by changes to the appearance of the target, occlusions and, in the case of 2D translation LPs, out of plane displacements, rotation or affine deformation. If, for example, part of the target being tracked by a LP flock should become occluded, then those LPs whose reference point and support pixels are occluded - or indeed close to the occlusion boundary - will be considerably less reliable during the occlusion and hence would benefit from receiving a low weight.

In chapter 4, appearance models that can handle pixel intensity changes on the target, such as those brought about by occlusion, will be developed and section 4.3 extends the weighting mechanisms presented here in order to control the contribution of LPs given the current estimated appearance of the target. To demonstrate the principle and effectiveness of the weighting mechanism presented, an experiment comparing both the weighted and unweighted LP flock is presented. Figure 3.10 shows the results of running the two methods on a convergence test involving deformation to the target. Sixty thousand randomly selected displacements from the 300 image patches are made and the prediction/convergence error is recorded. However, after the LP flock is learnt and for each of the 20 test ranges, a 5-by-5 pixel area randomly located within 20 pixels of the target reference point is masked *i.e.* the 25 pixels are set to white thus synthesising an occlusion on the target.

The flock agreement error, $\|\bar{\delta\mathbf{x}} - \mathbf{x}\|$, computed on the current flock output is used to re-weight contributions and hence compute the ultimate flock displacement prediction. As can be seen in figure 3.10, the weighted flock is consistently more accurate than the unweighted flock. Although there is only marginal improvement of the error, the variance of the errors is significantly lower, as can be seen by the shorter error bars in figure 3.10.

It should be noted that the results shown in figure 3.10 only demonstrate that the flock agreement error can be used to weight poorly performing or occluded LPs. The usefulness of the weighting approach is more thoroughly demonstrated in chapter 4 where the weighting is used to evaluate, remove and replace LPs as well as form aspect specific sets of LPs.

3.3 Evaluation and Conclusion

The Linear Predictor tracker has been introduced and investigated to characterise it's prediction performance and how this performance is affected by various parameter values (e.g. r_{tr}) and conditions (e.g. test displacements ranges). An approach to combining constellations of LPs into a LP flock is introduced and the improvements in accuracy and stability have been quantified. Mechanisms for evaluating LP performance and controlling the contribution each LP makes to the overall flock output have been tested and have shown improvements over regular unweighted flocks.

It is the LP's that perform the task accomplished by an optimisation procedure in registration techniques. Both registration and regression methods base their output on the same information, image intensity differences. The computational efficiency of the regression approach is a result of learning a simple and general mapping directly from patterns of image intensity differences to desired displacements, and applying this mapping at each displacement prediction step, rather than performing an optimisation process for each prediction step.

The inter-frame motion example in section 3.1.5 illustrates the power of regression based tracking approaches. The ability to accurately predict large inter-frame displacements regardless of the presence of local minima in the alignment cost surface, with relatively low computational cost, provides scope for developing a feature tracker capable of tracking very fast moving targets. The trade-off between prediction accuracy and stability on one hand and maximum inter-frame displacement (controlled by r_{tr}) has been identified and, along with the increase in accuracy and stability gained by flocking LPs, provides a framework for defining a tracker in terms of required accuracy and range.

The LP allows for the explicit tradeoff between predictor range, accuracy and computational cost as shown by the convergence tests in section 3.2.1. To increase the flexibility and accuracy of LPs, mechanisms to evaluate, weight, remove and relearn LPs online during tracking are required. A general tracking framework that combines online-LP flocks with methods for online appearance model learning is developed in the next chapter.

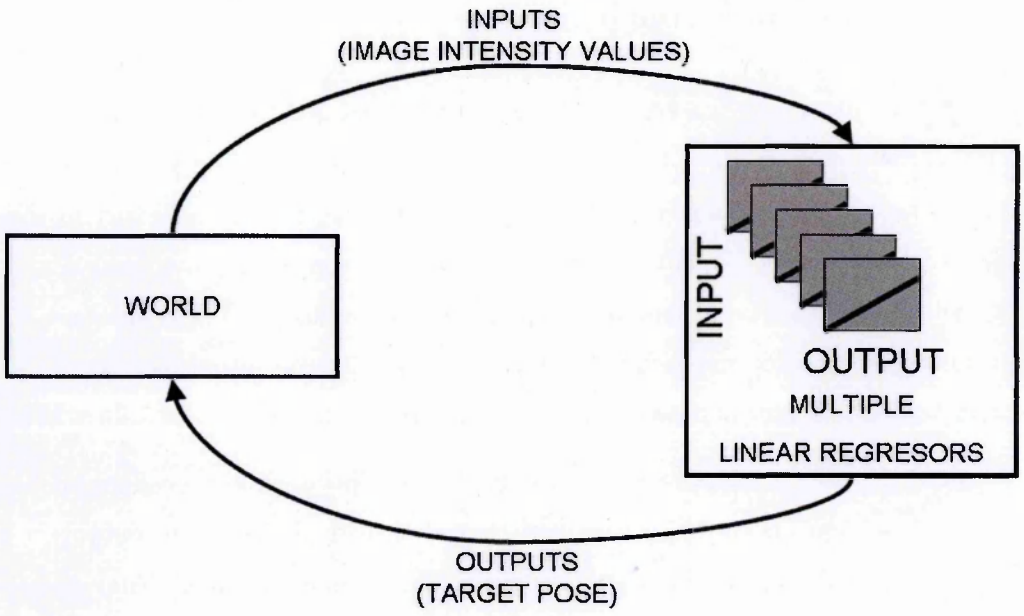


Figure 3.11: *Multi regressor approach to input-output mapping*: This figure represents the LP flock approach proposed in this chapter within the PAC framework introduced in chapter 1.

As stated in chapter 1 this thesis is concerned with learning mappings that map directly from input space to output space without referring to intermediate, high-level, symbolic or top-down representations imposed by an engineer. This chapter has developed such a mechanism and applied it to the problem of feature tracking. The use of multiple regression functions (LP flocks) has shown an improvement over single regression function mappings. This multi regressor approach is illustrated in figure 3.11. In the following chapters this framework is extended further by employing unsupervised learning approaches to the input and output spaces.

Chapter 4

Appearance Models for Tracking

This chapter introduces methods for unsupervised learning of models of the appearance of a target object during tracking. A general Simultaneous Modeling and Tracking approach is introduced within which the tracking process provides a mechanism for self supervision of the appearance model learning process and, in return, the appearance model provides information about the structure of the target appearance space that enables the tracker to cope with a high degree of variation in appearance.

Whilst prior models can be used to model target appearance, they place restrictions on the scope of applications for which the trackers can be easily used. Furthermore, visual tracking approaches that are able to adapt their representation of the target on-the-fly show increased robustness over approaches for which the representation is either specified (hard coded) or learned from a training set. Single template models, such as those employed in the Lucas-Kanade algorithm [51], aim to model the target appearance as one point on the appearance-space manifold. In order to increase robustness to appearance changes and minimise alignment drift, various template update strategies have been developed. These include naive update [56] where the template is updated after every frame and strategic update [56] where the the first template from the first frame is retained and used to correct location errors made by the updated template. If the size of the correction is too large, the strategic algorithm acts conservatively by not updating the template from the current frame. With template update methods, the template is intended to represent the current single point in the appearance-space

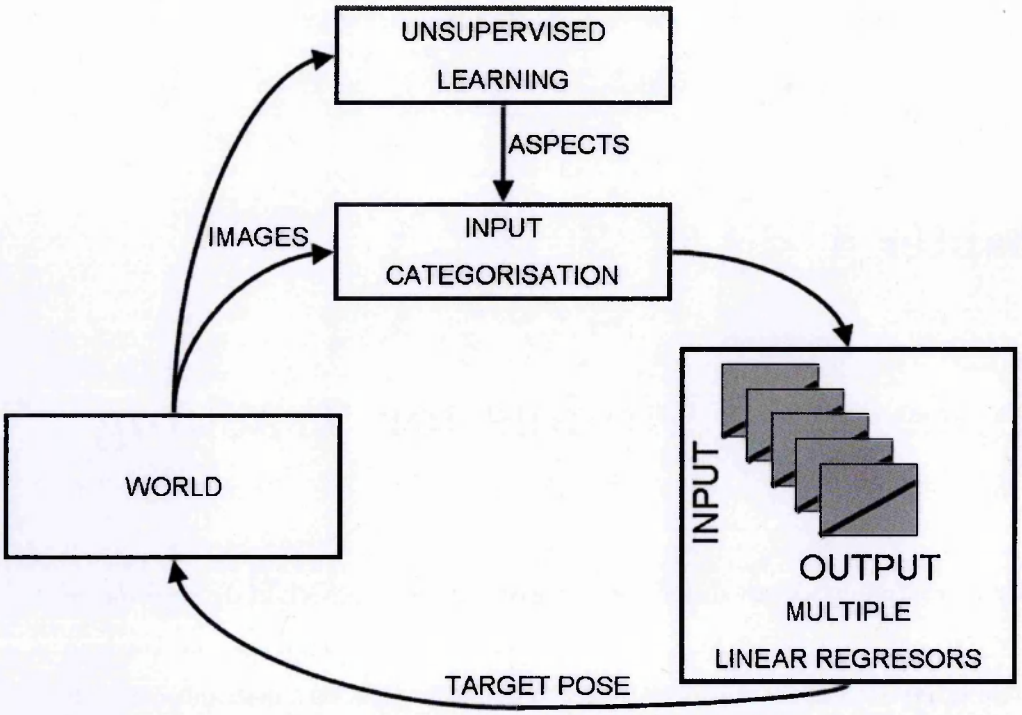


Figure 4.1: *Unsupervised learning in problem input space for input-output mappings*: This figure represents the approach proposed in this chapter within the PAC framework introduced in chapter 1. Unsupervised learning is applied to input images to discover aspects of the target that can be used to classify new inputs.

manifold. Approaches that use some or all templates [20, 23], drawn from all frames, represent a larger part of this manifold. In this work, all stored templates are incrementally clustered to discover modes or *aspects* of the target appearance.

Tracking methods that adapt the representation of the target during tracking are prone to drift, as the appearance model may adapt to the background or occluding objects. The approaches proposed here address this problem by maintaining modes of an appearance model that correspond to past appearances. Whilst this approach reduces the impact of drift, as erroneous appearance samples do not contaminate all modes of the appearance model, the method does not address the drift verses adaptation trade-off directly. The work of Kalal et al. [45] explicitly addresses the trade-off between adaptation and drift.

This chapter builds on the multiple regressor approach to input-output mapping introduced in the previous chapter. By applying efficient unsupervised learning methods to the input images, *aspects* of the target are identified. This approach is represented, in terms of the PAC framework introduced in chapter 1, in figure 4.1. The LP weighting strategies developed in the previous chapter are also extended to allow adaptable aspect specific weightings, producing aspect specific LP flocks. Identifying which aspect a new input image belongs to enables the LP weighting to be adjusted to suit the current target appearance.

Regression tracking techniques tend to require offline learning to learn suitable regression functions. The computationally efficient LP learning strategies introduced in the previous chapter mean that it is feasible to learn LPs online, during tracking. In comparison with approaches that use highly optimised offline learning procedures, the LPs introduced in the previous chapter have limited accuracy. Therefore, mechanisms to manage the flocks of LP's, such as the linear weighting mechanism introduced in the previous chapter, are required. The weighting mechanisms developed in this chapter facilitate the use of the *online-LP*, and so the approach benefits from the efficient learning procedures without loss of accuracy. Removing the need for offline learning greatly increases the applicability of the regression approach. The online-LP tracker can simply be seeded with an initial target location, akin to the ubiquitous Lucas-Kanade algorithm [51].

It is clear that reducing the level of supervision required to learn effective models is desirable from the point of view of the engineer (reducing the time spent hand labeling data). However, it is also true that semi-supervised or unsupervised methods can, in some cases, discover representations that are both more efficient and more grounded in the data than those obtained from highly supervised systems.

The methods developed in this and the previous chapter are designed to operate at high frame rates and as such need to be computationally efficient. The overarching design paradigm has been to use fast/simple methods: linear regression (for displacement prediction), random sampling (for learning displacement predictors and template extraction), incremental template clustering (for appearance modeling) and linear weighting

Table 4.1: Table of abbreviations used throughout text.

Abbreviations	Full meaning
LK	Lucas-Kanade: tracking by registration.
LP	Linear Predictor: tracking by linear regression.
SMAT	Simultaneous Modelling And Tracking: Adaptive multi-modal template based appearance model [20].
LK-SMAT	Tracking approach that combines LK displacement estimation with SMAT appearance modelling.
LP-SMAT	Tracking approach that combines LP displacement estimation with SMAT appearance modelling.
LP-MED	Tracking approach that combines LP displacement estimation with a medoidshift based appearance model.

(for associating displacement predictors with appearance modes). The use of simple regression methods is offset by an evaluation mechanism that allows both the weighting of the contribution of each displacement predictor and the continual disposal and replacement of poorly performing displacement predictors.

There are many different formulations of the tracking problem that lead to many and varied solutions: tracking by detection [78], tracking using graph cut algorithms to iteratively segment the target [11] and condensation algorithms [39] to name but a few. Each approach is thought to have a certain scope of applications for which it will work best. It has been established that a significant class of tracking problems can be solved using the Linear Predictor and this work aims to extend this class to problems requiring online feature tracking with appearance variation and real time operation. The experiments in section 4.4 go some way to delimiting the class of problems for which the proposed approach is suitable.

Table 4.1 lists the abbreviations used throughout this chapter. This includes the two displacement estimation methods used, LK (Lucas-Kanade: tracking by registration) and LP (Linear Predictor: tracking by linear regression) as well as the three configurations of the tracking framework developed: LK-SMAT (LK displacement estimation with the SMAT template based appearance model), LP-SMAT (LP displacement esti-

mation with the SMAT appearance model) and LP-MED (LP's with the medoidshift based appearance model).

The rest of this chapter is organised as follows: First, in section 4.1, an overview of the proposed tracking framework is presented. In section 4.2 the key motivation and fundamental approach to aspect learning for tracking is presented. In section 4.2.1 a variant of the SMAT model, first introduced by Dowson & Bowden [20], is described. The SMAT model provides a comparison with the medoidshift template clustering approach to appearance model learning that is introduced in section 4.2.2. Section 4.2.3 contains a discussion on appearance modeling methods. Section 4.3 details three configurations of the general tracking framework, namely: LK-SMAT, LP-SMAT and LP-MED. The LK-SMAT model, detailed in section 4.3.1, is essentially the SMAT model presented by Dowson & Bowden [20] cast in the tracking framework presented here. In section 4.3.2 the LP-SMAT tracker, that utilises the SMAT model to partition a bank of online-LP displacement estimators, is introduced and in section 4.3.3 the LP-MED tracker is introduced. The LP-MED model maintains a constantly updated bank of online-LP displacement estimators and adaptively associates these to aspects of the target appearance discovered by medoidshift template clustering. Section 4.4 presents an evaluation of each of the presented tracking approaches, and provides comparison with other state of the art approaches based on both computational efficiency, tracking accuracy and robustness. Section 4.5 contains an evaluation and conclusions regarding online appearance modeling and the proposed tracking approaches.

4.1 Simultaneous Modeling and Tracking Approach Overview

At the most general level the proposed tracking approach can be described by the following process:

1. **Estimate** the current target appearance using an appearance model
2. **Adapt** the displacement estimation mechanism to suit current estimate of appearance

3. **Estimate** inter-frame displacement of the target
4. **Adapt** the appearance model given new appearance data
5. **Repeat** steps 1-4

These stages are achieved by the interaction of two components, namely the **displacement estimator** and the **appearance estimator**. The displacement estimator, as well as generating the tracking output, provides a mechanism for supervision of the appearance model learning process *i.e.* it provides new examples of the target appearance that are added to the appearance model. In return, the appearance estimator provides information about the structure of the target appearance space that enables the tracker to cope with a high degree of variation in appearance. This basic methodology is represented in figure 4.2.

The target appearance samples - templates drawn from the image at the targets estimated location - will change during tracking. This is caused by all appearance variations that are not modeled by the pose parameters *e.g.* rotation (if translation transformations only are considered), lighting change, occlusion or changes of expression (when tracking faces) as well as frame-to-frame inaccuracies in displacement estimation. In the proposed appearance modelling approach, all stored templates are incrementally clustered to discover modes or *aspects* of the target appearance. Identifying the current aspect of the target appearance is the role of the appearance model, as shown in figure 4.3. By identifying aspects of the target, it becomes possible to adapt the displacement estimation mechanisms to suit the current appearance.

The proposed tracking framework associates these aspects to banks of displacement estimators - trackers - via an *association matrix*, see figure 4.3. The values in the association matrix reflect the suitability of each tracker to each aspect of the target. This provides a flexible way of controlling the influence of each tracker to the overall pose estimation.

Within this architecture there are many possible approaches to implementing the appearance model, association strategy, displacement estimation and final pose estimation processes. In section 4.2 two methods for on-the-fly appearance modelling are

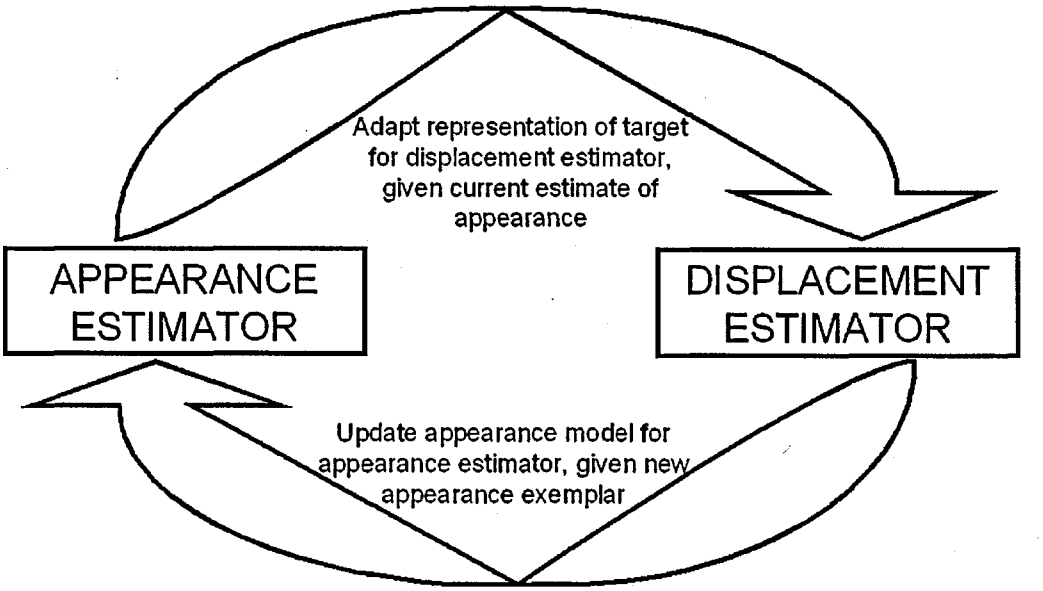


Figure 4.2: *Simultanious Modeling and Tracking methodology*: The displacement estimator, as well as generating the tracking output, provides a mechanism for supervision of the appearance model learning process i.e. it provides new examples of the target appearance that are added to the appearance model. In return, the appearance estimator provides information about the structure of the target appearance space that enables the tracker to cope with a high degree of variation in appearance.

introduced. Two displacement estimation methods - template registration and linear regression - were investigated in chapter 3. In section 4.3 various configurations of the complete tracking framework are detailed.

4.2 Aspect learning for tracking

Aside from the intrinsic requirement of a representation of the target appearance for all tracking methods, appearance models can additionally help cope with appearance changes not parameterised by the pose parameters. Provided a perfect geometric model of the target *and* environment was available, it would be possible to parameterise every possible change to the target appearance. Such a model would have to include parameterisations of not only all degrees of freedom (DOF) of the target object but

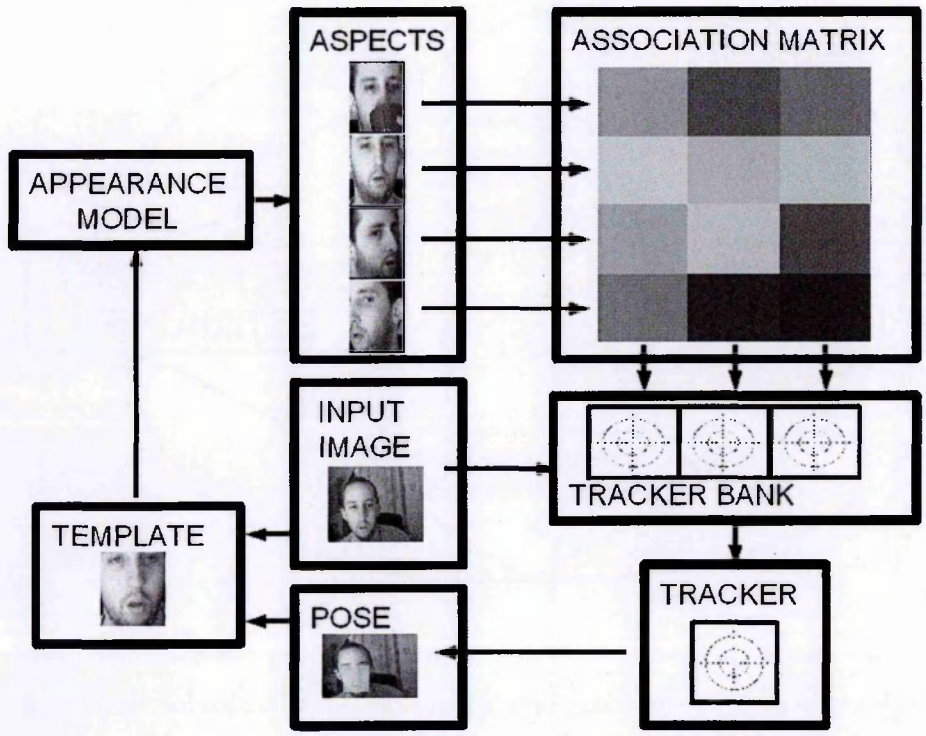


Figure 4.3: *Generic system architecture*: The appearance model stores all target templates and identifies aspects of the target. Aspects are associated to feature trackers by an association matrix. Each feature tracker contributes to the overall pose estimation, the level of contribution is determined by the strength of association to the current aspect i.e. the association matrix value.

also other objects in the environment that may occlude the target along with environmental effects such as changes in lighting. This is simply not feasible in any real scenario. In addition, the estimation of the huge number of parameters required by such a model would be intractable. Tracking approaches, therefore, tend to model only a subset of pose parameters, commonly translation (2 DOF) or affine (6 DOF). Any changes not represented by the selected pose parameters will often cause tracking failure. An appearance model can provide a means of compensating for this partial parameterisation.

The appearance modeling methods developed here model the variations in appearance not accounted for by translation pose transformations. First the displacement estimator

identifies the location of the target in the 2D image space then an *appearance sample* - image template at target location - is extracted. If the target has, for example, moved closer to the camera thus causing a scale increase in image space, the new appearance sample is likely to capture the appearance of only a sub-region of the target at the new scale (as the image template size will not change). Likewise, if the target moves away from the camera, causing a target scale decrease, the appearance sample template is likely to include some background around the target as the target is reduced in size and the template size remains constant. Another example of an appearance change not accounted for by a 2 DOF parameterisation is brought about by partial occlusion of the target. If another object moves between the camera and the target, occluding part of the target, the appearance sample template will feature this occluding object. All these (and more) appearance variations are considered part of the appearance space. The approaches introduced below model the appearance space manifold, created from the appearance sample templates extracted at each frame of tracking, in order to facilitate the efficient tracking of image patches undergoing arbitrary deformations.

Trackers with no prior model of the target appearance ([51, 41, 20, 31], that are intended to track a 3D object such as, for example, a cube, are initialised by identifying the region in the first frame that contains the cube. If the cube then starts to rotate, perhaps exposing a new face of the cube and hence presenting a new *aspect* of the target, the initial target representation may no longer be adequate. It would therefore be advantageous to identify that a new aspect of the target had been presented and to adapt the target representation used for tracking accordingly. Eventually the cube may rotate back to its original orientation and thus present the initial aspect of the target again. In this case it would be advantageous to recall the representation associated to that aspect. This is the function of the appearance models developed here: to identify different aspects of the target - clusters of appearance samples - such that the target representation used in estimating inter-frame displacement can be partitioned and associated with the aspects for which they perform well.

The term 'aspect' is used to describe some mode or cluster of the appearance manifold. As discussed above, the appearance manifold may include regions associated with all appearance variations not modeled by the pose parameters *e.g.* rotation, lighting

change, occlusion or changes of object appearance itself.

If a single template appearance of an object is considered as one point on the appearance-space manifold (as in the Lucas-Kanade method), the manifold can be represented by storing the set T of all templates, $T = \{\mathbf{G}^0 ... \mathbf{G}^t\}$ drawn from all frames $\{\mathbf{F}^0 ... \mathbf{F}^t\}$. In order to identify aspects of the target, the set of templates, T, should be clustered or partitioned, $T = \{T^0 ... T^M\}$ where $T^m \subset T$.

For a subset of templates, $T^m \subset T$, to represent a real aspect of the target appearance, the templates that make up an aspect should be similar to one another and different to the templates in all other aspects. Similarity is determined by a distance metric. The L_2 norm distance is used in the methods below due to its computational efficiency but others, such as Mutual Information or Normalised Correlation could also be used. Both of the clustering methods detailed below compute and maintain a matrix of L_2 norm distances between templates and use this to determine each templates aspect membership *i.e.* to which aspect that template belongs.

4.2.1 SMAT: Greedy template clustering

The SMAT model presented here is a variant of the SMAT model presented by Dowson & Bowden [20] cast in the tracking framework presented in this thesis. In order to identify different aspects of the target, modes or clusters of the appearance manifold must be discovered. The method presented here components the appearance manifold, assigning templates to components with a greedy incremental algorithm.

Each of the M aspects, $T^m \subset T$, $m = 1...M$ of the appearance manifold are represented by: a group of templates, the median template μ^m, a threshold τ^m, and a weighting w^m. Use of the median rather than the mean avoids pixel blurring caused by the averaging of multiple intensity values of templates that are not perfectly aligned. Weight w^m represents the estimated a priori likelihood that the m^{th} component best resembles the current appearance of the target. During tracking, a template is drawn from the new frame at the location determined by the displacement estimator. To identify the best matching component to the new template, a greedy search is performed starting with the component with the highest weight and terminating when a component, T^{m^*}, is

Figure 4.4: *Appearance model medians for head tracking sequence*: Two examples of the median templates of the four components of the appearance space are shown, ordered with decreasing weight from left to right. It is clear that the modes identify aspects of the target such as side/front/occluded views. The matched component for the current frame is marked with the bullseye.

found whose L_2 norm distance to the image patch is less than the threshold τ. The input image patch is then added to component T^{m^*} and the median, μ^{m^*}, threshold, τ^{m^*}, and weights, $w^m, m = 1...M$, are updated. See Eq. 4.1 for the component weight update strategy. If no match is made, a new component is created with the new template and the template from the previous frame. The learning rate, α, sets the rate at which component rankings change and is set to α=0.2 for all experiments. This value was found through experimentation.

$$w^m = \begin{cases} \frac{w^m+\alpha}{1+\alpha} & \text{if } m = m^*; \\ \frac{w^m}{1+\alpha} & \text{if } m \neq m^*. \end{cases} \tag{4.1}$$

To facilitate the efficient update of an appearance model component, a matrix $\mathbf{Q}^m$ maintains the L_2 norm distances between each pair of templates in the m^{th} component. Adding a new template to the component then requires only the computation of a single row of $\mathbf{Q}^{m^*}$ *i.e.* the distances between the new template and all other templates. The median template index, j^*, is calculated using Eq. 4.2 and the component threshold

τ^{m^*} is computed using Eq. 4.3 which assumes an approximately Gaussian distribution of distances and sets the threshold to three standard deviations of the distribution.

$$j^* = \underset{j}{\operatorname{argmin}} \sum_{i=0}^{n} Q_{ij}^m, j = 1....n \tag{4.2}$$

$$\tau^{m^*} = 3\sqrt{\frac{1}{n}\sum_{i=0}^{n}(Q_{ij^*}^m)^2} \tag{4.3}$$

The dimensions of $\mathbf{Q}^m$ depend on the number, n, of templates in the model but can be limited to bound memory requirements and computational complexity. In practice, new templates replace the worst template from the component. It is also possible to limit the number of components, M. When creating a new component the new component replaces the worst existing component identified by the lowest weight $m_{worst} = \underset{m}{\operatorname{argmin}}\, w^m, \{m = 1...M\}$.

For all the experiments presented in section 4.4 a maximum of n=60 templates are maintained in each of a maximum of M=4 components of the model. This is found to be sufficient to model a stable distribution whilst preventing computational costs becoming too high for real-time tracking. Figure 4.4 illustrates the SMAT model being used to identify aspects of a head during a head tracking sequence. It can be seen that the modes identify aspects of the target such as side, front or occluded views.

4.2.2 Medoidshift template clustering

The second appearance model presented is again constructed online by incrementally clustering image patches to identify various modes of the target appearance manifold. Here, the clustering is performed by the medoidshift algorithm introduced by Sheikh et al. [66]. Medoidshift is a nonparametric clustering approach that performs mode-seeking by computing shifts toward areas of greater data density using local weighted medoids. As Sheikh et al. [66] show, the procedure can be performed incrementally, meaning the clustering can be updated at the inclusion of new data samples and the removal of some existing data samples.

Figure 4.5: *Appearance model clustering for head tracking sequence*: The distance matrix pre and post clustering is shown with three subsets of exemplars **A**, **B** and **C**. Sets **A** and **C** are temporally separated but have the same appearance. Templates from each subset are also shown.

During tracking the appearance templates are collected into vectors $\{\mathbf{G}^0 ... \mathbf{G}^t\}$ and, as for the greedy clustering approach, a distance matrix, $\mathbf{Q}$ is populated with the L_2 norm distances. Where the SMAT model maintains a $\mathbf{Q}$ matrix for each model component, this model maintains one $\mathbf{Q}$ matrix recording the distance between each stored frame. The medoidshift algorithm uses $\mathbf{Q}$ to obtain a clustering[1]. The clustering is incrementally updated given a new $\mathbf{G}$ vector and hence (by computing L_2 norm values) a new row/column of $\mathbf{Q}$. In order to constrain the memory requirements and computational complexity of maintaining the appearance model, the number of templates retained, and hence the number of data points clustered, is limited. Once the limit has been reached the oldest template is removed and replaced with the new template. The cluster update must accommodate both the introduction and removal of data points. The incremental update is achieved in a computationally efficient manner exactly as described in [66].

The effect of this clustering is illustrated in figure 4.5 that shows the distance matrix at frame 275 of a head tracking sequence before and after matrix indices are sorted according to the cluster label. As can be seen, two temporally separated subsets, A and C, of templates are assigned to the same cluster, $A \cup C \subset T$, identifying the *front view* aspect whilst a third subset, $B \subset T$, is partitioned and identifies a *side view* aspect of the face. It is obvious that a displacement estimator that represents the target appearance of the hidden side of the face will be less reliable while the viewable

[1] As no meaningful partitioning is possible with small sample sets, the clustering procedure is not carried out until frame 11 of tracking

side aspect is presented.

4.2.3 Appearance model discussion

While the greedy approach provides a computationally efficient method of partitioning the templates $T = \{\mathbf{G}^0...\mathbf{G}^t\}$ into aspects, $T = \{T^0...T^M\}$ where $T^m \subset T$, the algorithm lacks some flexibility. Rather than the number of aspects being a predefined value, M should ideally be data dependent and reflect (rather than determine) the number of modes present in the data's distribution. Also once a template is assigned to a certain component it will never become part of another component. This rigidity in terms of template-to-cluster assignment and fixed number of modes is likely to cause problems as the target appearance manifold evolves during tracking.

The data driven, mode seeking medoidshift incremental clustering algorithm offers greater flexibility to the appearance modelling process. The number of aspects, M, are not predefined and, as the appearance manifold grows and changes over time, so too can the aspect membership of each template.

Whilst the flexibility of the medoidshift approach allows a representation that is more reflective of the real underlying appearance distribution, the resulting representation of the aspects are less straightforward to interpret than the SMAT model. As the SMAT model has a fixed number of aspects, it is straightforward to construct an association matrix that associates a set of displacement predictors to each of the models aspects. With the medoidshift approach however, the varying number of aspects discovered and the adaptive cluster membership of templates necessitates a less straightforward association mechanism. Section 4.3 gives details of how both the appearance models are used within the tracking framework.

A significant factor in the computational overhead of these appearance models is the maintenance of the distance matrix, $\mathbf{Q}$. As stated, this can be controlled by limiting the number of templates stored by the model. Another way to control the computational cost is to reduce the dimensionality of the distance function *i.e.* to subsample the image templates prior to computation of L_2 norm distances.

4.3 Tracking Framework

This section details three configurations of the tracking framework: LK-SMAT, LP-SMAT and LP-MED. The first method uses the appearance model to identify different aspects of the target appearance and to provide a template - the median template of the best matching model component - for use in the registration process. For the LP methods, the function of the appearance models developed is to identify different aspects of the target such that the set of LPs can be partitioned and associated with the aspects for which they perform well.

Details of the mechanisms used to associate flocks of LPs with appearance modes identified by each of the appearance models are presented. Due to differences in the clustering approaches used - greedy and medoidshift clustering - different strategies for this partitioning and association are required. Specifically, with the medoidshift approach, there is not a fixed number of modes and an appearance template may change the cluster to which it belongs, whereas with the SMAT approach, there is a fixed number of modes and a template is assigned to just one mode for the duration of tracking.

4.3.1 LK-SMAT: Registration based Simultaneous Modeling and Tracking

This configuration of the tracking framework is a variant of the SMAT tracker presented by Dowson & Bowden [20].

The LK-SMAT tracker uses the SMAT appearance model to identify different aspects of the target appearance and thus provide a template - the median template of the best matching model component - for use in the registration process.

There is a one-to-one association between the target aspect and the templates used for tracking, this is illustrated by the identity association matrix in figure 4.6. Only one template, the median of the matched component, is associated to an aspect.

Tracking is the process of registering new image data with the median template from the estimated best aspect, extracting a template from the estimated location, updating

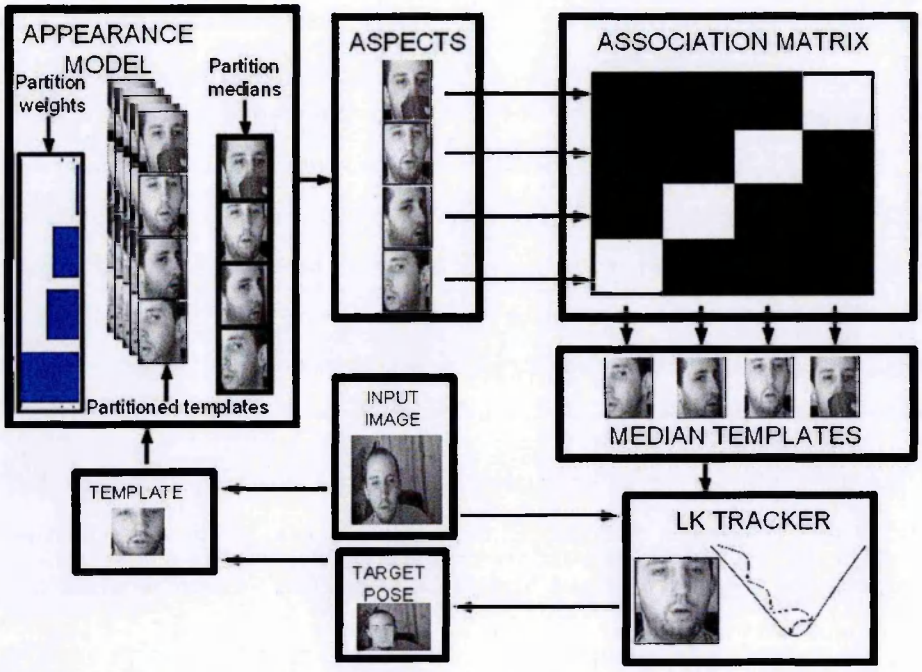


Figure 4.6: *LK-SMAT system architecture*: The SMAT appearance model identifies aspects by partitioning templates using a greedy clustering algorithm. Identifying the current aspect selects the template for use in registration process. The association matrix in this formulation is simply an identity matrix.

the appearance model (with the greedy algorithm), selecting the best component and hence medium template for registering with the next frame and so on.

The complete tracking procedure is detailed in Algorithm 1

4.3.2 LP-SMAT: Linear Predictors for Simultaneous Modeling and Tracking

The LP-SMAT tracker learns LPs specific to a particular aspect of the target object in order to continue to track through significant appearance changes. This association between aspects and LPs is achieved by an association matrix, $\mathbf{A}$, as illustrated in figure 4.7. Given a bank of L linear predictors and M appearance model components, the association matrix $\mathbf{A}$ has dimension $(L \times M)$. A zero value at $\mathbf{A}_{lm}$ indicates that predictor l is not associated to component m and therefore is deactivated when

Algorithm 1 LK-SMAT tracking procedure

$\mathbf{F}^0 \leftarrow$first image

Initialise target position $\bar{\mathbf{x}}^0$, height h and width w from user input

Extract first appearance template $\mathbf{G}^0$

Set initial component weight $w^m = 1$

$\mathbf{G}^* \leftarrow \mathbf{G}^0$

$t \leftarrow 0$

while $\mathbf{F}^t \neq NULL$ **do**

Register currently selected appearance template $\mathbf{G}^*$ with new frame $\mathbf{F}^t$ as in eq. 3.4

Extract new appearance template $\mathbf{G}^t$ at estimated target location

Assign new template to component m^* using greedy search algorithm

Compute L_2 norm distances for a single row of $\mathbf{Q}^{m^*}$

Compute median template index, j^*, and component threshold, τ^{m^*}, using Eq. 4.2 and Eq. 4.3

Update component weights, $w^m, m = 1...M$, as in Eq. 4.1.

$t \leftarrow t + 1$

end while

component m is active *i.e.* $m = m^*$. Each of the M components are associated to L/M LPs. For all the experiments, $M = 4$ and L=160 meaning 40 LPs are associated to each component and hence that 40 linear predictions are computed each frame.

An error function is used to continually evaluate each LP's performance over time. Rather than assigning a single error value to predictor l, error values are instead assigned to the association between each of the L predictors and each of the M appearance model components. The error values are stored in the association matrix $\mathbf{A}$ and can also be interpreted as a measure of the strength of association between a predictor and an appearance model component. The performance value used is a running average of prediction error with exponential forgetting; meaning that high values indicate poor performance. The error function used is the L_2 norm distance between predictor output $\delta\mathbf{x}_l$ and the overall tracker output $\bar{\delta\mathbf{x}}$, $\|\delta\mathbf{x}_l - \bar{\delta\mathbf{x}}\|$. Equation 4.4 details how the asso-

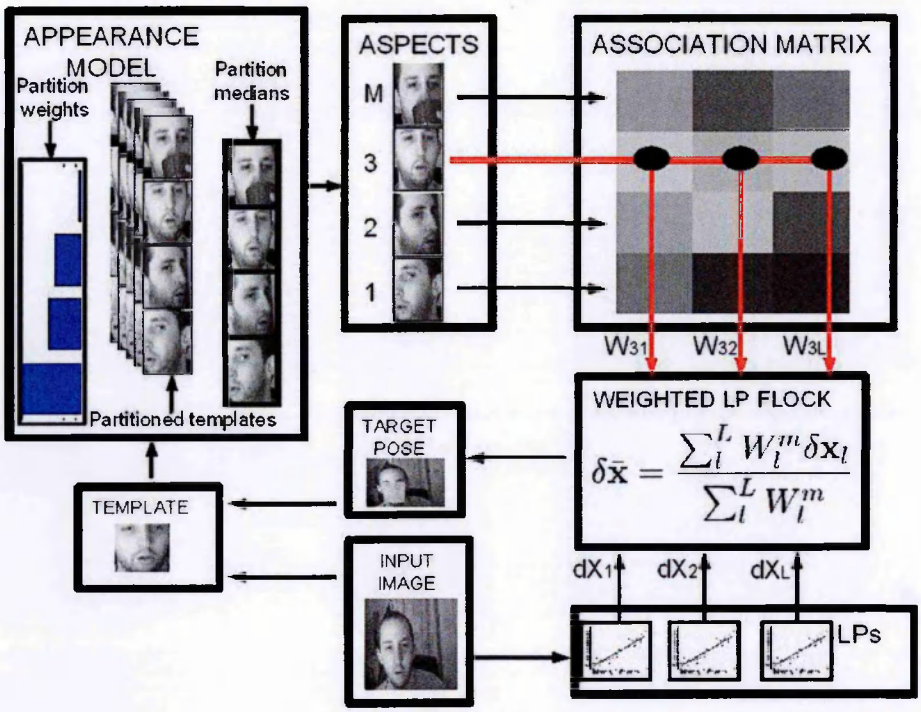


Figure 4.7: *LP-SMAT system architecture*: LPs associated to the active SMAT appearance model component through association matrix are activated for tracking. The contribution each LP makes is determined by its strength of association with the current aspect. Association strengths are updated to reflect the LPs performance for the current aspect each frame.

ciation matrix is updated with these error values. The rate of forgetting is determined by parameter β=0.1, set experimentally and unchanged in all experiments.

$$A_{lm}^{t+1} = ((1-\beta)\cdot A_{lm}^{t}) + (\beta \cdot \|\delta\mathbf{x}_l - \bar{\delta\mathbf{x}}\|) \tag{4.4}$$

This record of LP performance provides a method for weighting each LP's contribution to overall tracker output, $\bar{\delta\mathbf{x}}$, defined in Eq. 4.5 and 4.6.

$$W_l^m = \begin{cases} 1 - \frac{\mathbf{A}_{lm}}{\max(\mathbf{A}_{im})}, i = 1...L & \text{if } \mathbf{A}_{lm} > 0; \\ 0 & \text{if } \mathbf{A}_{lm} = 0. \end{cases} \tag{4.5}$$

$$\bar{\delta\mathbf{x}} = \frac{\sum_l^L W_l^m \delta\mathbf{x}_l}{\sum_l^L W_l^m} \tag{4.6}$$

A further advantage of maintaining a performance metric on each LP-aspect association is that it allows poorly performing LPs to be replaced by LPs learnt online. A new predictor is learnt for every frame from synthetic displacements of the previous frame and is evaluated on its prediction of the current frame. The worst predictor, ϕ, is identified from the current active component m^* using Eq. 4.7.

$$\phi = \underset{l}{\operatorname{argmax}} A_{lm^*}, l = 1...L. \tag{4.7}$$

If the prediction error of the new LP is less than the ϕ^{th} (worst) LP's error, $\|\delta\mathbf{x}_{new} - \bar{\delta\mathbf{x}}\| < \|\delta\mathbf{x}_\phi - \bar{\delta\mathbf{x}}\|$, then the new predictor replaces the ϕ^{th} (only in the current active component). This process serves both to introduce view-specific predictors as well as prevent outliers from contributing to the tracker output. Note that a predictor can be used by multiple components and is only completely destroyed if it has zero values for all components.

Note that when a new component of the appearance model is created all the predictors from the previously used component are assigned to the new component by copying a column of $\mathbf{A}$.

The complete LP-SMAT tracking algorithm is summarised in Algorithm 2.

4.3.3 LP-MED(oidshift): Online partitioning of linear predictors for tracking

Similarly to the LP-SMAT approach, by learning aspect specific predictor weightings, each predictor can be associated to a greater or lesser extent to each aspect. However, the medoidshift clustering approach does not have a predetermined number of clusters, as in the SMAT model. The flexibility of the model is further enhanced by the possibility for appearance templates to change their cluster membership as the dataset is expanded incrementally. In order to utilise this clustering for partitioning the set of

Algorithm 2 LP-SMAT tracking procedure

$\mathbf{F}^0 \leftarrow$first image

Initialise target position $\bar{\mathbf{x}}^0$, height h and width w from user input

$M \leftarrow 4, L \leftarrow 160$

for $l = 0$ to L/M **do**

$\mathbf{x}^l = \{rand(-h/2 : h/2), rand(-w/2 : w/2)\}$ {Randomly select reference point}

Generate $\{\delta\mathbf{x}_i, \mathbf{d}_i\}$ {Training data}

Compute $\mathbf{P}^l$ as is Eq. (3.7)

$\mathbf{A}_{l,m=1} = 1$ {Assign all initial predictors to first mode with equal weight}

$m^* = 1$ {Set first mode as active}

end for

$t \leftarrow 0$

while $\mathbf{F}^t \neq NULL$ **do**

Compute $\delta\mathbf{x}^l$ as in Eq. (3.6) $\forall l \exists\ A_{l,m^*} > 0\ l = \{0...L\}$

Compute $\delta\bar{\mathbf{x}}$ as in Eq. (4.6)

Update predictor states $\mathbf{x}^l = \mathbf{x}^l + \delta\bar{\mathbf{x}}$

Update association matrix, $\mathbf{A}$, as in Eq. 4.4

Identify the worst predictor, ϕ, from the current active component m^* using Eq. 4.7.

Extract new appearance template $\mathbf{G}^t$

Obtain $m^* \subset \{1...M\}$ {Active component obtained by greedy assignment of new template to model component}

Assign template $\mathbf{G}^t$ to m^* component

Update m^* component mean and threshold as in Eq. 4.2 and 4.3.

Learn new predictor as in Eq. (3.7)

if new predictor performance $\geq$ old predictor performance **then**

Replace worst predictor ϕ

Update association matrix, $\mathbf{A}$, as in Eq. (4.4)

end if

$t \leftarrow t + 1$

end while

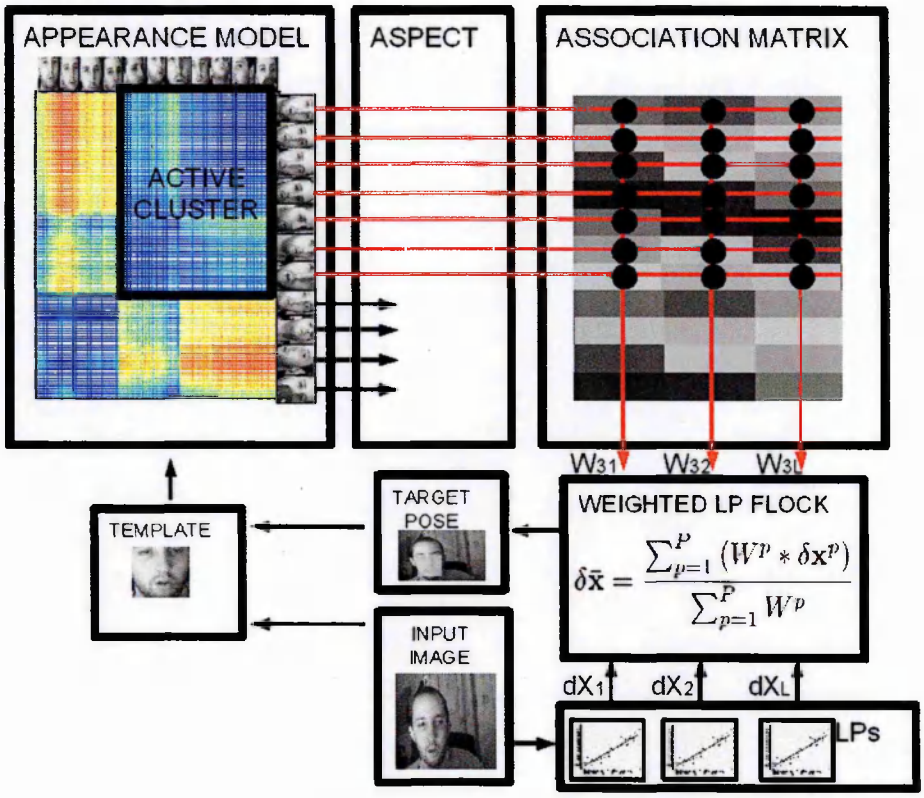


Figure 4.8: *LP-MED system architecture*: The appearance templates are incrementally clustered using the medoidshift modes seeking algorithm. Each LP makes a prediction each frame and the level of contribution made is determined by its performance during each of the frames that form part of the current appearance aspect.

LPs, a flexible mechanism for associating clusters to LPs is required. This is achieved by maintaining a record of the performance of each LP for each template as opposed to each component in the SMAT model. A combination of template membership and these performance measures are used to compute a strength of association between each LP and any aspect.

The weighting mechanism is achieved by an association matrix, $\mathbf{A}$, as illustrated in figure 4.8. Given a bank of L linear predictors and a set, $\mathbf{T}$, of M appearance templates, $\mathbf{T} = \{\mathbf{G}^0 ... \mathbf{G}^M\}$, the association matrix $\mathbf{A}$ has dimension $(L \times M)$. Note that M is much larger here than for the SMAT model where M indicates the number of modes rather than the number of templates. The value at $\mathbf{A}_{lm}$ indicates the strength (or

weakness) of association between predictor l and template (exemplar) m. The values of $\mathbf{A}$ are set and updated using Eq. (4.8) and (4.9). Equation (4.8) shows how the prediction error is computed and used to initialise the association values between each predictor and the new appearance template m^t. The error is the flock agreement error, as in the LP-SMAT approach, and as detailed in section 3.1.4.

$$\mathbf{A}_{lm^t} = \|\bar{\delta\mathbf{x}} - \mathbf{x}^l\|, l = 1...L \tag{4.8}$$

The association values for all the other templates in the active aspect, $\mathbf{T}_{m^*} \subset \mathbf{T}$, are then updated as follows, for all predictors $l = 1...L$:

$$\mathbf{A'}_{lm} = \begin{cases} ((1-\beta)\cdot\mathbf{A}_{lm}) + (\beta\cdot\|\bar{\delta\mathbf{x}} - \mathbf{x}^l\|), & \text{if } \mathbf{G}^m \in \mathbf{T}_{i^*} \\ \mathbf{A}_{lm} & \text{if } \mathbf{G}^m \notin \mathbf{T}_{i^*} \end{cases} \tag{4.9}$$

This has the effect of smoothing the performance measures within a cluster. The values are a running average prediction error with exponential forgetting; meaning that low values of $\mathbf{A}_{lm}$ indicate greater association between predictor l and clusters containing exemplar m. As in the LP-SMAT model, the rate of forgetting is determined by parameter β=0.1, set experimentally. In all the experiments $M \leq 200$ - meaning after 200 frames, the oldest template is removed from the model - and L=80. These parameters are also set experimentally.

This error function and update strategy are used to continually evaluate predictor performance over time. This provides a means for appearance dependent weighting of each predictors contribution to overall tracker output, $\bar{\delta\mathbf{x}}$, as defined in Eq. (4.10) and Eq. (4.11).

$$w^l = 1 - \frac{\sum_{\forall m \exists \mathbf{T}_{m^*}} \mathbf{A}_{lm}}{\max \sum_{\forall m \exists \mathbf{T}_{m^*}} \mathbf{A}_{lm}} \tag{4.10}$$

$$\bar{\delta\mathbf{x}} = \frac{\sum_{l=1}^{L} (w^l \cdot \delta\mathbf{x}^l)}{\sum_{l=1}^{L} w^l} \tag{4.11}$$

The continuous evaluation of predictor performance also allows poorly performing predictors to be replaced by predictors learnt online. The worst predictor, l^*, is identified

Table 4.2: Parameters settings for all methods.

Parameter	Meaning	Value
r_{sp}	Range around LP reference point within which support pixels are sampled.	20
r_{tr}	Maximum magnitude of displacements used for training LP	30
k	LP complexity: number of support pixels.	150
N	Training complexity: number of training examples.	100
L	Max. number of predictors across all modes.	160(LP-SMAT) 80(LP-MED)
M	SMAT: Max. number of modes in model.	4
M	LP-MED: Max. number of appearance templates.	200
α	SMAT model learning rate	0.2
β	LP-SMAT and LP-MED rate of forgetting for association updates	0.1

as in Eq. (4.12). The LP whose minimum error (over all exemplars) is greatest of all minimum errors (over all LPs) is selected.

$$l^* = \underset{\{l=1,\ldots,L\}}{\operatorname{argmax}} \left(\min_{\{m=1,\ldots,M\}} \mathbf{A}_{lm} \right) \tag{4.12}$$

The entries in $\mathbf{A}$ relating to the replaced predictor are updated as in Eq. (4.13).

$$\mathbf{A}_{l^*m} = \frac{\sum_{l=1}^{L} \mathbf{A}_{lm}}{L}, m = 1 \ldots M \tag{4.13}$$

The entries in $\mathbf{A}$ relating to the replaced LP are averaged across all LPs for each exemplar. The complete tracking algorithm is summarised in Algorithm 3.

4.4 Tracking evaluation

This section presents experiments evaluating the tracking performance in terms of accuracy and efficiency and provides comparison to other state of the art simultaneous

Algorithm 3 LP-MED tracking procedure

$\mathbf{F}^0 \leftarrow$first image

Initialise target position $\bar{\mathbf{x}}^0$, height h and width w from user input

for $l = 0$ to L **do**

$\mathbf{x}^l = \{rand(-h/2 : h/2), rand(-w/2 : w/2)\}$ {Randomly select reference point}

Generate $\{\delta\mathbf{x}_i, \mathbf{d}_i\}$ {Training data}

Compute $\mathbf{P}^l$ as is Eq. (3.7)

$w^l \leftarrow 1$ {Set all initial predictor weights to 1}

end for

$t \leftarrow 0$

while $\mathbf{F}^t \neq NULL$ **do**

Compute $\delta\mathbf{x}^l$ as in Eq. (3.6) for l = {0 ... L}

Compute $\delta\bar{\mathbf{x}}$ as in Eq. (4.11)

Update predictor states $\mathbf{x}^l = \mathbf{x}^l + \delta\bar{\mathbf{x}}$

Extract new appearance template $\mathbf{G}^t$

Compute new row and column of distance matrix, L_2 norm $\mathbf{G}^t$ and $\{\mathbf{G}^0 ... \mathbf{G}^{t-1}\}$

Obtain $\mathbf{T}_{i^*} \subset \{\mathbf{G}^0 ... \mathbf{G}^{t-1}\}$ {Obtained by clustering $\mathbf{T} = \{\mathbf{G}^0 ... \mathbf{G}^{t-1}\}$}

Update association matrix, $\mathbf{A}$, as in Eq. (4.9)

Identify worst predictor as in Eq. (4.12)

Learn new predictor as in Eq. (3.7)

if new predictor performance $\geq$ old predictor performance **then**

Replace worst predictor l^*

Update association matrix, $\mathbf{A}$, as in Eq. (4.13)

end if

Compute predictor weightings for next frame as in Eq. (4.10)

$t \leftarrow t + 1$

end while

modelling and tracking approaches. First each of the investigated trackers is reviewed, then the datasets used are detailed and tracking performance is evaluated. Videos demonstrating each of the trackers on the sequences are available here[2].

Trackers[3]: The trackers under investigation in this section are:

1. **LK** - the inverse compositional LK tracker using L_2 norm and Levenberg-Marquardt optimisation,
2. **LK-SMAT** - as described in section 4.3.1 and [20],
3. **LP-FLOCK** - as in section 3.1.4 with 60 LPs.
4. **LP-SMAT** - as in section 4.3.2,
5. **LP-MED** - as in section 4.3.3,
6. **Online-Boost** - The tracker introduced by Grabner et al. [31] that tracks by online boosting discriminative foreground/background classifiers, and
7. **Semi-Online-Boost** - the online boosting tracker with a semi-supervised classifier update [32].

Results for tracker (7) are only presented if and when tracker (6) is shown to fail where other techniques succeed. All parameters for trackers (6) and (7) are default and for all other trackers are as detailed in table 4.2 and no parameter tuning is performed.

Datasets[4]: The datasets used for evaluation are detailed in table 4.3. The **Car-Surveillance** is a benchmark sequence in the IEEE International Workshops on Performance Evaluation of Tracking and Surveillance (PETS'2000) featuring a car from a surveillance camera. The **Dudek-Face** sequence was presented in [41] to demonstrate the trackers handling of appearance changes and un-modeled pose deformations. The

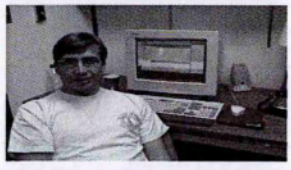
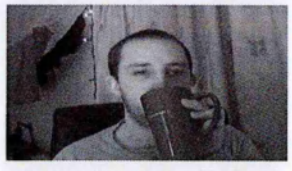
[2] www.cvl.isy.liu.se/research/adaptive-regression-tracking

[3] Links to implementations for trackers (1) and (2) available at www.cvl.isy.liu.se/research/adaptive-regression-tracking and for (6) and (7) at www.vision.ee.ethz.ch/boostingTrackers

[4] All datasets and ground-truth (where present) available at www.cvl.isy.liu.se/research/adaptive-regression-tracking

Runner sequence is a typical track athletics sequence. The **Head-Motion** is a sequence of a moving head and torso, and the **Camera-Shake** sequence is taken from a moving webcam pointing at a phone and undergoing vigorous shaking causing motion blur and large inter-frame displacements.

Table 4.3: Summary of datasets

Name	Image	#frames	Introduced
Car-Surveillance		282	PETS'2000
Dudek-Face		1144	Jepson et al (2001)
Runner		400	Dowson et al (2006)
Head-Motion		2350	New
Camera-Shake		989	New

4.4.1 Car-Surveillance sequence

For the Car-Surveillance sequence, the target was successfully tracked by the LP-SMAT, LP-MED and Online-Boost in all 282 frames (as the Online-Boost tracker is successful the Semi-Online-Boost tracker is not tested). All other trackers fail to track the target through the appearance changes within the first 50 frames. Figure 4.9 shows tracking results for the first last and middle frames of the sequence. Only the LP-MED and

Figure 4.9: *PETS Car-Surveillance sequence results*: Tracking results for LP-MED (solid rectangle) and Online-Boost (dashed rectangle) are shown for first, last and middle frame of sequence along with the median templates identified by the SMAT appearance model during frame 141.

Online-Boost trackers are marked for clarity - the LP-SMAT result is very similar to the LP-MED result in this case. Also shown in figure 4.9 are the four SMAT model median templates at frame 141. The current aspect median is marked with a bullseye. The LP-SMAT tracker operated at an average of 24 frames per second (fps), the LP-MED at an average 20 fps and the Online-Boost at 16 fps.

4.4.2 Dudek-Face sequence

The Dudek-Face sequence was not tracked entirely by any of the tracking approaches under investigation. Figure 4.10 highlights frames from the sequence with each trackers estimated pose marked. At around frame 155 both the LK and the LK-SMAT trackers begin to drift. Between frames 203 and 223 the hand is passed over the face causing the LP-FLOCK tracker to leave the target. The LP-SMAT, LP-MED and Online-Boost trackers are able to track through the hand occlusion. At around frame 364 the glasses are removed from the face, this causes a momentary appearance change as well as a longer term change. All the remaining three trackers cope with this appearance change. At around frame 650 the LP-SMAT tracker begins to drift, tracking only the lower part of the face, this is followed around 100 frames later by the LP-MED tracker loosing the

Figure 4.10: *Dudek-Face sequence results*: Highlighted frames from the Dudek-Face sequence. Tracker key: Dark blue - LK, black - LP-FLOCK, green - LK-SMAT, light blue - LP-SMAT, red - LP-MED, yellow - Online-Boost.

target. Around 50 frames after the LP-SMAT tracker has lost track the Online-Boost tracker also looses track and begins to adapt to the background. By chance the face moves back into the area being tracked and the Online-Boost tracker is able to recover for a short while before losing track again for the last 10 frames. The Semi-Online-Boost tracker was also tested on this sequence, but produced poorer results than the LP-SMAT, LP-MED and Online-Boost trackers. During much of the sequence, the Semi-Online-Boost tracker produces no output and a number of false positive detections. It should be noted that [31] report results showing another version of the Online-Boost boost tracker successfully tracking the Dudek-Face sequence, however these results are not achievable with the simpler implementation that is made publicly available.

The tracked region in the Dudek-Face sequence is 130x130 pixels. The Online-Boost tracker runs at a fairly constant 4 fps and the LP-SMAT and LP-MED trackers run at an average 8 fps and 10 fps respectively. Figure 4.11 compares the computational cost of these three methods. As the frame by frame processing time is not available for the Online-Boost tracker just the average frames per second is plotted. The Online-Boost

algorithm has a very consistent computation time per frame. From this figure it can be seen that the LP-MED tracker also operates at a stable speed after an initial period. This initial high frame rate is due to having few examples in memory and hence a small distance matrix and association matrix. Interestingly, the occasional peaks in the LP-SMAT frame rate (seen in figure 4.11) coincide with significant events in the sequence *e.g.* the first and second peaks (at around 200 frames and 350 frames) coincide with the hand passing over the face and with the glasses being removed respectively. This is due to the creation of new modes during these transient appearance changes. As a new mode will be represented by very few templates the cost of maintaining the distance matrix between each template is low. As the component becomes populated with new templates the cost of maintaining the distance matrix rises again as is shown by the falling frame rate after each event.

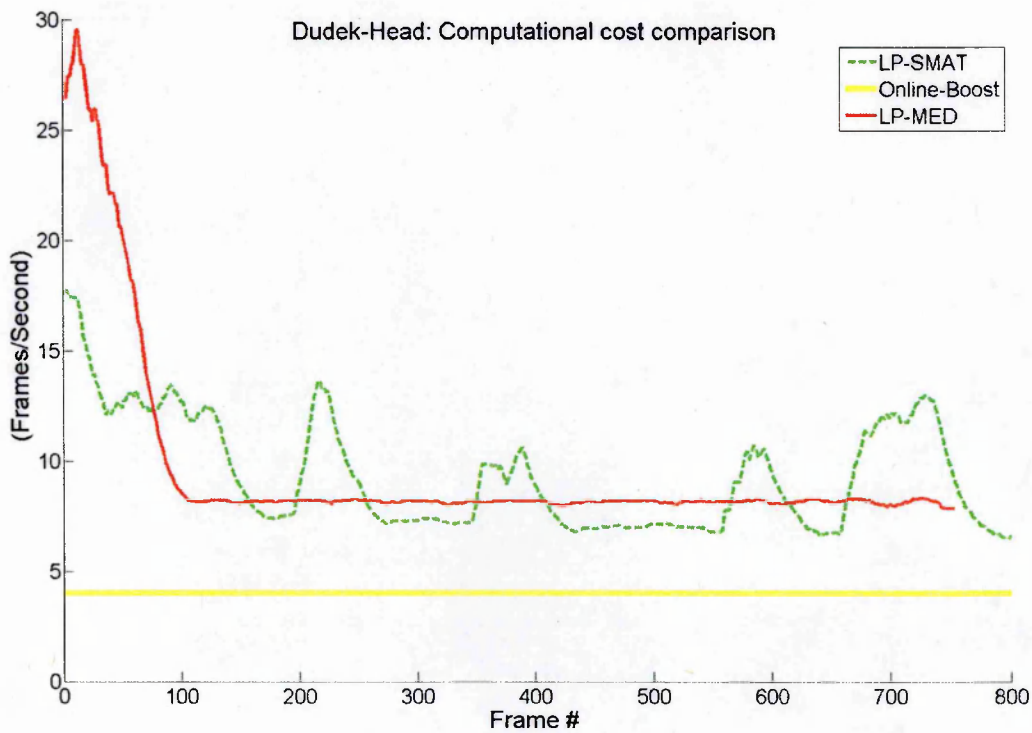


Figure 4.11: *Frame rate comparison on Dudek-Face sequence*: The floating average frames per second is shown for LP-SMAT and LP-MED trackers is shown. The average speed of the Online-Boost tracker on this sequence is shown for reference.

Figure 4.12 shows the SMAT model medians for four key frames. The medians are sorted with decreasing weight left to right. The four key frames are: during and after the hand occlusion, and during and after the removal of the glasses. It can be seen that the SMAT model quickly builds a new mode to represent each of the transient changes of appearance. After the hand passes away from the face the appearance returns to an earlier aspect and so the previously learnt predictor weightings are re-employed. After the glasses have been removed a new mode is created to represent the new appearance and this mode soon has the highest weight *i.e.* most resembles the estimated target appearance.

Figure 4.12: *SMAT model medians for four frames from Dudek-Face sequence*: The medians are sorted with decreasing weight left to right. The four key frames are: during and after the hand occlusion, and during and after the removal of the glasses.

4.4.3 Runner sequence

The Runner sequence features athletes running through the bend of a race track and then down the straight towards the camera. The trackers are initialised in the first frame on the only athlete to remain in the scene for the whole sequence. As can be seen in figure 4.13, the Online-Boost tracker (yellow dashed rectangle) jumps up to start tracking the head and upper torso of the athlete at frame 10 whereas all the other trackers stay tracking the central torso area. This is reflected in the positional accuracy graph in figure 4.14. As the ground-truth position (obtained by hand labeling) is centered on the athlete the Online-Boost tracker accumulates greater area once it starts to track the head of the athlete. Result for the Semi-Online-Boost tracker are not shown here as the tracker fails early on and, due to the significant appearance changes early in the sequence, does not produce any output for most of the sequence. The LK tracker (blue rectangle) begins to drift and loses track completely by frame 150 (refer to figures 4.13 and 4.14). The LP-FLOCK tracker begins to lose track at around frame 100. The Online-Boost, LK-SMAT, LP-SMAT and LP-MED continue to track through significant appearance changes (even though the Online-Boost is tracking a different part of the athlete it does not loose track until around frame 410). The LP-MED tracker stays on the target longer than all the other trackers and gives a more stable track of the target. All trackers fail by frame 420. The frame rates for each tracker on this and all other sequences are given in table 4.4.

4.4.4 Head-Motion sequence

The head tracking sequence consists of 2500 frames with the head undergoing large pose variations and at one point becoming occluded by a cup for over 100 frames. The LK-SMAT, LP-SMAT, LP-MED, Online-Boost and Semi-Online-Boost trackers each track the head throughout the whole sequence but the methods with no online appearance learning (LK and LP-FLOCK) both fail to track the target. Figure 4.15 shows the head being tracked by the LP-MED tracker (top row). On the bottom row of figure 4.15, the position of the LPs are indicated (black spot) as well as the support pixel range, r_{sp}, (white dashed circles). Also marked on the bottom row are the positions of

Table 4.4: Average frame rate per second. Template sizes for each sequence are: CAR - 40x20, DUDEK - 130x130, RUNNER - 38x126, HEAD - 90x100, CAM-SHAKE - 25x25. Parameters for all methods are unchanged for each sequence and set as detailed in the relevant part of section 4.3.

Tracker	**CAR**	**DUDEK**	**RUNNER**	**HEAD**	**SHAKE**
LK-SMAT	12	2	6	6	12
LP-FLOCK	65	16	24	25	65
LP-SMAT	24	10	15	16	35
LP-MED	20	8	12	12	20
Online-Boost	16	4	7	7	16

the worst predictor from the current frame (red mark) and the predictor learnt in the current frame (green mark). As can be seen the worst predictors often lie with most or all the support pixels on the background or, in the case of the cup, on the occluding object. These predictors are not necessarily removed, they may be re-employed later in the sequence when a previous aspect is presented.

4.4.5 Camera-Shake sequence

The Camera-Shake sequence is captured from a low cost web cam and is of a static scene and a moving camera. The camera undergoes considerable shaking causing large inter frame displacements as well as translation, rotation and tilting. Figure 4.16 shows results on this sequence for the LP-MED (red), Online-Boost (yellow solid) and Semi-Online-Boost (yellow dashed) trackers (results for the LP-SMAT tracker are similar to the LP-MED tracker on this sequence, though a little less accurate). All trackers except LP-SMAT and LP-MED fail on this sequence by frame 280 (the onset of some camera shake). The Semi-Online-Boost algorithm is able to re-initialise a number of times during the sequence but is never able to track while the camera is being shaken due to high inter-frame displacements and image blur. The displacement predicted from frame 374 to 375 is 37 pixels (16 vertical and 33 horizontal) and despite the significant

Figure 4.13: *Highlighted frames from the track running sequence*: Tracker key: Dark blue - LK, black - LP-FLOCK, green - LK-SMAT, light blue - LP-SMAT, red - LP-MED, yellow - Online-Boost.

blurring in frame 375, the tracker still succeeds in making a low error prediction. Due to online learning of predictors, some are learnt from blurred images allowing for prediction during this blur. Figure 4.17 shows the positional accuracy plots generated by the six trackers on this sequence. Due to the limited basin of convergence both the alignment based trackers fail to deal with the large inter frame displacements and SMAT loses track as soon as the camera starts to shake. On this sequence the LP-MED approach achieves more accurate results than the other two LP trackers.

4.4.6 Results summary

Of the six trackers evaluated the LK and LP-FLOCK trackers demonstrate the poorest performance. This is due to the significant changes of target appearance in each of the sequences, although for the LK method it is also due to the limited basin of convergence. The LK-SMAT tracker demonstrates the ability to cope with significant appearance changes (as in the Head-Motion sequence) but fails in situations where the target moves too quickly in the image. Both the LK-SMAT and the Online-Boost trackers

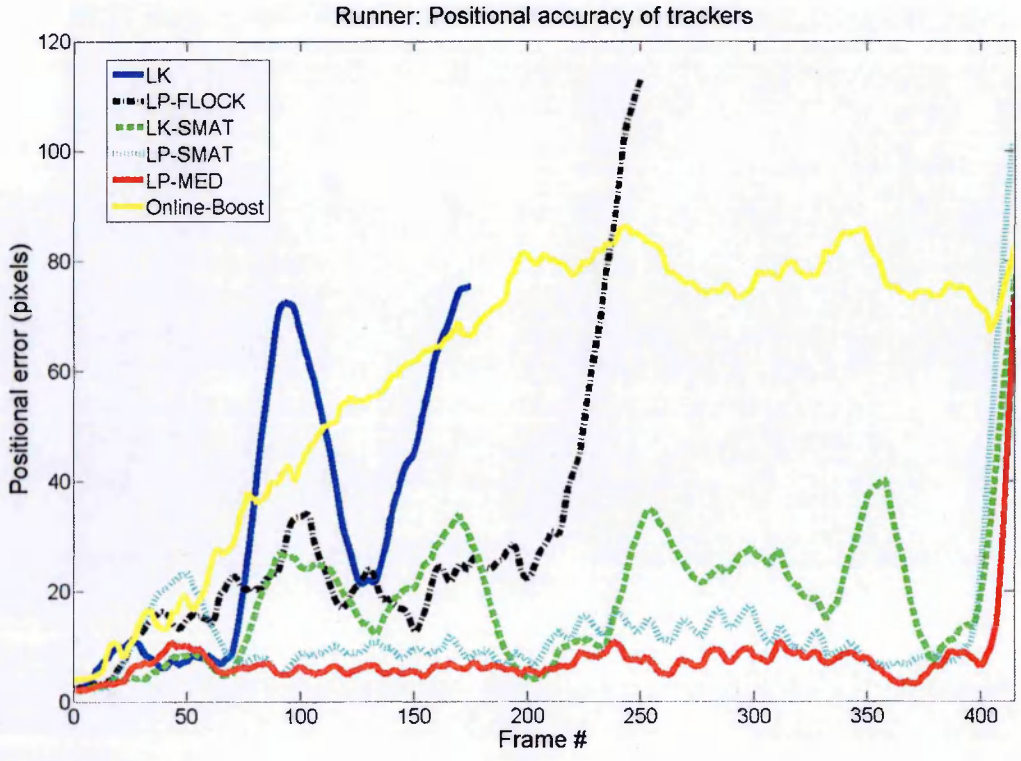


Figure 4.14: *Positional error plots for Runner sequence*: The positional error (in pixels) for each tracker is shown. Ground-truth was obtained by hand labeling the sequence.

fail to handle the large inter-frame displacements present in some of the sequences (e.g. Camera-Shake). On most of the sequences the LP-SMAT, LP-MED and Online-Boost methods achieve similar results, with the LP-MED achieving slightly higher accuracy in the Runner sequence and the Online-Boost achieving marginally better performance on the Dudek-Head sequence.

LK-SMAT operates at the lowest frame rate (frames processed per second) followed by Online-Boost, LP-MED and LP-SMAT with LP-FLOCK achieving by far the highest frame rates. The average frame rate for each method on each sequence is presented in table 4.4. The frame rate for each tracker is determined by the size of region being tracked.

For all sequences the target patch is identified by hand only in the first frame, all algorithm parameters are unchanged between sequences. Ground truth for every frame

Figure 4.15: *LP replacement result*: The medoidshift algorithm tracks the head sequence (top row). LP positions (black dots) and support pixel ranges (white dashed circles) are shown as well as the worst predictor from the current frame (red) and the predictor learnt from this frame (green). It is this process that generates view-specific displacement predictors within the model.

of the athletics and camera motion sequences was achieved by hand labeling and was used to generate the error plots in figures 4.17 and 4.14.

4.5 Evaluation and Conclusion

A general tracking framework that combines online-LP flocks with methods for online appearance model learning has been developed and various configurations investigated. Within this framework, the tracking process provides a mechanism for self supervision of the appearance model learning process and, in return, the appearance model provides information about the structure of the target appearance space that enables the tracker to cope with a high degree of variation in appearance. The only supervision required is to identify the location of the target in the first frame.

Two methods for appearance model learning were introduced. The SMAT model components the appearance space into a predefined number of modes that represent aspects

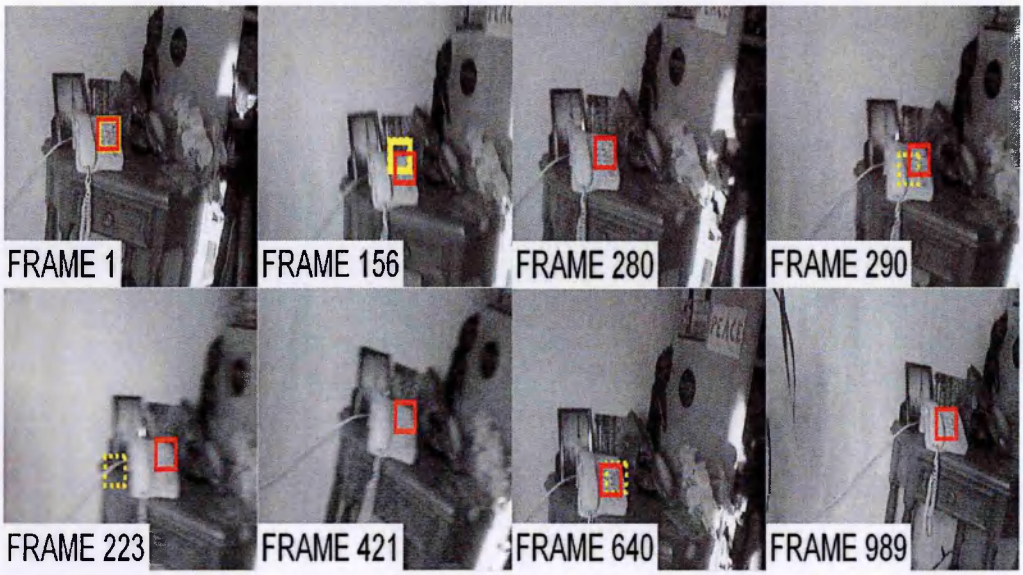


Figure 4.16: *Highlighted frames from Camera-Shake sequence*: Tracker key: Solid dark grey (red) - LP-MED, solid white (yellow) - Online-Boost, dashed white (yellow) Semi-Online-Boost.

of the target. While this does limit the flexibility of the model and may lead to potential misrepresentation of the underlying appearance distribution, the SMAT tracker has demonstrated the ability to adapt to large appearance changes very quickly. The ability to introduce new modes on-the-fly to represent transient target appearance changes (e.g. occlusion of face by hand, see figure 4.12), whilst maintaining unchanged the representation of other parts of the appearance space, proves highly beneficial for tracking many objects.

The medoidshift algorithm on the other hand benefits from the increased flexibility and hence greater likelihood to build more representative models of the object appearance space, regardless of the temporal evolution of the target appearance. This flexibility also extends into the general tracking framework by allowing a more flexible association strategy between aspects of the target and the online-LPs.

The tracking approaches investigated in this work model 2D target transformations (translation). Any higher order transformations of the target (*e.g.* scale or rotation) are considered as appearance changes, and treated as new aspects within the adaptive

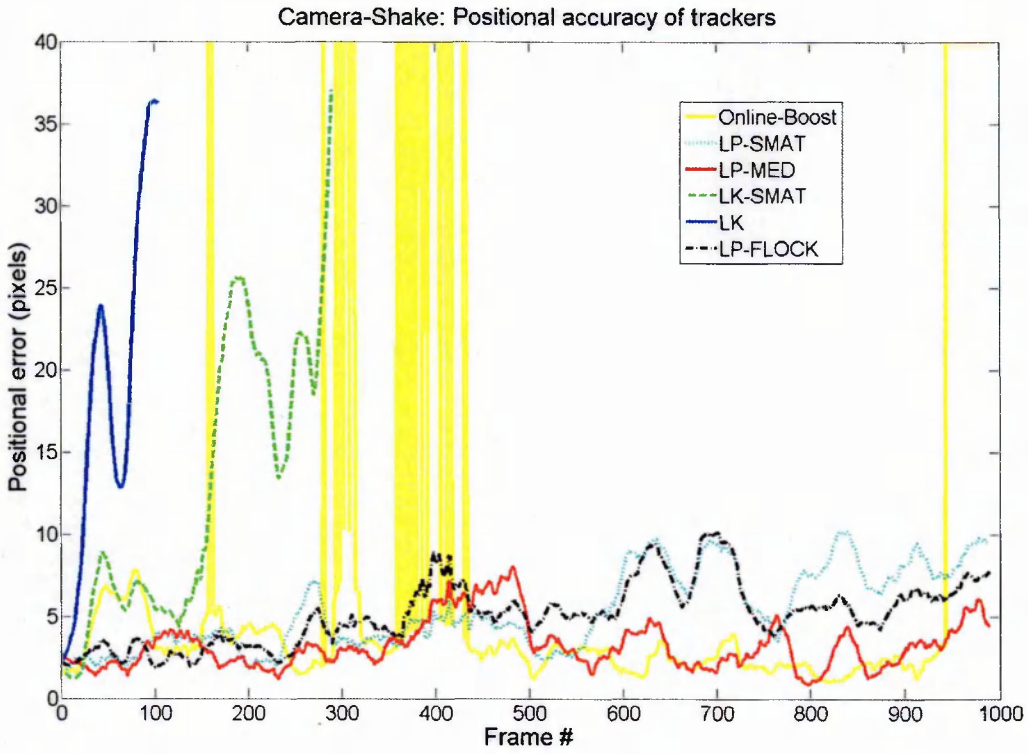


Figure 4.17: *Positional error plots for Camera-Shake sequence*: The positional error for each tracker is shown. Where the Online-Boost tracker fails and is re-initialised is indicated by the vertical yellow lines.

appearance models. Whilst the introduction of higher order transformations is possible, and may considerably improve results in some cases (*e.g.* the Dudek-Face and Runner sequence, in which the target undergoes considerable scale changes), it has been found that the introduction of more parameters into the warping function can increase the risk of drift and the number of local minima [20].

An objective of this work is to delineate the class of problems for which the proposed methods are preferential by including examples of partial failure (Dudek-Face) and examples where this method outperforms the other approaches (Camera-Shake). The success of the LP based techniques on the Camera-Shake sequence highlight the ability of regression techniques to cope with large inter-frame displacements in the presence of local minima. On the other hand, registration techniques tend to achieve higher accuracy provided the displacement is within the basin of convergence.

Other tracking approaches ([64, 41]) have been shown to track the Dudek-Face sequence. Compared to the proposed approach, these methods are slower, but handle the significant appearance changes in this sequence more robustly. Given that these are affine trackers and the trackers compared here only model translation, direct comparison is not included as this would raise issues regarding the DoF used in tracking that are not addressed here.

Both the appearance models developed have demonstrated the ability to adapt to large variations in appearance in order to manage flocks of online-LPs and to facilitate accurate and efficient tracking. When compared to the online boosting methods, there are clearly cases where the discriminative classifiers are better able to represent the target, but for a class of tracking problems the LP-MED/LP-SMAT model provides comparable accuracy with an increase in computational efficiency. An example of a class of targets for which LP-MED and LP-SMAT would fail is wire frame or transparent objects such as a bicycle wheel or a glass of water. The majority of support pixels in a region of an image containing such an object would would lie on the background so this is what would most likely be tracked. The discriminative approach of the Online-Boost method is likely to be able to handle such objects more effectively than the LP based methods.

Many applications require tracking that operates at high frame rates and can handle high object velocities as well as be robust to significant appearance changes and occlusion. This is achieved here by utilising the computationally efficient technique of least squares prediction and online target appearance modeling.

Due to the computational efficiency of the online-LP, the tracker is able to track targets in real time (35/20 frames per second for LP-SMAT/LP-MED), even whilst building and maintaining the appearance model. It is shown that the approach can handle large inter frame displacements and adapt to significant changes in the target appearance with low computational cost. The online-LP tracker has been shown to be particularly effective at tracking through considerable camera shake.

The ability to control the level of each LP's contribution to the overall tracking output enables a high level of adaptability and flexibility to the feature tracker - LP's can be

associated to various aspects of the target feature. Furthermore, evaluating each LP's performance provides the possibility to discard LP's that consistently perform poorly. The process of evaluating, discarding, re-learning and weighting performs a similar optimisation process to that performed in offline training approaches or registration processes such as the Lucas-Kanade tracker, but it does so incrementally whilst the tracker is operating.

The risk of any adaptive system, and specifically trackers that learn about the target appearance online, is that adaptations can, in some cases, cause the model to drift and fail. This lack of robustness in the learning method is caused by learning from noisy or erroneous data. For example, if the LP flock makes an incorrect prediction this would result in the inclusion of an erroneous appearance template being added to the model. Over time this could result in tracking failure, even on data that would not cause failure without the adaptivity.

The advantages of such a simultaneous modeling and tracking approach are clear when considering how much hand crafting, offline learning, hand labeling and parameter tuning must be done in order to employ many existing object tracking approaches. By developing the online-LP and mechanisms to manage LP-flocks, the class of applicable tracking problems has been extended to included, amongst others, automatic seeding of the tracker. For example, a computer vision application may seed a target tracker - such as LP-SMAT or LP-MED - on a region of detected motion, without any prior on target appearance.

This chapter employs unsupervised learning to discover structure in the input (appearance) space and identify target aspects. This structure is used both to classify new inputs and to form subsets of input-output mappings. Partitioning the input-output mappings allows for different categories (or aspects) of the input space to be interpreted using category specific mappings. This idea of using category specific mappings is taken forward in the next chapter. However, the next chapter applies unsupervised learning in the output space, rather than the input space. In the next chapter, the structure discovered in the output (action) space is used to organise the input space to form perceptual categories that are grounded in the agents interactions with the

environment.

Chapter 5

Output Space Clustering for Exemplar Learning

In the previous chapter, unsupervised learning is applied to the input space to discover structure and to identify target aspects. This structure is used both to classify new inputs and to form subsets of input-output mappings. Partitioning the input-output mappings allows for different categories (or aspects) of the input space to be interpreted using category specific mappings. This chapter takes the idea of partitioning the space of mappings to form category specific mappings from the previous chapter. However, this chapter presents an alternative approach, whereby the unsupervised learning is applied to the output space, rather than input space.

This chapter introduces a novel exemplar learning framework applied to the percept-action domain. The system uses minimal hard-coding, favouring learning by example to build generalised input-to-response mappings from percept-action exemplars provided by a human "teacher". Generalisation is achieved by applying unsupervised learning in the output space exemplars. The structure discovered in the output space is used to hierarchically group both input exemplars and output exemplars. This results in the formation of meaningful groupings of both input (perceptual) and output (action) spaces. The variance and invariance of features of the input space groupings characterise the input pattern that predicts its associated class of outputs. Applied to the percept-action domain, this Percept-Action (P-A) hierarchy provides a hierarchy of general to

specific perceptual categories and generative action models, as well as an efficient and context aware hierarchical coarse to fine search in the percept space.

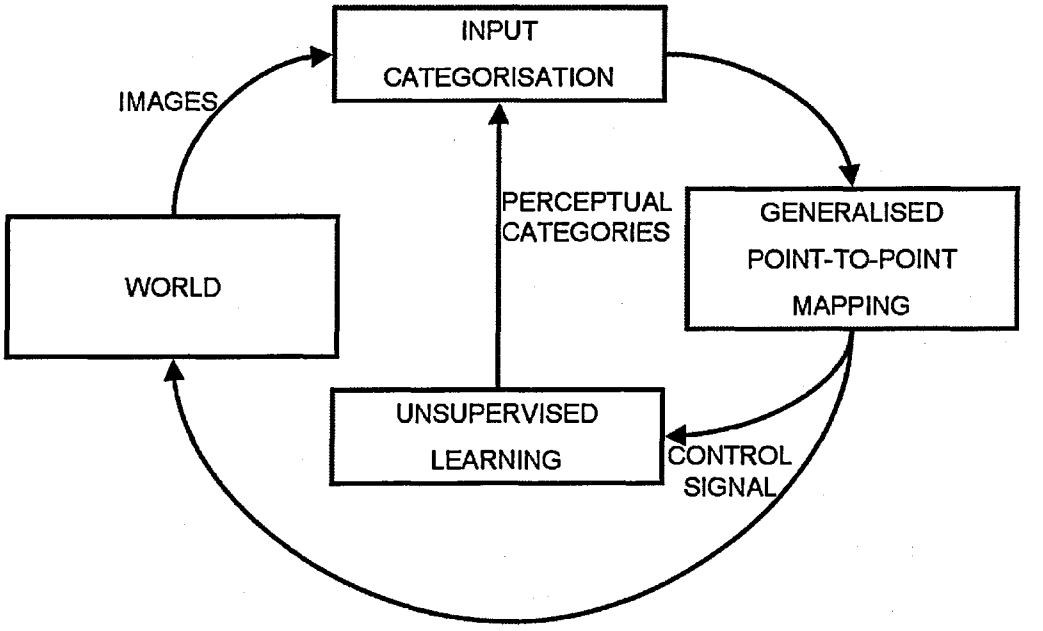


Figure 5.1: *Unsupervised learning in output space for input-output mappings*: This figure represents the approach proposed in this chapter within the PAC framework introduced in chapter 1 and developed in chapters 3 and 4. Unsupervised learning is applied to the problem domain's output space to discover structure in the output space. The structure discovered is used to define perceptual categories that facilitate the categorisation of new inputs.

Figure 5.1 represents the approach proposed in this chapter within the PAC framework that was introduced in chapter 1 and developed in both chapters 3 and 4. This figure illustrates that the discovery of structure in the output space results in the formation of perceptual categories that are used to classify new inputs. The mappings developed here are generalised point-to-point mappings formed by hierarchically clustering percept-action (input-output) exemplars.

The proposed approach is demonstrated in an autonomous navigation task. The agent is able to imitate the behaviour of the teacher to complete a demonstrated task. With only a set of exemplars and no hard-coding or application specific knowledge, the

system is able to identify the visual features of importance to the task and to generate appropriate responses.

The rest of this chapter is organised as follows: In section 5.1 an overview of the proposed PAC modeling approach is presented. As a starting point, a Nearest Neighbour based input-output mapping approach is described. Sections 5.1.1 and 5.1.2 detail methods for partitioning, and generalising over, an exemplar set. In section 5.1.3, the mechanism, central to the approach, of organising both the input and output space in terms of similarity of outputs is detailed. These techniques are then utilised in section 5.2 to build the Percept-Action (P-A) hierarchy, that forms general to specific mappings from percepts to actions. The process of constructing the P-A hierarchy from an exemplar set is detailed in section 5.2.1. Section 5.2.2 describes the process of searching the P-A hierarchy and how outputs are generated in response to new inputs. A discussion of the P-A hierarchy is presented in section 5.2.3. Experiments that evaluate the approach in terms of: ability to generalise, the efficiency of the searching procedure, the perceptual weighting characteristics, and the ability to complete a demonstrated autonomous navigation task, are presented in section 5.3.

5.1 Modelling Percept-Action Cycle (PAC)

In the approach used throughout this thesis, the Percept-Action Cycle (PAC) is represented by a set of percept-action exemplar vectors, $E = \{(\mathbf{p}^1, \mathbf{a}^1), ..., (\mathbf{p}^N, \mathbf{a}^N)\}$, where $P = \{\mathbf{p}^1 ... \mathbf{p}^N\}$ is the set of percept vectors, $\mathbf{p}^n = [p_1^n, ..., p_i^n] \in \Re^i$ (real numbered vectors with i dimensions) and $A = \{\mathbf{a}^1 ... \mathbf{a}^N\}$ is the set of action vectors, $\mathbf{a}^n = [a_1^n, ..., a_j^n] \in \Re^j$ (j dimensions). The task is to obtain a mapping $\Re^i \rightarrow \Re^j$ that best approximates the relationships implicit in the exemplars.

If the exemplar set, E, accounts for every possible situation the system may encounter, then generating an appropriate response becomes a simple case of recall. A function, $f : P \rightarrow A$, that maps the set P onto set A will suffice, for example a simple lookup table or *point-to-point* mapping. However, even in some trivial domains, this approach is intractable. Obtaining and storing such an exemplar set is infeasible in most practical domains. The problem is how to generalise over an incomplete set of exemplars.

One approach to generalising a point-to-point mapping is to relax the input matching criteria. The Nearest Neighbour (NN) search procedure finds the closest exemplar in a set P, to a query point $\mathbf{p}$. The NN percept-action mapping (NN-PA) is illustrated in figure 5.2. The NN-PA approach selects action $\mathbf{a}^n$ where,

$$n = \underset{n=1}{\overset{N}{\operatorname{argmin}}} ||\mathbf{p}^n - \mathbf{p}|| \tag{5.1}$$

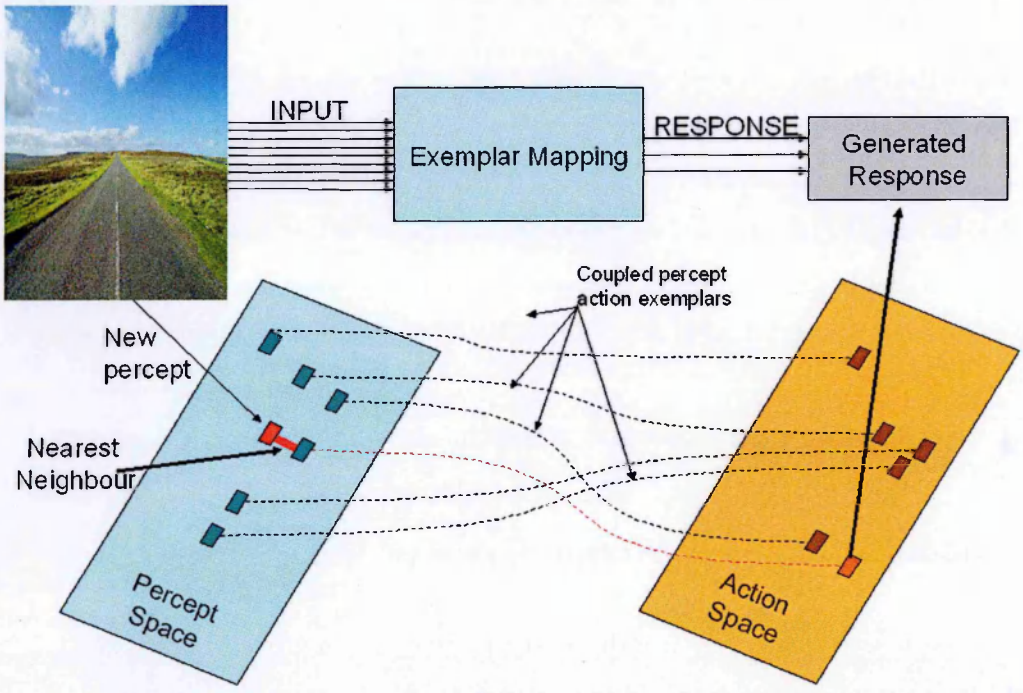


Figure 5.2: *NN-PA mapping*: A nearest neighbour search in the input space is used to identify best matching percept. The generated response is the action coupled to the best matching percept.

The NN-PA approach suffers from two significant drawbacks. First, the $\mathcal{O}(Ni)$ complexity of naive NN search procedures scales linearly with the size of the exemplar set, N, and the dimensionality of the data, i. Second, there is no generalisation of the action space. This means the mapping never generates new outputs and so the outputs do not reflect the differences between query percepts and exemplar percepts. The latter of these drawbacks is a particular problem for sparse exemplar sets where new inputs match to unpopulated regions of the percept exemplar space. Increasing

the size of the exemplar set goes some way to alleviating this problem but, as a result of the first drawback, will incur increased computational cost. Furthermore, as already stated, complete population of the exemplar set is infeasible.

5.1.1 Partitioning the search space

To overcome the drawback of the linear complexity of the naive search, the search space can be hierarchically partitioned. This allows for regions of the search space to be disregarded early on in the search procedure. A KD-tree provides an efficient approach to this partitioning task [28, 7]. The KD-tree iteratively bisects the exemplar space into two regions containing half of the points of the parent region. Queries are performed, with average complexity $\mathcal{O}(logN)$, via traversal of the tree from the root to a leaf by evaluating the query point at each split. The KD-tree partitioning of six data points is illustrated in figure 5.3. The leaf nodes of the tree contain single data points whereas branching nodes contain decisions representing partitions of the data.

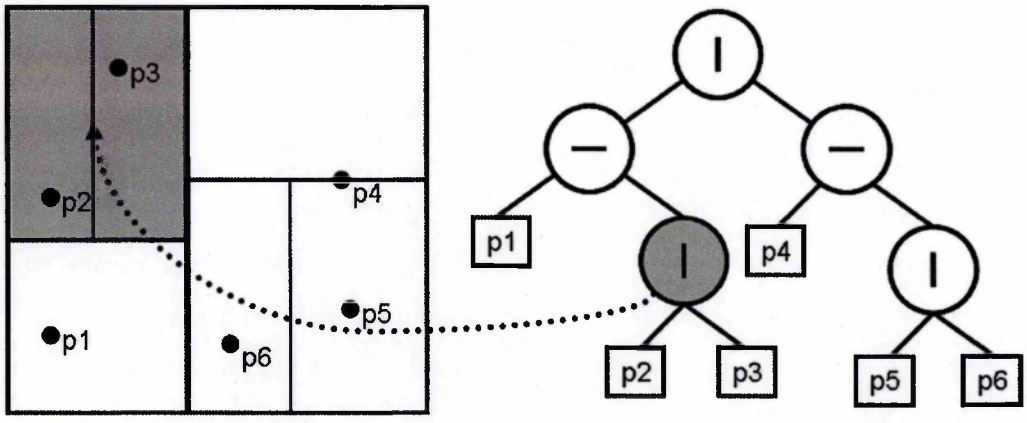


Figure 5.3: *KD-tree partitioning*: A recursive subdivision of the search space facilitates an efficient nearest neighbour search.

The hierarchical partitioning provides an efficient search structure but does not, in itself, provide a solution to generalising over the action exemplars. A mapping using the KD-tree to search (rather than the NN search as in the NN-PA mapping) can still only select one of the stored exemplars for output. However, the hierarchical structure does offer more than just an efficient search procedure. Each decision partitions the complete

set of exemplars into two subsets and these subsets are partitioned into two further subsets and this continues recursively until the new subset has only one member, at the leaf node. This hierarchy of general to specific subsets provides a starting point for generalising over the exemplars. If the characteristics of a subset can be identified, then in the percept space, a new percept can be classified as either belonging to that subset or not, based on whether it has similar characteristics. This requires a *discriminative* percept classifier learnt on the subset of percept vectors. Also, the characteristics of a subset of actions could be used to generate new actions with similar characteristics. This requires a *generative* action model learnt on the subset of action vectors. A subset of percepts forms a perceptual category and a subset of actions forms a class of action.

5.1.2 Generalising over an exemplar set

To generalise over an exemplar set, E, or any subset of E, some way of characterising the set members is required. Additionally, some method of generating new exemplars that have the same characteristics is needed, for the generation of new actions.

The multivariate Gaussian Probability Density Function (PDF) provides a tool for generalising the exemplar set. It provides a means of characterising the set members, in terms of the mean and variance. The PDF can also be used to generate new exemplars that have the same characteristics, by drawing values from the distribution. Assuming the action exemplar vector, $\mathbf{a} = [a_1, ..., a_j]$, to be a random variable with mean vector $\bar{\mathbf{a}}$, and covariance matrix Σ_a, the multivariate Gaussian PDF is as in equation 5.2.

$$\mathcal{N}_j(\bar{\mathbf{a}}, \Sigma_a) = \frac{1}{(2\pi)^{j/2}|\Sigma_a|^{1/2}} exp(-\frac{1}{2}[\mathbf{a} - \bar{\mathbf{a}}]^T \Sigma_a^{-1}[\mathbf{a} - \bar{\mathbf{a}}]) \tag{5.2}$$

The shorthand notation $\mathcal{N}_j(\bar{\mathbf{a}}, \Sigma_a)$, is used to state that j-dimensional random vector $\mathbf{a}$ is a multivariate Gaussian PDF, with mean vector, $\bar{\mathbf{a}}$, and covariance matrix Σ_a, which are computed from the exemplars as in equations 5.3 and 5.4.

$$\bar{\mathbf{a}} = \frac{\sum_{n=1}^{N} \mathbf{a}^n}{N} \tag{5.3}$$

$$\Sigma_a = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{a}^n - \bar{\mathbf{a}})(\mathbf{a}^n - \bar{\mathbf{a}})^T \tag{5.4}$$

Building a multivariate Gaussian PDF, $\mathcal{N}_j(\bar{\mathbf{a}}, \Sigma_a)$ from each of the action exemplar subsets identified by the hierarchical partitioning, generates a hierarchy of general to specific generative action models. The hierarchy of general to specific subsets of percept exemplars also provides a structure for classifying new percepts as belonging to any of the general to specific percept categories. The multivariate Gaussian PDF, fitted to the perceptual subsets, $\mathcal{N}_i(\bar{\mathbf{p}}, \Sigma_p)$, can form the basis of a discriminative classifier. The Mahalanobis distance function provides a means of determining the similarity, $D_M(\mathbf{p}, \mathcal{N}_i(\bar{\mathbf{p}}, \Sigma_p))$, of a query sample, $\mathbf{p}$, to a known distribution, $\mathcal{N}_i(\bar{\mathbf{p}}, \Sigma_p)$. The Mahalanobis distance function is given in equation 5.5.

$$D_M(\mathbf{p}, \mathcal{N}_i(\bar{\mathbf{p}}, \Sigma_p)) = \sqrt{(\mathbf{p} - \bar{\mathbf{p}})^T \Sigma_p^{-1} (\mathbf{p} - \bar{\mathbf{p}})} \tag{5.5}$$

The Mahalanobis distance function takes into account the covariance of the data set.

5.1.3 Output space clustering

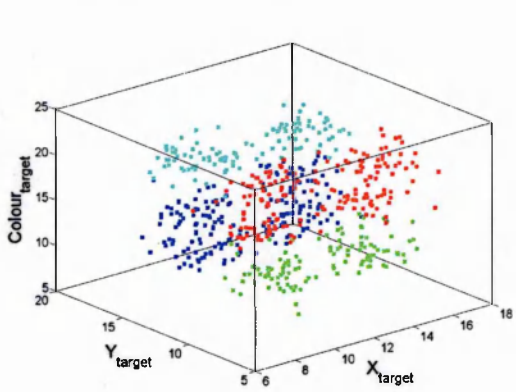
An intuitive approach is to apply the hierarchical partitioning to the input or percept vector space. Given that the search procedure is carried out in the input space, a conventional hierarchical search structure would subdivide the exemplar set based on the parameters being compared to the query. It might also be expected that the subsets of outputs formed by the input partitioning, would represent a certain meaningful class of outputs or actions. The assumption being that similar problem configurations (inputs) will require similar problem solutions (outputs). However this assumption breaks down for problems where a small change in the input require significantly different responses or conversely when inputs appear quite different but require similar responses. Essentially, implicit in the assumption is that the distance function used for partitioning the data, reflects the relative importance of features of the input space in determining the response. Such a distance function could in some cases be designed - specified by an engineer - but this is a form of hard-coding and is not a generic solution.

The proposed solution is to partition input and output exemplars in the output vector space. In general, related points in the action domain exhibit greater continuity and are more similar than related points in the percept domain [33]. Furthermore, the dimensionality of the action space $\Re^j$ will generally be significantly lower than the dimensionality of the percept space $\Re^i$, $i > j$. These observations imply that partitions of the action space vectors are more likely to reflect the underlying structure of the data than partitions in the percept space.

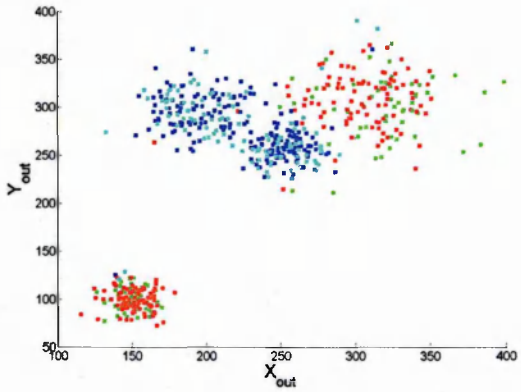
By basing the partitioning of exemplars on similarity in the action space, the resulting action subsets will contain similar members and will therefore result in meaningful action models, $\mathcal{N}_j(\bar{\mathbf{a}}, \Sigma_a)$. In the proposed approach, the percept exemplars are also partitioned based on the partitioning of the action exemplar vectors. The resulting subsets form groups of percepts that each illicit a similar action. Therefore the subsets form meaningful perceptual categories, because they represent the perceptual stimulus for a meaningful class of actions. The covariance, Σ_p, of perceptual features within a category reflect the relative importance of each dimension of the percept data to the associated class of action. Modelling the percept groups as multivariate Gaussian PDFs, $\mathcal{N}_i(\bar{\mathbf{p}}, \Sigma_p)$, and using the Mahalanobis distance, equation 5.5, to compute the similarity of a query percept to a perceptual category, provides a mechanism of classifying perceptual information based on the correlations and covariances of the category.

To illustrate this output space based input-output exemplar clustering approach, a synthetic data set of percepts and actions is clustered in both input and output space. Figure 5.4 shows the results of these two clustering approaches on the data. The data set is constructed to simulate the data that would be obtained from a simple percept-action task. The percept exemplars are generated from twelve 3-dimensional Gaussian PDFs with means evenly spaced about a cube at equal intervals from each other. Dimensions one and two could represent the position of a target object in a percept-action task. Dimension three could represent the colour of the target object. To generate the output exemplars, all the percept exemplars drawn from PDFs with the same mean position (mean of first two dimensions of percept space) are scaled and transposed by an arbitrary transform. Thus it is the two dimensions that represent the position of the target that determines the output value.

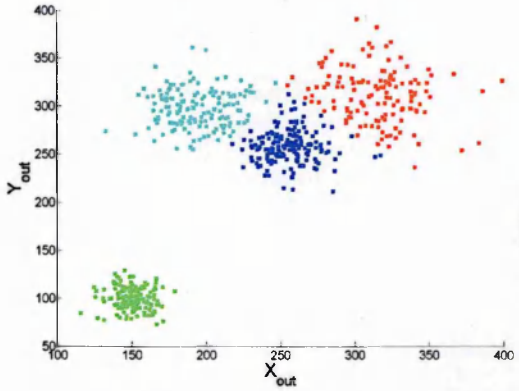
Figure 5.4(a) shows the result of running a k-means clustering on the percept data, and figure 5.4(b) shows the resulting groups formed in the action data. Clearly clustering the data set in the percept space does not produce meaningful action groupings. Figure 5.4(c) shows the result of running the k-means clustering on the action exemplars, and figure 5.4(d) shows the resulting groups formed in the percept data. In this case, the output space clustering has identified the predictor variables, i.e. the target position, and the redundant input variable, the colour of the target. Alternatively, if the same action exemplars were generated from percept exemplars with the same mean colour, the output clustering would identify the colour was the action predictor variable.



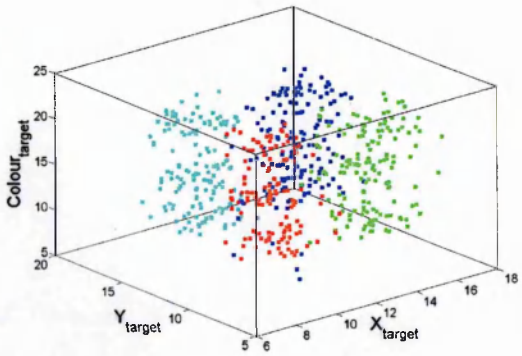
(a) Input clustering of percept exemplars.



(b) Input clustering of action exemplars.



(c) Output clustering of action exemplars.



(d) Output clustering of percept exemplars.

Figure 5.4: *Output space vs. input space clustering*: Clustering output exemplars based on similarity of inputs results in meaningless groups of actions. Clustering input exemplars based on similarity of outputs identifies important variance and invariance of perceptual groups.

This discussion, on percept vs action space partitioning is summarised in figure 5.5. The figure illustrates how the structure discovered in the percept space can result in the formation of meaningless action models. However when the structure discovered in the action space is imposed on the perceptual space, the result is both meaningful action models and meaningful perceptual categories. This finding is consistent with Granlund's postulate quoted in the introduction to this thesis, "Related points in the response domain exhibit a much larger continuity, simplicity and closeness than related points in the input domain. For that reason, the organisation process has to be driven by the response domain signals." [33].

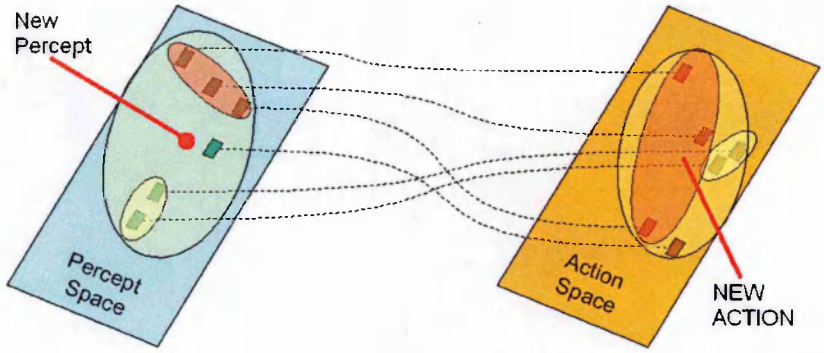
5.2 The Percept-Action (P-A) Hierarchy

This section details the Percept-Action (P-A) hierarchy, that combines the hierarchical partitioning, exemplar set generalisation and output space clustering techniques just described. The P-A hierarchy is composed of a hierarchy of general to specific percept classifiers that are each associated to a probabilistic action generation model and provides an efficient mechanism for responding to new percepts. First, the offline process of building the P-A hierarchy is detailed in section 5.2.1. In section 5.2.2 the process of searching the P-A hierarchy for best matching perceptual categories, and generating responses from the associated action models, is described. A discussion of the P-A hierarchy is presented in section 5.2.3.

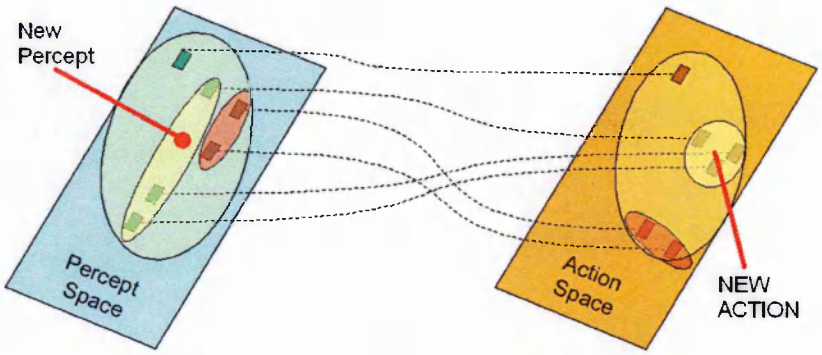
5.2.1 Building the hierarchy

The first process in building the P-A hierarchy is to compute the KD-tree partitioning of the set of action vectors, $A = \{\mathbf{a}^1 ... \mathbf{a}^N\}$. The KD-tree algorithm employed [1] is based on a recursive subdivision of the parameter space into disjoint hyperrectangular regions. Any region containing more than one data point (action vector) is split into two boxes by an axis-orthogonal hyperplane that intersects the regions hyperrectangle. The hyperplane is selected using the 'sliding midpoint rule' in which a hyperplane is

[1] http://www.cs.umd.edu/~mount/ANN/



(a) Percept space clustering. Although grouping percepts that look similar may seem intuitive, the resultant action groupings can be meaningless.



(b) Action space clustering. Grouping experiences in terms of structure discovered in the action space results in the formation of groups of percepts. Analysis of the variance of the percept groups allows the definition of classifiers for particular modes of action.

Figure 5.5: *Percept vs action space clustering*: The variances and co-variances of input features within groups of inputs associated to action clusters reflect the relative importance of the features to the matching criteria. This is the process of forming perceptual categories by structuring the perceptual space according to structure discovered in the action space.

first selected that cuts the current hyperrectangle through its midpoint orthogonal to its longest side and then, if the case arises that all the points lie on one side of the hyperplane then the hyperplane is translated toward the points until the first data point is encountered. The result of performing this partitioning on an exemplar set of control signals for navigating a robotic platform, is illustrated in figure 5.6.

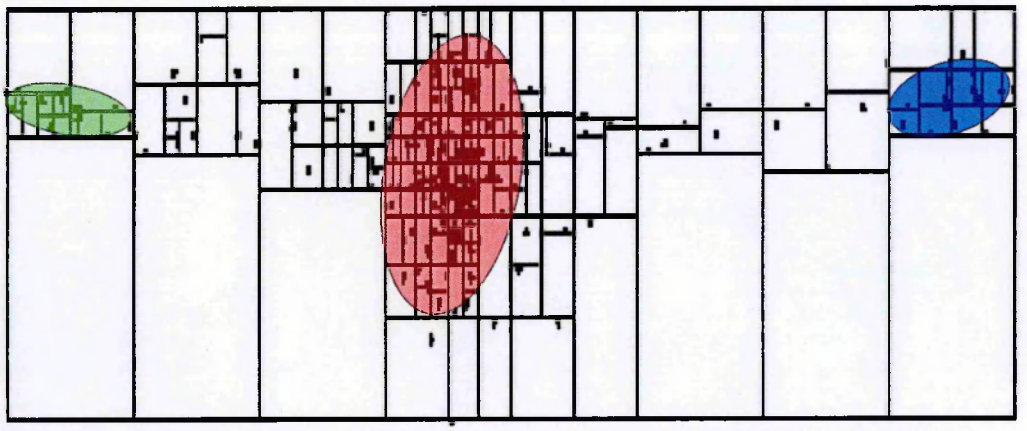


Figure 5.6: *Action space KD-tree partitioning with action models*: The KD-tree partitioning of action exemplars collected in a autonomous navigation task is shown. Also marked on the figure are ellipses representing the action model Gaussian PDFs fitted to three of the subsets. The green and blue ellipses represent models of 'hard-left' and 'hard-right' actions. The red ellipse represents a general model representing 'roughly-straight-ahead' type actions. The ellipses are not data generated, they are for illustration only.

The KD-tree partitioning of the action space is used to construct the hierarchical tree data structure, with percept models, $\mathcal{N}_i(\bar{\mathbf{p}}, \Sigma_p)$, and action models, $\mathcal{N}_j(\bar{\mathbf{a}}, \Sigma_a)$ at each node. Starting at the root node, the root percept and action models are built from the complete exemplar set as in equations 5.3 and 5.4. The percept and action models are then learnt for each of the root nodes child nodes, and for these nodes child nodes and so on recursively until a leaf node is reached. No model is built at the leaf nodes as they only contain one percept vector and one action vector. The P-A hierarchy is illustrated in figure 5.7, where each node contains PDFs representing the distribution of percepts and actions. The models nearer the leaf are more specific, the models nearer the root at more general. Consider the complete set of percept models:

$$\hat{P} = \{\mathcal{N}_i^{0,0}, ..., \mathcal{N}_i^{n,h}\} \tag{5.6}$$

and action models:

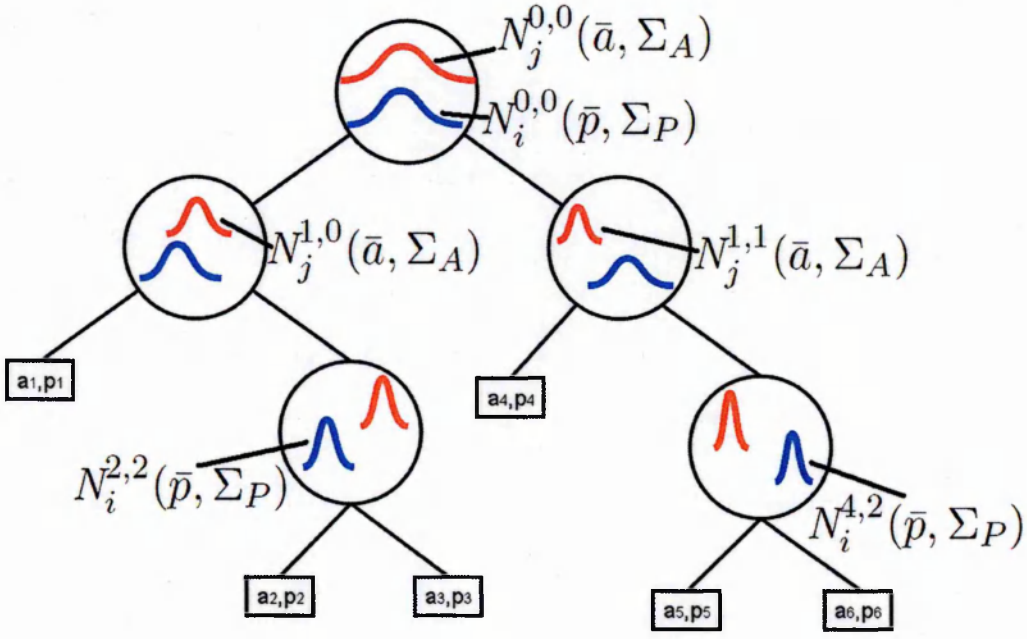


Figure 5.7: *P-A hierarchy construction*: The P-A hierarchy is a recursive tree data structure with two PDFs at each splitting node, one representing the distribution of actions vectors in the region represented by that branch, the other representing the percept distributions.

$$\hat{A} = \{\mathcal{N}_j^{0,0}, ..., \mathcal{N}_j^{n,h}\} \tag{5.7}$$

where n is the node index at level h in the hierarchy, as shown in figure 5.7. The only node at level $h = 0$ is the root, containing $\mathcal{N}_i^{0,0}$ and $\mathcal{N}_j^{0,0}$, In general, as h increases, indicating nodes further from the root, the models become more specific, until at the root node the model is a point-to-point mapping.

5.2.2 Searching the hierarchy

Given a new percept, $\mathbf{p} = [p_1, ..., p_i]$, as input, the objective is to find the most specific matching percept model, $\mathcal{N}_i^*$, from $\hat{P}$, the set of all percept models, and so identify the best action model from which to generate a response. This is achieved by recursively searching the P-A hierarchy, from the root node down, by comparing $\mathbf{p}$ to the percept

model at each node. If the matching criteria is met then the child nodes are compared, if not met, the search of that part of the hierarchy terminates. If no child nodes exist i.e. the node is a leaf, then the leaf model, a single percept vector, is selected as $\mathcal{N}_i^*$. This happens when there is an exact match for $\mathbf{p}$ in the exemplar set.

The criteria for matching percept $\mathbf{p}$ to model $\mathcal{N}_i^{n,h}$, is defined in terms of a matching function, u,

$$u_{n,h}(\mathbf{p}) = \begin{cases} 1 & \text{if } D_M(\mathbf{p}, \mathcal{N}_i^{n,h}) < \tau \\ 0 & \text{Otherwise} \end{cases} \tag{5.8}$$

Where $D_M(\mathbf{p}, \mathcal{N}_i)$ is the Mahalanobis distance as defined in equation 5.5 and τ is a threshold. An investigation of the effects of this threshold is given in section 5.3.

This search procedure is unlike a standard binary search tree, as no hard decision is made at each node. Instead, it is possible for the search to continue down a number of separate branches, providing the input matches the model representing each branch. This means that the search can terminate at more than one node in the hierarchy, providing a set of matching models or specific examples. These matches constitute a set of hypotheses of how best to react to the current percept. To select the best hypothesis, the model with the minimum Mahalanobis distance may be selected.

Having identified the best matching percept model, $\mathcal{N}_i^*$, new actions can be generated by drawing samples from the associated action model, $\mathcal{N}_j^*$. If the best match is on a leaf node, then there is an exact percept match, so the exact same action is selected.

5.2.3 Discussion of P-A hierarchy

There are three interesting, and useful properties of the P-A hierarchy. First, the output space clustering identifies which input features are important to the task and uses this to categorise the inputs. Second, the specificity (generality) of the selected action model reflects the familiarity (unfamiliarity) of the perceptual input. This property results in the generation of more exploratory actions, with a higher degree of randomness, in unfamiliar situations, and more exploitative actions, with a lower degree of randomness, in more familiar situations. This smooth, context aware transitioning between

exploitative and exploratory behaviour is a desirable property for an emergent agent. Finally the hierarchical search process is efficient as only a subset of the exemplar space is searched. These three properties, *perceptual weighting*, *generalisation* and *efficiency*, are each demonstrated and evaluated in the following section.

Figure 5.8 illustrates the intended functionality of the P-A hierarchy for a simple navigation task. The task is for the car to drive toward the circular object. The square object is a distractor object, it does not determine the response. The hierarchy is built from three exemplars, representing 'hard-left', 'hard-right', and 'straight-ahead' actions. The variance in the position of the circle and the square, within a percept model, is illustrated by their size and shade in the general models. The input percept requires a 'soft-left' class of action. The result of the search is the selection of an action model that generalises the 'hard-left' and 'straight-ahead' actions. A sample drawn from this model is likely to result in appropriate behaviour. This figure illustrates the *perceptual weighting*, *generalisation* and *efficiency* of the P-A hierarchy.

5.3 Experiments

5.3.1 Experimental setup

The problem domain used to evaluate the P-A hierarchy is a robotic platform, learning to drive towards a target object. The robotic platform is constructed from a standard Remote Control (RC) car fitted with a wireless camera. A Graphical User Interface (GUI) is used in the training stage that presents the teacher with the current image from the camera and allows the user to select appropriate actions to determine the control signals sent to the RC car, see figure 5.9. Each action starts and ends with the RC car stationary, user selected actions cause the car to move, in the direction determined by the position of the mouse click, before stopping. Twenty sequences (304 percept-action exemplars) were recorded and used to construct the exemplar set. The system is trained to drive toward a red balloon placed in the robots environment. There is also a green balloon placed in the environment that represents a distractor or background object. If the red balloon is not in the image, the teacher turns the car

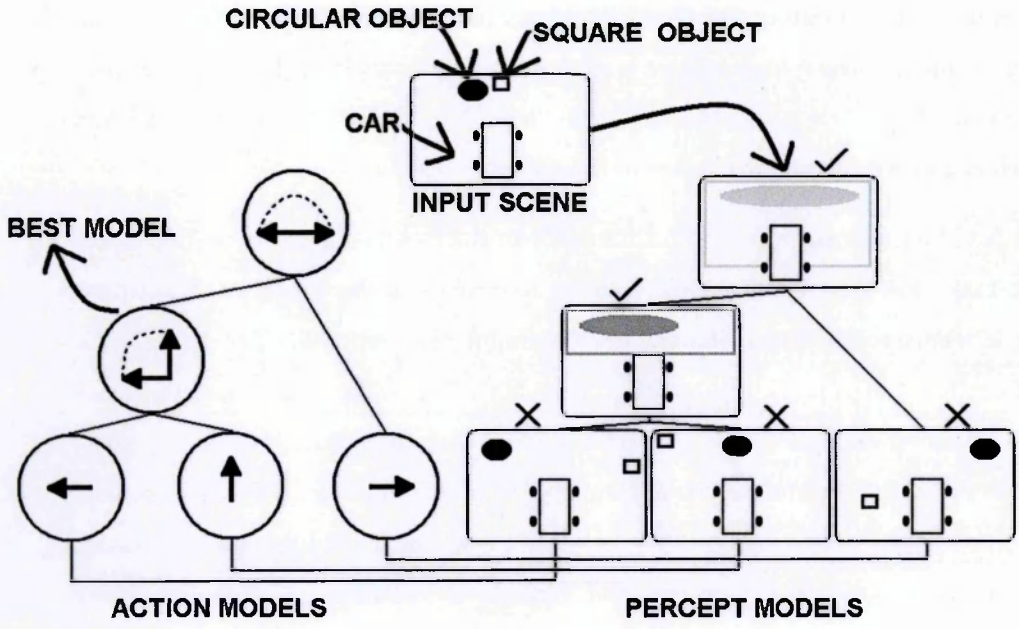


Figure 5.8: *P-A hierarchy as percept-action mapping*: Illustration of generalising and feature importance weighting capabilities of percept-action hierarchy. The car has been trained to follow the circular target object, the square object is not important to the task. The input percept is first compared to the most general percept model and as it fits the matching criteria it is compared to both child models. It fits only the left child and is subsequently compared to this models children, both of which it fails to match. Therefore the best matching model is one that generalises over two examples. The associated general action model is thus selected as the most applicable to the current situation.

until it appears.

This imitation training framework facilitates the imparting of knowledge by demonstration rather than by explanation/specification. Importantly this knowledge is embodied, grounded in the agents own percept-action representations. Thus by allowing a teacher to demonstrate appropriate behaviour through the agents control hardware, the agent is bootstrapped with a set of embodied experiences. These experiences can be used to facilitate exact mimicry, the objective then is to extend this behaviour to more general imitative behaviour.

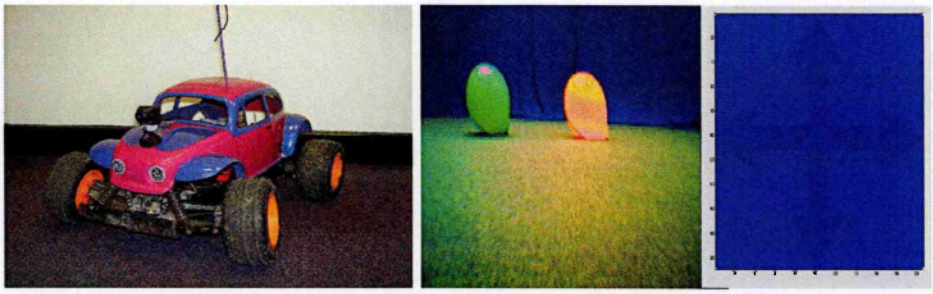


Figure 5.9: *Autonomous navigation hardware and control interface*: A standard RC car is fitted with a miniature wireless camera and is controlled by clicking on a GUI to generate the drive and turn control signals.

5.3.2 The exemplar set

The set of action vectors $A = \{\mathbf{a}^1 ... \mathbf{a}^{304}\}$, where $\mathbf{a}^n = [a_1^n, a_2^n] \in \Re^2$ are 2-dimensional vectors that represent the turn and drive parameters used to control the robot.

The percept vectors, $P = \{\mathbf{p}^1 ... \mathbf{p}^{304}\}$, where $\mathbf{p}^n = [p_1^n, ..., p_{18}^n] \in \Re^{18}$ are 18-dimensional vectors representing the state of the world. As illustrated in figure 5.10, each image is processed with a blob feature extraction algorithm [27]. The extracted blob features, that represent homogeneous regions in the image, are parametrised by six moments (describing shape) and the RGB encoding of the mean colour of the detected region. These parameters are concatenated to form a single nine element feature vector for each blob. A colour lookup table is used to identify the two balloons (red and green) and the percept vector is the ordered concatenation of the two 9-dimensional balloon blob features:

$$\begin{aligned}\mathbf{p}^n = [&M00^n_{red}, ..., M05^n_{red}, R_{red}, G_{red}, B_{red}, ..., \\ &M00^n_{green}, ..., M05^n_{green}, R_{green}, G_{green}, B_{green}]\end{aligned}$$

If either of the balloons is not present in the image the corresponding elements in the percept vector are set to zero. The P-A hierarchy is built using these exemplars.

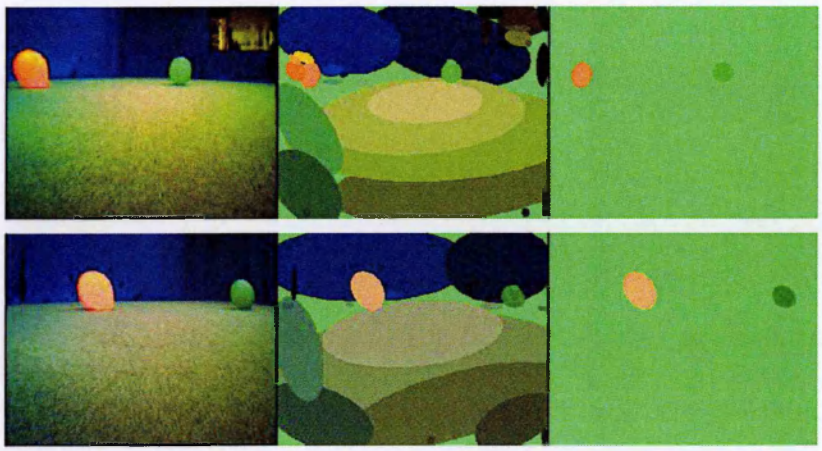


Figure 5.10: *Percept representation for autonomous navigation:* Left images are received from wireless camera. The middle images represent the output from the blob detection algorithm. The right images represent the percept representation.

5.3.3 Evaluating perceptual weighting of output clustering

One of the important properties of the P-A hierarchy is the facility to identify, through output space clustering, the relative importance of perceptual parameters to the task exemplified by the exemplar set. To evaluate this property, the discriminative percept classifiers, learnt through output space clustering, are tested with a set of unseen data.

The complete set of action exemplars used to train the P-A hierarchy are shown in figure 5.11. The exemplars are represented both as points in a 2D space and as vectors with magnitude representing the drive parameter and direction representing the turn parameter. Also shown in figure 5.11 are three subsets of actions. The subsets broadly represent turn left, go straight ahead and turn right action modes. Associated to each of these action subsets is a percept subset, from which a percept classifier is built, based on the Gaussian PDF and the Mahalanobis distance function. The response (Mahalanobis distance) of each of these three classifiers to a set of six test images is shown in figure 5.12, along with the six test images.

The results presented in figure 5.12 demonstrate the perceptual weighting property of output space clustering. The classifier scores for test images 1 and 2, that show the target to the left, are considerably lower than all other test images for the turn left

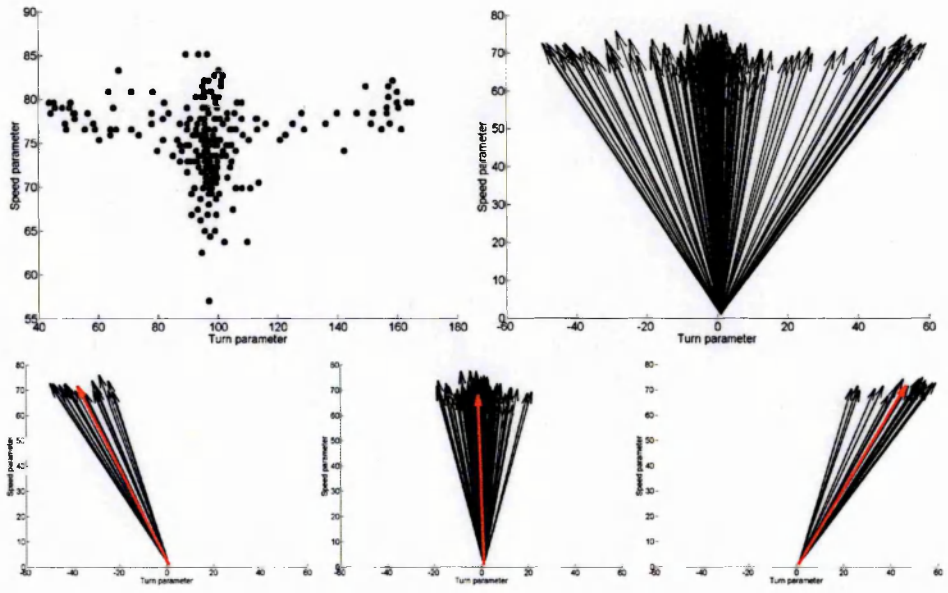
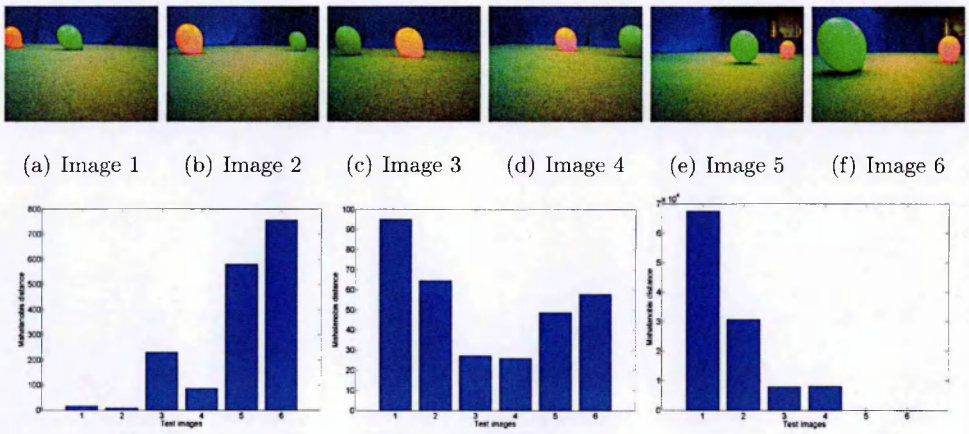


Figure 5.11: *Action exemplars*: Top left image is plots of all action vectors as points in 2D space. The top right image represents the actions as 2D vectors, with magnitude representing the drive parameter and direction representing the turn parameter. The bottom row represent three subsets of actions, identified by the output space clustering. The subsets broadly represent turn left, go straight ahead and turn right action modes (from left to right).

class. Furthermore, the test scores for images 1 and 2 are also considerably higher in the other two classes. This indicates that for the turn left class of actions, the percept classifier is strongly biased towards representations with the target object to the left. This is further illustrated by noting the classifier scores for test images 1 and 5, on the turn left and turn right classes. Despite the similarity of the location of the distractor object (green balloon) in the two images, the scores for the two classifiers are radically different, reflecting a radical difference in the position of the target object. Similar results and comparisons are also evident for images 3 and 4 where the target is consistently to the center and the distractor has high positional variance, and for test images 5 and 6 likewise.



(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4 (e) Image 5 (f) Image 6

Figure 5.12: *Perceptual weighting through output space clustering*: The top row shows six test images selected to evaluate the perceptual weighting properties of the P-A hierarchy. The bottom row shows the Mahalanobis distances between each test image and the Gaussian PDFs representing the subset of percepts associated to the three actions subsets illustrated in figure 5.11.

5.3.4 Evaluating generalisation and efficiency

To evaluate the generalising capabilities of the P-A hierarchy, unseen test exemplars, $\mathbf{p}^n$, for $[n = 1...(N_{test})]$ are applied to the P-A hierarchy to generate outputs, $\mathbf{a}^*$. The prediction error is the L_2 norm distance between $\mathbf{a}^*$ and $\mathbf{a}^n$, the expected output, as in equation 5.9.

$$Prediction\ error = f(\mathbf{p}^n) = \|\mathbf{a}^* - \mathbf{a}^n\| \tag{5.9}$$

The Mahalanobis distance threshold, τ, used in the percept matching function u (see equation 5.8), is a determining factor in the specificity/generality of the models returned by the percept matching process, and therefore has impact on the generalising capabilities of the P-A hierarchy. The test exemplars are tested over a range of Mahalanobis threshold values.

The result of this test, for a randomly selected training (70% of exemplars) and test (30% of exemplars) set is shown by the dashed black line in figure 5.13. It was noted that running this test with different training/test partitioning of the exemplar set results

in different plots. To characterise the general shape of these plots, the test is run 50 times, each time with a different random partitioning of the exemplar set into 70% training and 30% test. The mean plot over these 50 tests is shown by the blue line in figure 5.13. The average prediction error for the NN-PA approach is also computed over each of the 50 tests and is 23.6 (constant over all τ as this parameter is not used in the NN-PA model).

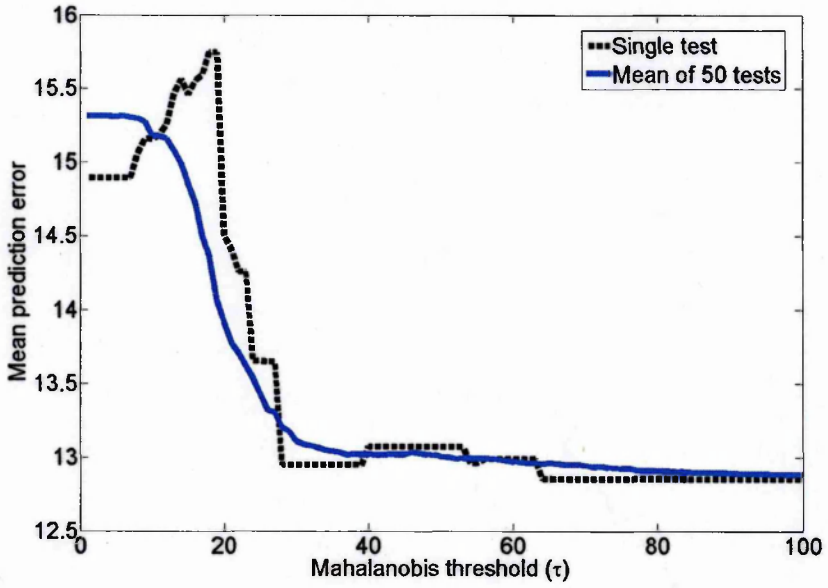


Figure 5.13: *Prediction error as a function of Mahalanobis threshold, τ*: The mean prediction error on the test exemplars for values of *tau* ranging from 1 to 100, are plotted for a single train/test data partition. Also shown is the mean test result over 50 tests with randomly selected train/test data partitions.

This result demonstrates the ability of the P-A hierarchy to generate appropriate outputs given unseen inputs, therefore generalising over the exemplar set. The P-A hierarchy also achieves considerably lower prediction errors on unseen data than the NN-PA approach. The result shows that, as the percept matching criteria is relaxed, with $10 < \tau < 30$, the mean prediction error falls rapidly, this is the result of matching more specific models. It is also found that for values of τ above 30, there is little improvement in prediction error, at least up to $\tau = 500$.

For the results in figure 5.14 the same set (the training set) is used both to build the

P-A hierarchy and evaluate the recall rate and efficiency trade off. The recall rate indicates the ratio of exact matches found to total number of exemplars, where 100% recall implies all exemplars are correctly matched, see equation 5.10. The efficiency of the hierarchy reflects the proportion of the search spaced that is searched, see equation 5.11. The search space includes all exemplars as well as all the general models created by the hierarchy.

$$Recall\ rate = \frac{\#\ exact\ matches}{total\ \#\ of\ exemplars} * 100 \tag{5.10}$$

$$Efficiency = (1 - \frac{\#\ nodes\ searched}{total\ \#\ nodes}) * 100 \tag{5.11}$$

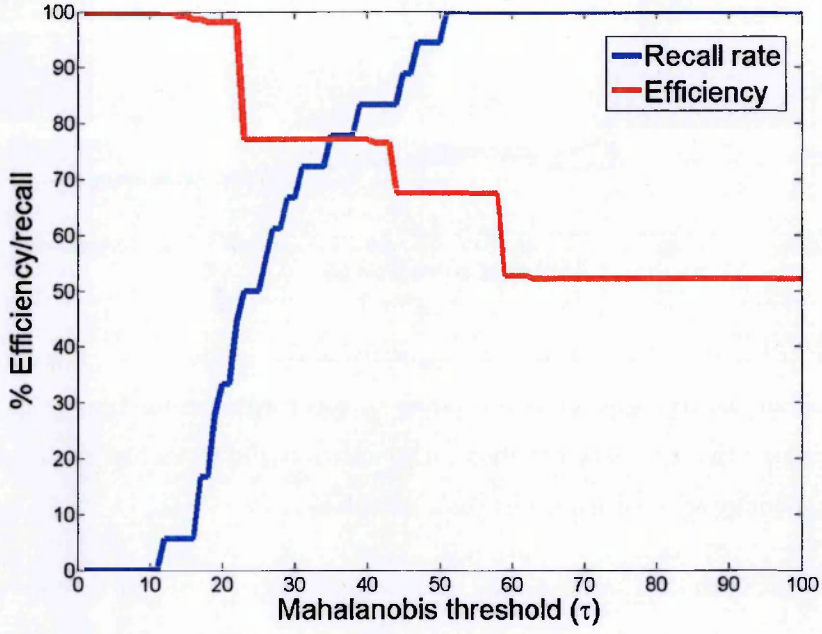


Figure 5.14: *Recall and efficiency as a function of* τ: The P-A hierarchy is built and tested using all exemplars and the recall rate is the ratio (number of exact percept matches)/(total number of exemplars)*100. Efficiency is defined as (1-((number of nodes searched)/(total number of nodes)))*100.

Figure 5.14 shows that, as τ increases in the range $10 < \tau < 30$, the ability of the hierarchy to recall exact exemplar matches increases from zero to 70%. In the same

range, the efficiency of the hierarchy falls from nearly 100% to approximately 75%. In the range $30 < \tau < 50$, the efficiency falls by approximately 10% and the recall rate increases by 30% to achieve 100% recall for values of τ above 50%. It can be seen that efficiency falls in large steps at three points in the curve, indicating that large areas of the search space are included when τ is raised above the three coresponding thresholds.

The results presented in figures 5.13 and 5.14 not only evaluate the generalising, recall and efficiency properties of the P-A hierarchy, but also provide a means of selecting a value for τ. Given that little improvement is made in the prediction error for values of $\tau > 30$, and that this value provides a reasonable trade off between recall and efficiency, τ is set to be 30, for the autonomous navigation experiments presented in section 5.3.5.

5.3.5 Autonomous navigation evaluation

These experiments are carried out by placing the robot within an environment with the two balloons. The experiments are designed to test whether the system has learnt a useful mapping from the percept space to the action space, $\Re^i \rightarrow \Re^j$.

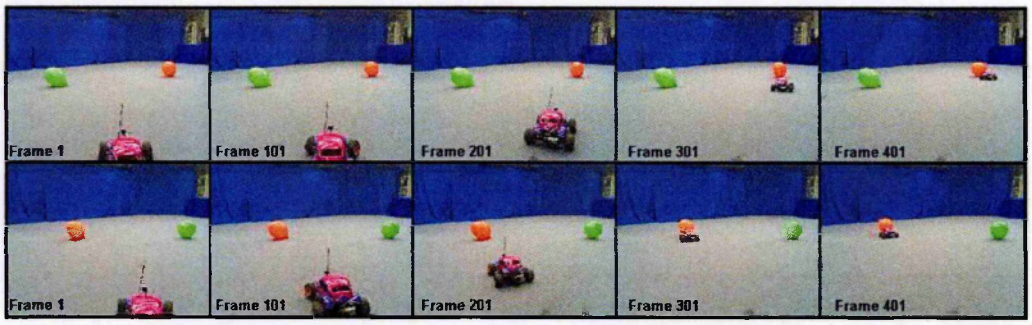


Figure 5.15: *Frames of video demonstrating autonomous navigation*: Every 100th frame from two video sequences showing the agent performing the task of driving towards the target object. The top/bottom row show the agent driving to the right/left in response to the position of the target.

Figure 5.15 shows frames from a video sequence of the car reproducing the trained behaviour. The car and balloons are placed at many different positions relative to each other and the car consistently locates and drives towards the target object. Occasionally the system over steers in a certain direction but usually compensates with the next

action. If the red object is not in the scene, the car turns until it appears, sometimes this continues indefinitely, with the car constantly turning and not finding the balloon, or with the car encountering an obstacle such as a wall. In these situations, the test is restarted. Errors due to vision system occur rarely as the blob detection and colour lookup table approaches work well in this relatively constrained environment.

Figure 5.16 illustrates the autonomous navigation system. The KD-tree decomposition of the action vector space is presented at the top in figure 5.16(a). Below that in figures 5.16(b), 5.16(c), 5.16(d) and 5.16(e), four steps from two sequence of the car driving toward the target balloon are shown. It can be seen that the actions sampled from the hierarchy (right image for each step) reflect the position of the red balloon in the perceptual representation. This indicates that the percept matching process is being based on the location of the red rather than the green balloon, a result of clustering in the action space.

5.4 Conclusion and Discussion

This chapter introduces a novel exemplar learning framework that employs hierarchical output space clustering to achieve search efficiency and input and output space generalisation as well as a mechanism for identifying the important variance and invariance in the input space. The *exemplar hierarchy* provides, in a single structure, a mechanism for classifying unseen inputs and generating appropriate outputs. The organisation of the hierarchy is driven by the structure of the output space and supports Granlund's postulate quoted in the introduction to this thesis.

Applied to the percept-action domain, this approach provides a realisation of an embodied and embedded agent that organises its perceptual space and hence its cognitive process based on interactions with its environment. The exemplar hierarchy also provides a mechanism that allows the embodied agent to move on a continuum between exploratory and exploitative modes of behaviour, depending on the familiarity of the situation.

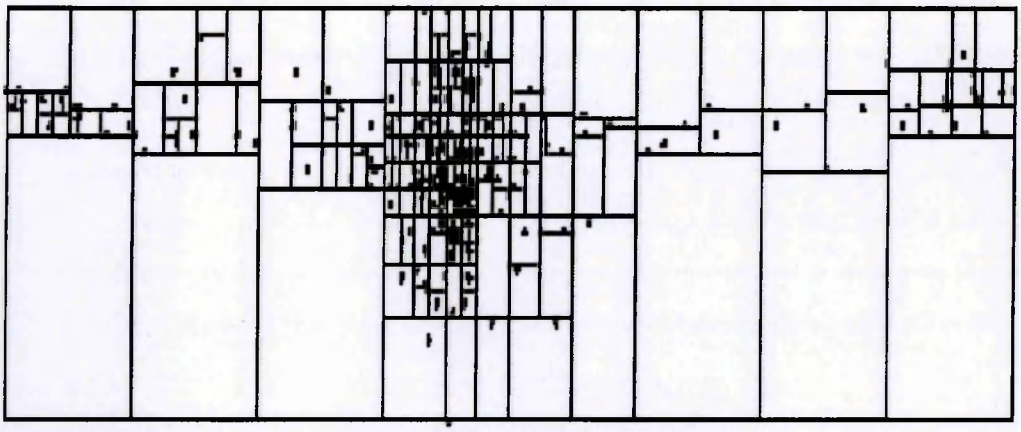
The results demonstrate that the system is capable of learning to respond appropri-

ately without the need for task specific knowledge, other than knowledge imparted by demonstration. Furthermore, it is shown that the hierarchy allows for generalisation of experiences that allow it to perform in unseen scenarios.

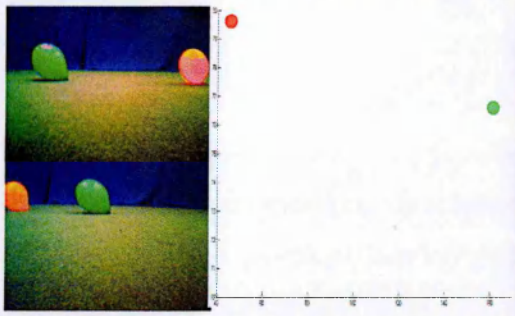
The integrated perceptual categorisation and motor control mechanism developed demonstrates the self organizing principle typical of emergent systems. By constructing a perceptual system in a bottom up manner, the categories of perception are physically grounded in the sensory-motor experience of the agent and so there is no symbol grounding problem. This is achieved by exploiting the fact that structure can be discovered in the action space due to its low dimensional representation and the similarity of related points. This structure provides, in a sense, the meaning/semantics implicitly linked to the formed perceptual categories.

Although it is a feature of emergent systems, the random exploratory behaviour exhibited as a result of the Gaussian action model sampling process is not suitable for all application domains. This degree of randomness is due to the probabilistic nature of the mappings. There is no mechanism for interpolating between input-output mappings, other than increasing the generality of the matched model and therefore increasing the randomness of the generated actions. The next chapter extends this work but replaces the point-to-point and generalised point-to-point mappings with mappings that explicitly model the relationship between inputs and outputs. The mappings employed are similar to the regression based mappings introduced in chapter 3.

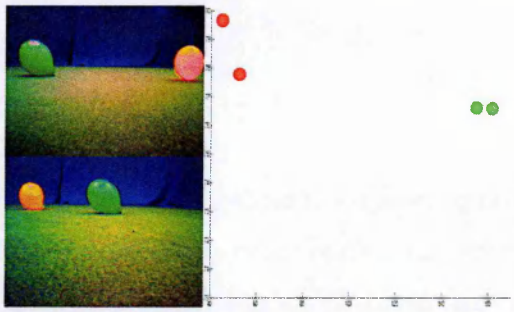
The relative simplicity of the problem domains selected indicates some limitations of the approach. The strict ordering of perceptual features in the percept representation relies on a robust feature correspondence matching algorithm. In order to model the covariance of features from a group of percepts such a matching is always required. One proposal for future work arising from this issue is the development of a feature matching algorithm that utilises the statistics captured within the exemplar hierarchy to efficiently match image features and hence maintain a consistent perceptual representation. Using the statistics used to build the percept classifiers, within a detection framework, could be one solution to the problem.



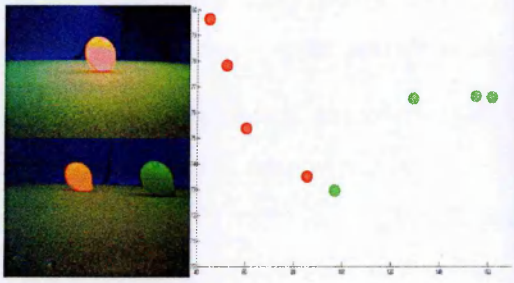
(a) The KD-Tree decomposition of the action space.



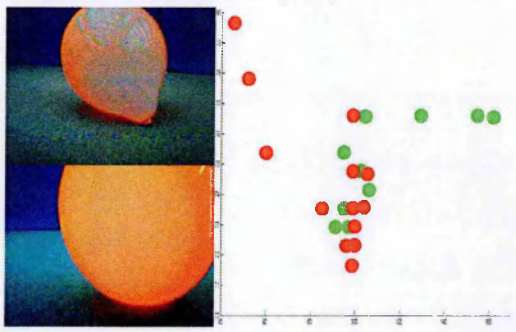
(b) First step of two sequences (top and bottom), with target to left and with target to right.



(c) Second step of sequences. Generated actions reflect change of position of target blob.



(d) Forth step of sequences. Target is centered in image.



(e) Final step in sequence. The goal state is achieved by a number of go-straight-ahead actions.

Figure 5.16: *Demonstration of RC car experiment on two sequences*: Left figures: input images; right figures: generated actions. The blob feature classifiers interpret the perceptual space in order to activate the correct action models for generating a response.

Chapter 6

Affordance Mining: Forming Perception Through Action

In the previous chapter, output space clustering is used to organise the input space, by forming perceptual groups associated to classes of actions. A simple Gaussian assumption was employed to weight the relevance of features of the input space to the discovered classes of action. This approach required a vector representation of the percept space, with consistent ordering of the perceptual features i.e. the order of the features representing the red and green balloons was fixed. While the approach demonstrates the organising principle of output space clustering, the strict ordering of the features of the percept vector leads to a requirement for an engineered perceptual representation and so limits the scope of application of the approach. This chapter focuses on removing this constraint.

This chapter introduces a method for discovering the visual entities that are important to a vision system given a specific problem (e.g. a robotics tasks). Mappings are learnt from these emergent perceptual entities, onto the agents action space. This is achieved by first applying unsupervised learning in the problem output space (e.g. the agent's actions). The structure discovered in the output space is then used to

organise the input space (e.g. the agent's perceptual representation), in order to form meaningful input representations. This organisation process is achieved by finding strong associations between modes of the output space and configurations of features in the input space. Association rule data mining algorithms are employed to efficiently find these associations. Local feature configurations that are strongly associated to a particular 'type' of action (and not all other action types) are considered as likely to be relevant in eliciting that action type. By learning mappings from these relevant feature configurations onto the action space, the system is able to generate real-time, continuous responses to novel visual stimuli.

This chapter adopts many of the approaches developed throughout this thesis. The imitation training framework in which domain specific embodied knowledge is imparted by allowing a teacher to demonstrate appropriate behaviour through the agents control hardware is adopted from chapter 5. So to is the use of unsupervised learning to discover structure in the output space in order to construct exploitable representations in the input space. However, rather than the generalised point-to-point mappings used in chapter 5 the mappings adopted in this chapter are similar to the linear regression functions introduced for displacement estimation in chapter 3.

In chapter 5 it was established that low dimensional action spaces are better suited to unsupervised learning than high dimensional percept spaces, allowing for structure to be discovered in the action space and imposed on the percept space to form meaningful categories. As discussed, this approach relates to the embodied mind theory, that is based on the premise that the nature of the mind is determined by the embodiment of the cognitive agent [48] [13]. Related to this is *affordance* theory, that states that the world is perceived not only in terms of object shapes and spatial relationships but also in terms of object possibilities for action [29]. The work presented in this chapter demonstrates an embodied approach to constructing an affordance based representation of the world. This is achieved by directly coupling action generation models to each discovered visual entity, such that observing the entity directly activates the associated action model.

Data mining algorithms are useful for efficiently identifying correlations in large sym-

bolic datasets. These methods have begun to be applied to vision tasks such as: identifying features which have high probability of lying on previously unseen instances of an object class [62], mining compound spatio-temporal features for scale invariant action recognition [30], and finding near duplicate images within a database of photographs [17]. These methods benefit from both the scalability and efficiency of the data mining methods. This work employs data mining algorithms to the novel domain of percept-action association mining. The mechanism of mining frequent and distinctive feature configurations employed here is most similar to that of Quack et al. [62], however, here the discovered configurations are used directly in an action generation process, rather than as a pre-processing step for identifying useful features for other classification techniques. Furthermore, whilst in [62] supervision is required to label the classes of objects that are learnt, in this work classes of actions are obtained by an unsupervised learning approach.

The exemplar hierarchy, introduced in chapter 5, was shown to be capable of learning to respond appropriately with no explicit definition of desired behaviour. This chapter addresses the following limitations of the approach proposed in chapter 5:

1. *Partially engineered representation of visual information*: Although the approach presented in chapter 5 was shown to identify the important variance and invariance in the input space, and to weight the relative importance of features to the percept matching criteria, this was demonstrated using a percept representation that required a certain degree of engineering or hard-coding. The input image space was represented as an ordered concatenation of two 9-dimensional blob feature vectors, where the two blobs represented were selected based on colour, which was chosen by the engineer to match the task specific target and distractor objects. The work presented in this chapter builds on the mechanisms presented in chapter 5 to develop a system that forms a task specific percept representation directly from image data with no hard-coded knowledge about the objects of interest.

2. *Probabilistic action generation*: The actions generated by the system presented in chapter 5 were drawn from probabilistic generative models. No mechanism was

proposed for determining the relationship between the percept vectors belonging to a class of percepts and the action vectors belonging to the associated action model. This chapter draws on the ideas from chapter 3 to learn linear functions that encapsulate the relationships implicit in the training data. Using regression to learn these functional approximations allows for new actions to be generated based on an interpolation between training data examples, thus avoiding the introduction of randomness in the response generation process.

3. *Discrete action space*: In the imitation training framework employed in chapter 5, the actions are discrete (in time) impulses, with the vehicle coming to a stop after each action. Both the training procedure and the response mechanisms developed in this chapter are developed to allow the system to generate continuous signals, in real time, in response to dynamic events in the environment.

4. *Search space dependent on amount of experience*: The exemplar hierarchy relied on a potentially large and costly search procedure, thus limiting the system response time. The size of the search space being determined by the number and distribution of the exemplars representing the systems experience. In the approach presented in this chapter, inputs are matched to rules discovered within the exemplars, and so the search complexity is dependent on the (controllable) number of rules discovered, rather than the number of exemplars.

Figure 6.1 represents the approach proposed in this chapter within the PAC framework that was introduced in chapter 1 and developed in chapters 3, 4 and 5. The figure illustrates that the discovery of structure in the output space is used in a data mining process to form perceptual entities that are used to interpret new inputs. Action type specific linear mappings are learnt and attached to each discovered perceptual entity. The response of the linear mappings attached to the perceptual entities matched in the input are combined to generate a system response.

The proposed approach is demonstrated on an autonomous navigation task. The resulting visual servo control strategy requires minimal calibration and no explicit definition of behaviours but instead is learnt from training data. Conventional visual servo control

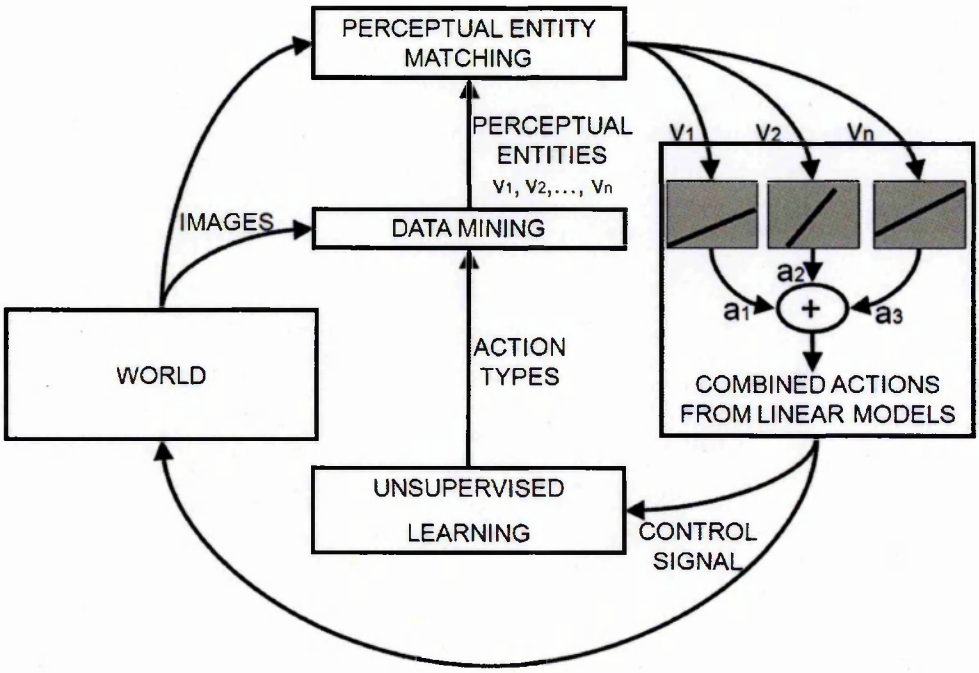


Figure 6.1: *Unsupervised learning in output space and percep-action mining for input-output mappings*: This figure represents the approach proposed in this chapter within the PAC framework introduced in chapter 1 and developed in chapters 3, 4 and 5. Unsupervised learning is applied to the problem domain's output space to discover structure in the output space. The structure discovered is used in a percept-action mining process to discover task specific visual entities. Matching these entities in input images activates linear input-to-output mappings. The response of the mappings attached to the matched perceptual entities are combined to generate a system response.

systems employ the use of a priori knowledge in the form of camera calibration models and 3D models of the geometry of objects of interest. This conventional top-down approach is consistent with the cognitivist paradigm. To avoid the problems encountered by cognitivist systems the approach presented here concentrates on learning low level behavioural models from embodied experiences.

The rest of this chapter is organised as follows: Section 6.1 provides details of the robotic platform used to collect the training data and test the system and then presents an analysis of the collected training data. Section 6.2 gives details of a supervised

method of learning a percept-action mapping that provides a baseline performance for comparison with the unsupervised approach developed in the rest of the chapter. Section 6.3 describes the central mechanism of action space clustering and how this identifies classes of actions and percept groupings. Section 6.4 presents a complete overview of the proposed system, identifying the key processing stages involved, which are presented in detail in sections 6.5 and 6.6. Section 6.5.1 details the approach used to encode visual information as feature configurations and sections 6.5.2 and 6.5.3 detail the method of finding associations between classes of actions and these feature configurations. Section 6.6 details how mappings are learnt between associated percept and action data and how these mappings are exploited to generate responses to novel image data. Section 6.7 presents the experimental evaluation of the system and section 6.8 contains a discussion and conclusions.

6.1 Embodied Percept-Action Experiences

The approach developed in this chapter is based around learning from embodied Percept-Action (P-A) exemplars. The system aims to learn how to respond to new inputs in a way that imitates the behaviour implicit in a training set of these exemplars. This section details the robotic platform used both to collect the training data and to test the behaviour imitation capabilities. An analysis of the collected training data is also presented, that identifies the type of relationships that should be discovered from the data.

6.1.1 Embodied agent hardware

The robotic platform developed is a relatively inexpensive platform for the investigation of embodied artificial cognitive agents. Based on a standard Remote Control (RC) model car fitted with a wireless camera, the system allows a teacher to demonstrate the desired driving behaviour by viewing the images from the camera on a PC monitor and using a standard computer game steering wheel and foot pedal controller to navigate the car. This training approach is an adaptation of the mechanism presented in chapter

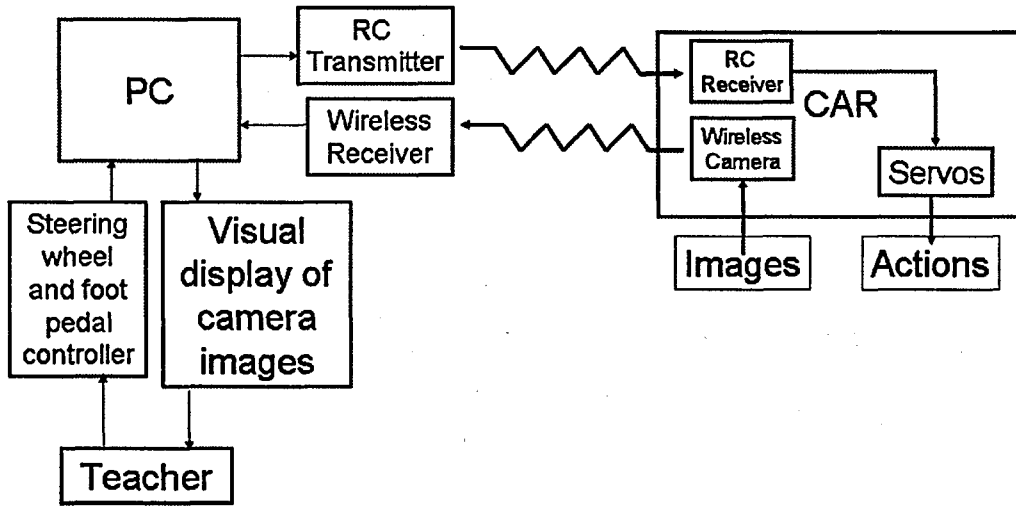


Figure 6.2: *Mobile agent hardware*: A standard Remote Control (RC) car is fitted with a miniature wireless camera. The images are transmitted to a standard PC via a wireless USB receiver. During training these images are recorded by the PC and displayed to the teacher. The teacher then responds to the images via a standard steering wheel game controller, these signals are recorded along with the images and are transmitted via a USB-to-PWM converter and RC transmitter to the RC car.

5 that allows for a more intuitive and natural control of the RC car that generates continuous control signals. It is important that the teacher is presented with the images from the camera rather than viewing the RC car directly. This ensures that the teacher is responding to the same inputs the agent will have to respond to.

An overview of the mobile agent hardware is illustrated in figure 6.2. The images from the wireless camera are transmitted to the wireless receiver that is connected via USB to a PC. During the training stage, these images are displayed on the monitor and stored. Also during the training stage a standard steering wheel and foot pedal game controller are connected to the PC and the control signals generated by the user interaction are transmitted to the car and stored along with the associated image. A device called an SC-8000P that converts USB to PWM is used to interface the PC to a standard RC transmitter, allowing control signals to be sent from the PC to the car.

The training process involves the teacher driving the agent in order to follow a lead ve-

hicle. This collects a sequence of pairs of images and control parameters - the embodied training data - as illustrated in figure 6.3.

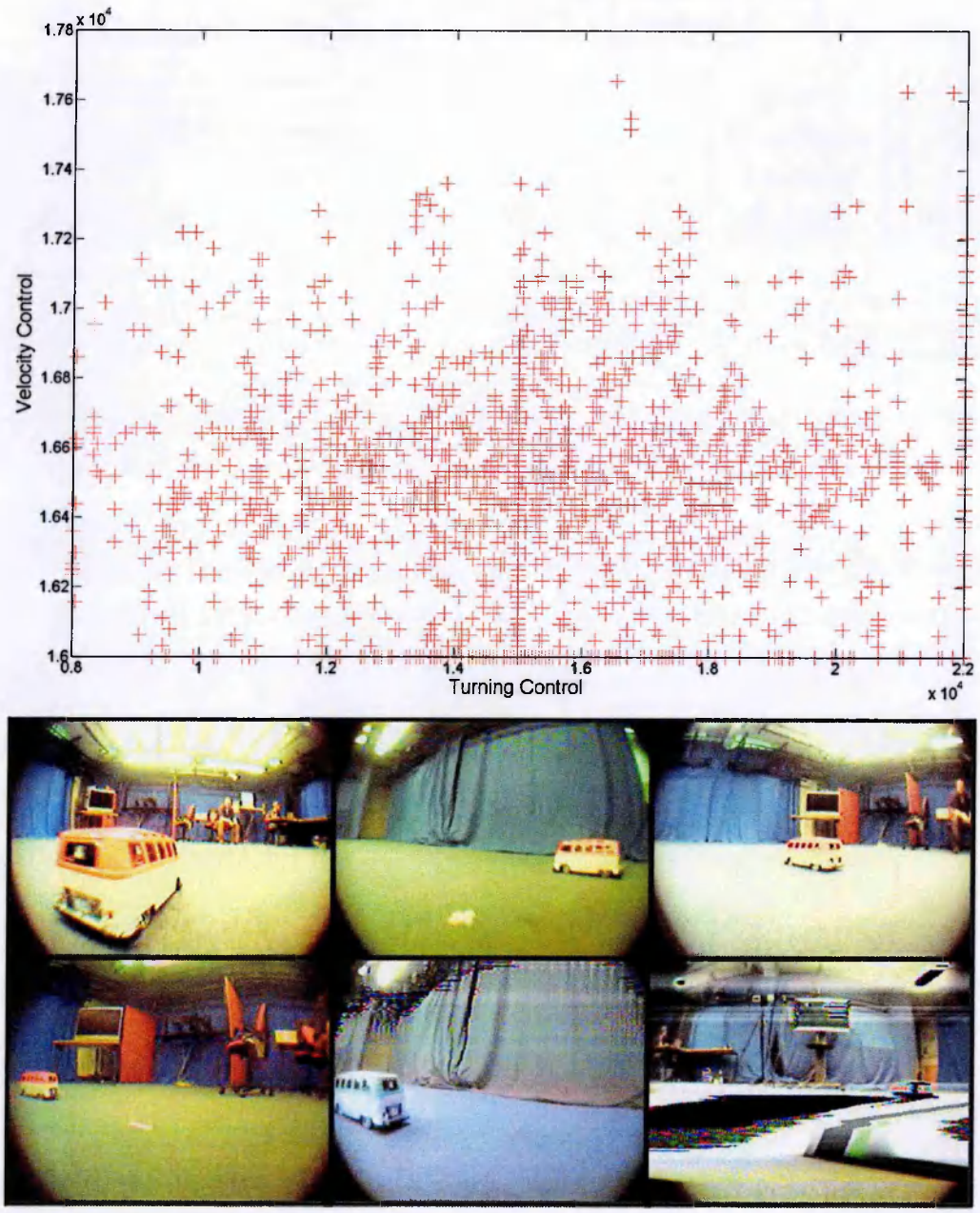


Figure 6.3: *Embodied training data*: The action data (top) and examples of the image data (bottom) are shown. Occasionally the image data is corrupted due to interference from the wireless camera (as shown bottom left and bottom center).

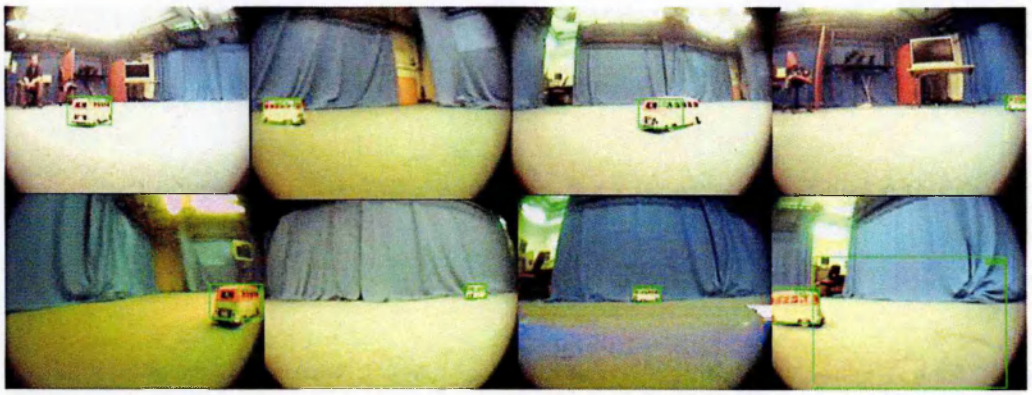


Figure 6.4: *Lead vehicle tracking:* The waldboost detector is run on every frame of a training sequence to provide pose estimates of the lead vehicle. The pose parameters are *x-position*, *y-position*, *height* and *width*.

6.1.2 Analysis of embodied training data

The action data is the log of the actions performed by a teacher, who is responding to the motion of the lead vehicle in order to follow it. Therefore, the desired behaviour should be implicitly represented by the relationships between the pose of the lead vehicle and the associated control signals. To test this, the lead vehicle is tracked over the entire sequence, producing a sequence of pose parameters. A version of the LK-SMAT tracker presented in chapter 4 was extended [20] to affine deformations and was initially used in this tracking application, with good results. However in the experiments presented here a waldboost object detector [70] is used instead, as detection recovers well from occlusions and image noise. The result of running the walboost detector on a number of the training images is illustrated in figure 6.4.

The set of control and pose parameters obtained from the tracker/detector thus provides a new data set for the vehicle following behavioural model. This data set is constructed with a significant level of supervision, as the detector is trained on hand labeled examples of the lead vehicle. Occasionally the detector produces multiple detections (see bottom right image in figure 6.4), this is usually resolved automatically by selecting the detection with the highest confidence, but in some cases user interaction is required to ensure the correct detection is selected. This supervised data set

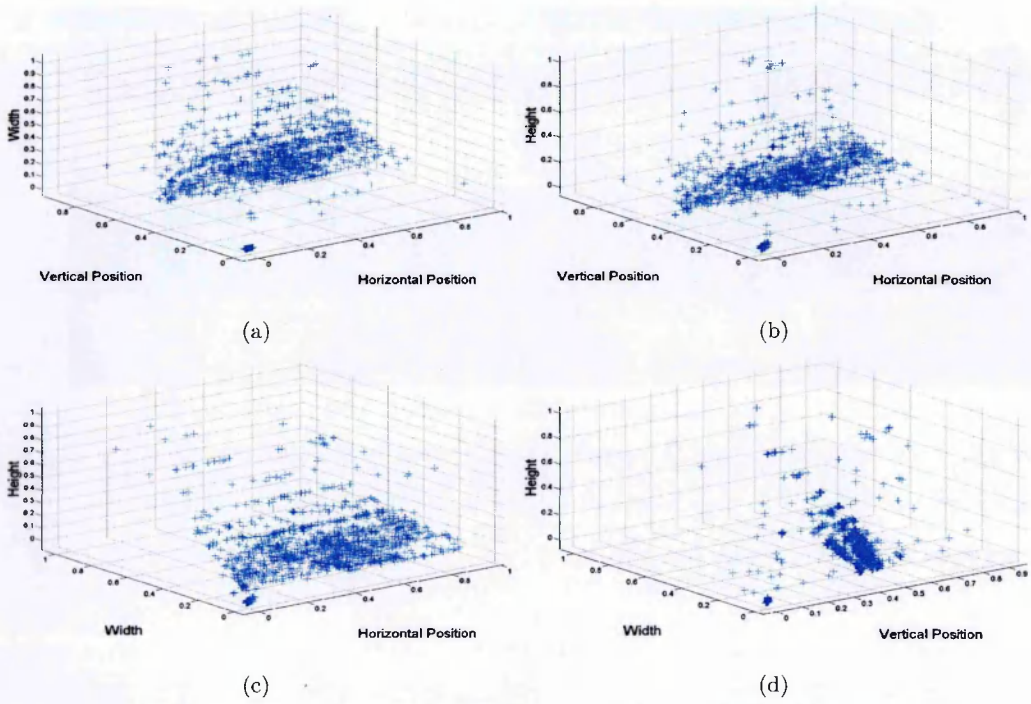


Figure 6.5: *Supervised vehicle following behaviour dataset*: The four dimensional pose data obtained by tracking the lead vehicle is represented in four 3D plots. The cluster of points around the origin are the result of non-detections - resulting in a null pose vector. Note the high correlation between the height and width parameters, evident in figures 6.5(c) and 6.5(d).

is useful as it provides a way to analyse the relationships implicitly represented in the training data. These are the relationships that should be discovered by the proposed, unsupervised approach.

Figure 6.5 presents the data obtained as described above. In figures 6.5(a), 6.5(b), 6.5(c) and 6.5(d) the four dimensional pose data is represented in four 3D plots. Observe the cluster of points, in each plot 6.5(a), 6.5(b), 6.5(c), 6.5(d), around the origin. These vectors are the result of non-detections - resulting in a null pose vector. The high correlation between the height and width parameters is evident in figures 6.5(c), 6.5(d).

Figure 6.6 shows a plot of horizontal position of the lead vehicle against the turn control action parameter. This figure illustrates the strong correlation between the pose of the lead vehicle and the teachers actions.

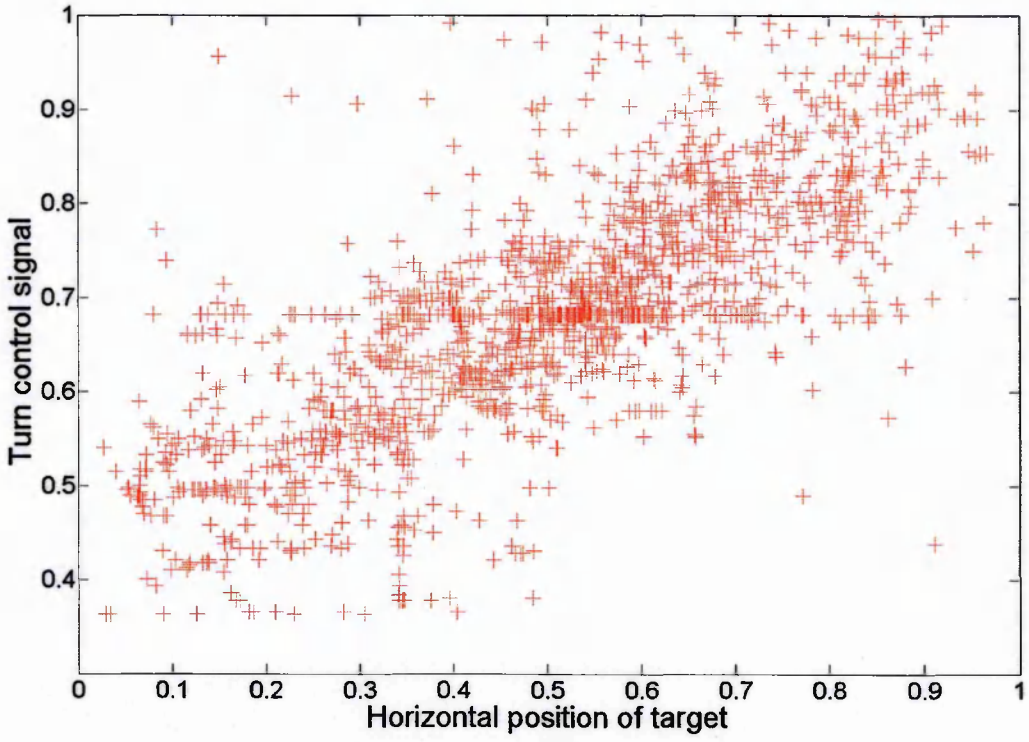


Figure 6.6: *Selected pose parameter to control relationship*: The approximately linear relationship between the horizontal position of the lead vehicle and the turn control action parameter.

6.2 Supervised Percept-Action Mapping

As figure 6.6 shows, there is a strong correlation between the lead vehicle pose and the associated actions. It should therefore be possible to predict the actions from the output of the lead vehicle detector. Whilst such a system relies on a significant level of supervision, both in training the detector and in disambiguating multiple detections, it is described here as it provides a baseline performance for a system expected to predict actions from percepts on this dataset.

The supervised percept-action mapping approach maps from the 4-dimensional pose parameters, $\mathbf{p}$, obtained from the lead vehicle detector, onto the 2-dimensional action parameters, $\mathbf{a}$, $\Re^4 \rightarrow \Re^2$. The mapping is achieved using a single linear regression model, $\mathbf{H}$. A bias term is included in the linear model. So an action, $\mathbf{a}$ is computed from a pose vector, $\mathbf{p}$, as in equation 6.1.

$$\mathbf{a} = \mathbf{H}\mathbf{p} + b \tag{6.1}$$

The training set for learning $\mathbf{H}$ is the supervised dataset illustrated in figures 6.3 and 6.5, i.e. $\{\mathbf{a}_i, \mathbf{p}_i\}$ pairs, $(i \in [1, N])$. All the pose vectors, $\mathbf{p}$, and the associated action vectors, $\mathbf{a}$, are stacked into the training matrices, $\mathbf{P}$ and $\mathbf{A}$ respectively. To learn the bias for the linear model an additional column of 1s is added to the end of $\mathbf{P}$, giving: $\mathbf{P}' = (\mathbf{P}, [1])$, where [1] denotes a column vector of N rows. Using least squares, $\mathbf{H}$ can now be obtained as follows:

$$\mathbf{H} = \mathbf{A}\mathbf{P}'^{+} = \mathbf{A}\mathbf{P}'^{T}(\mathbf{P}'\mathbf{P}'^{T})^{-1} \tag{6.2}$$

Where $\mathbf{P}'^{+}$ is the pseudo inverse of $\mathbf{P}'$ and $\mathbf{P}'^{T}$ is the transpose of $\mathbf{P}'$.

The result of applying this mapping to an unseen test dataset of pose data (obtained from the trained lead vehicle detector) is shown in figure 6.7. It can be seen that the signal generated by the approach approximately follows the expected signal. The spikes in the generated signal correspond to false detections (not removed in the test set), usually these detections are of excessively large scale.

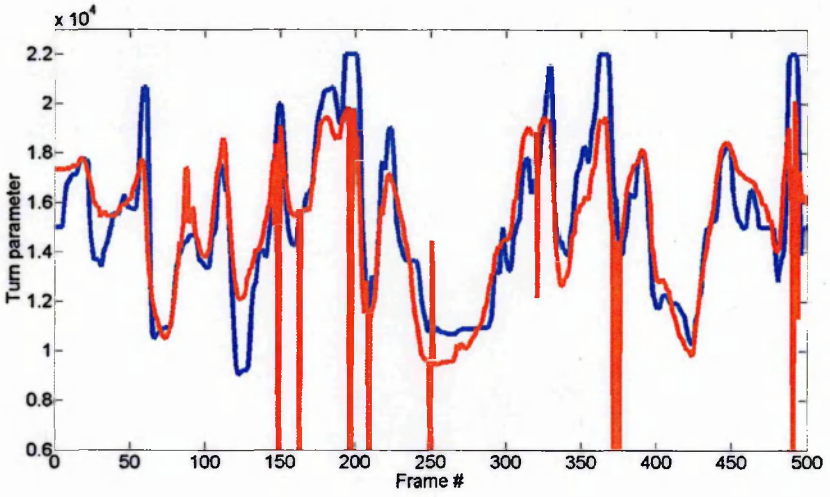


Figure 6.7: *Supervised percept-action mapping*: The generated 'turn-control' action signals (red) are shown for the supervised method, along with the expected action signal (blue).

The rest of this chapter focuses on removing the supervision inherent in this approach. The system should be able to identify the object of interest (the lead vehicle) and to identify and exploit the relationship between the state of this object and the desired actions, without any supervision.

6.3 Action Space Clustering

As was discussed in the previous chapter, unsupervised learning techniques are often applied to percept spaces (e.g. image or feature space), but are prone to yielding ambiguous or erroneous results. This is often due to assumptions about suitable distance metrics used to cluster the data. For an embodied agent (e.g. all natural cognitive systems and the system proposed in this work), percept data is never obtained in isolation - it is always coupled to action data. This coupling is exploited in this work by clustering coupled percept-action exemplars, in the action space. This results in the formation of meaningful classes of action or 'action-types', as well as meaningful perceptual groups. Besides this organising property, action space clustering also pro-

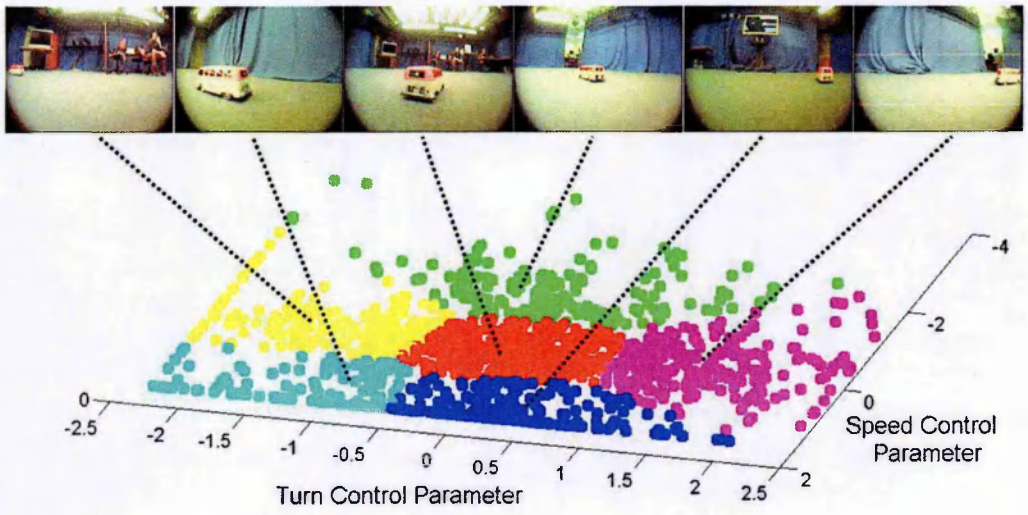


Figure 6.8: *Action clustering*: Action clusters are formed along with sets of associated images.

vides a means of symbolically representing the action space. Attaching symbols to the identified action-types allows symbolic data mining techniques to be applied to the continuous action space.

Figure 6.8 illustrates the result of performing this action space clustering on the collected training data. The action data is clustered - using k-means clustering - into $k = 6$ clusters, and examples of the associated images are shown. In order to obtain invariance to displacement, scale and rotation, the action data is whitened prior to clustering. The data is translated (by the mean sample value), scaled (each dimension by the associated eigen values of the sample covariance matrix) and rotated such that the features have zero mean, unit variance and the data axis coincide with the eigenvectors of the sample covariance matrix.

6.4 Affordance Mining Process Overview

An overview of the proposed approach is illustrated in figure 6.9. First an exemplar set, E, of training data of the form $E = \{(\mathbf{p}^1, \mathbf{a}^1), ..., (\mathbf{p}^N, \mathbf{a}^N)\}$, where $\{\mathbf{p}^1 ... \mathbf{p}^N\}$ is the set of images, and $\{\mathbf{a}^1 ... \mathbf{a}^N\}$ is the set of action vectors, $\mathbf{a}^n = [a_1^n, a_2^n] \in \Re^2$, is collected

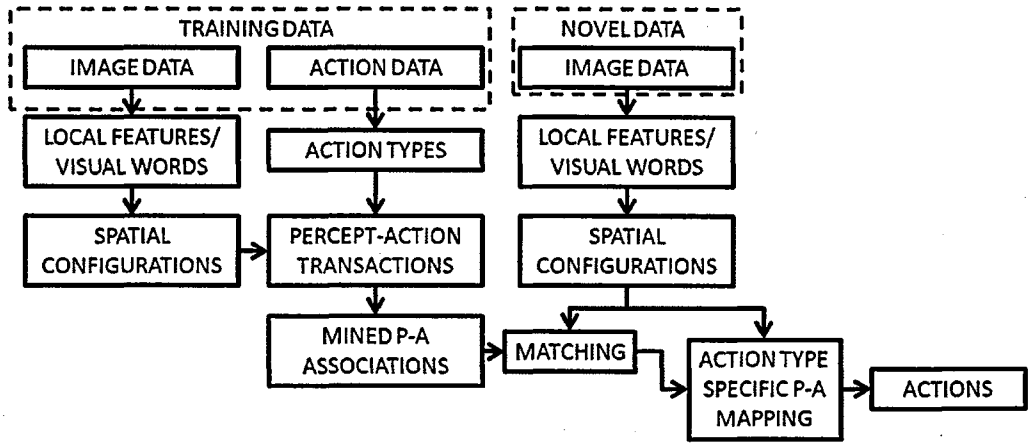


Figure 6.9: *System overview*: Coupled percept and action data are represented as Percept-Action (P-A) transaction vectors by concatenating visual codeword configuration vectors and action-type labels. Data mining is then used to discover P-A associations that identify feature configurations that are associated to a particular action-type. Matching these association rules in training images then provides data for learning P-A mappings for each association rule, that map from feature configurations to actions. Matching the association rules in novel images then activates the associated P-A mappings, thus providing a mechanism for generating appropriate responses to novel image data.

(details of this data gathering process are given above). Symbolic representations of both the actions, and percepts, are then formed. For the action data, k-means is applied directly to action vectors, resulting in k action-types, as detailed above. For image data, a visual codebook of local image SIFT features is built using k-means clustering, where the k cluster centers make up the codewords. Spatial relationships between features are represented by encoding local feature configurations, as described in section 6.5.1. The visual information in each image is thus represented as a set of codeword configurations.

Links between the symbolic percept and action spaces are then obtained by performing data mining on a combined Percept-Action (P-A) representation, named P-A transactions. Each transaction represents an action-type coupled to a codeword configuration,

where one item in each transaction represents the action-type, and the remaining items represent a visual codeword feature configuration, as detailed in section 6.5.2. The data mining algorithm then processes these transactions to produce P-A association rules.

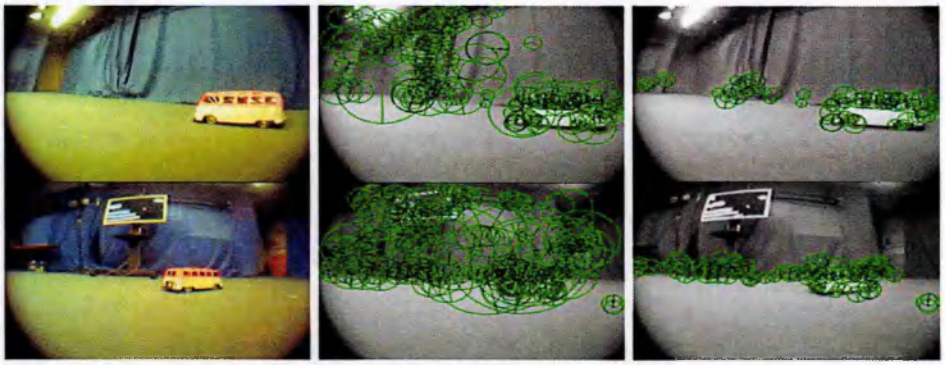
The continuous training data, and the mined symbolic association rules are then used to learn action-type specific P-A mappings, as in section 6.6.1. These mappings map from the continuous (un-quantised) pose of image features that are associated to an action-type, onto the continuous action vectors belonging to that action-type. These mappings constitute affordances for the mined perceptual entities.

Still referring to figure 6.9, when presented with novel image data, the system constructs the visual codeword configurations as before. These configurations are matched to the mined association rules and the P-A mappings associated to the rules are applied to the features that form the matching configurations, in order to generate a response. This process of generating responses to novel image data is detailed in section 6.6.2.

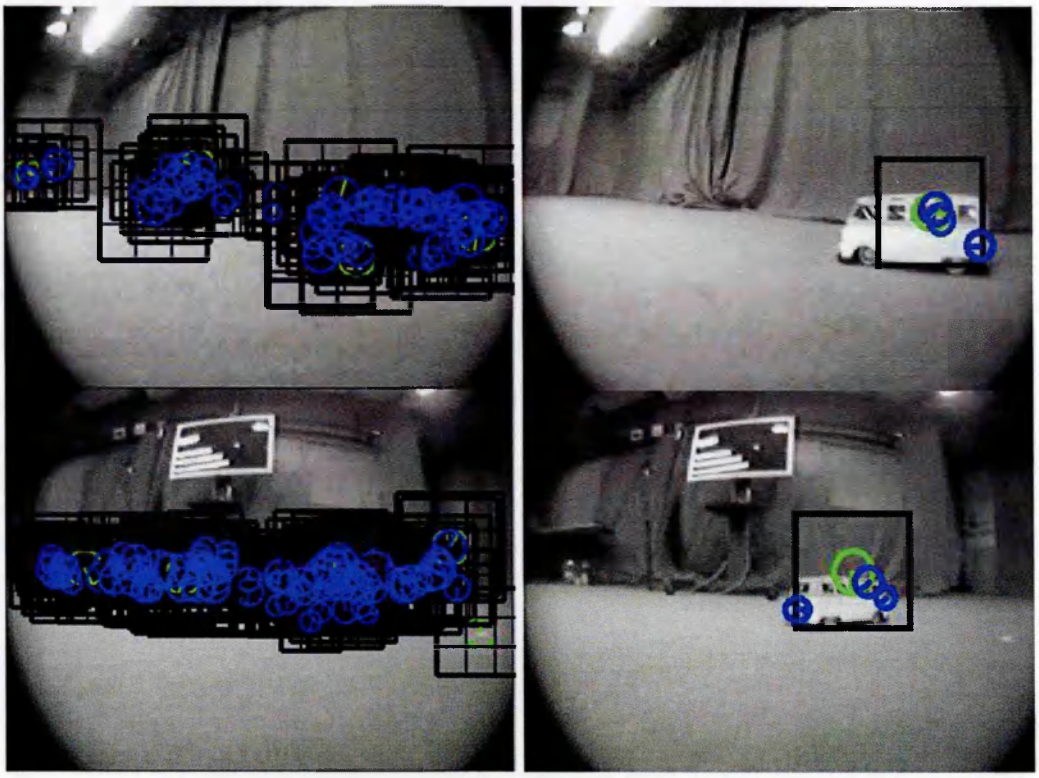
6.5 Mining Percept-Action Associations

The proposed vision system is based on local feature descriptors. A Difference of Gaussian (DoG) detector is used to extract regions and the SIFT descriptor [50] is used to describe the regions. A prior is placed on the scale and location of the SIFT features used in the later stages of the process. This results in a filtering of the set of SIFT descriptors extracted from each image. Figures 6.10(b) and 6.10(c) illustrate this filtering stage. As the lead vehicle will always remain on the ground plain, and as features on the lead vehicle will have a limited scale in the images, features are rejected that appear too near the top of an image or have overly large scales.

The 128-dimensional SIFT feature descriptors are clustered to form a visual word vocabulary, using k-means clustering. Additionally, the scale and orientation of the features are clustered to form 'scale words' and 'orientation words'. Meaning that each SIFT feature can be described using three discrete labels - descriptor, scale and orientation words - and the continuous horizontal and vertical position. For clustering the descriptor, $k = 50$, for scale and orientation, $k = 5$.



(a) Input image. (b) Sift descriptors. (c) Sift filtering.



(d) Feature configurations. (e) Mined configuration.

Figure 6.10: *P-A mining process*: Four stages of the feature mining process are illustrated. Sift descriptors are extracted from the input images. These are then filtered to remove features near the top of the image or that have overly large scales. Feature configurations are then assembled and those configurations that are associated to particular action-type are then discovered through data mining.

6.5.1 Feature configurations

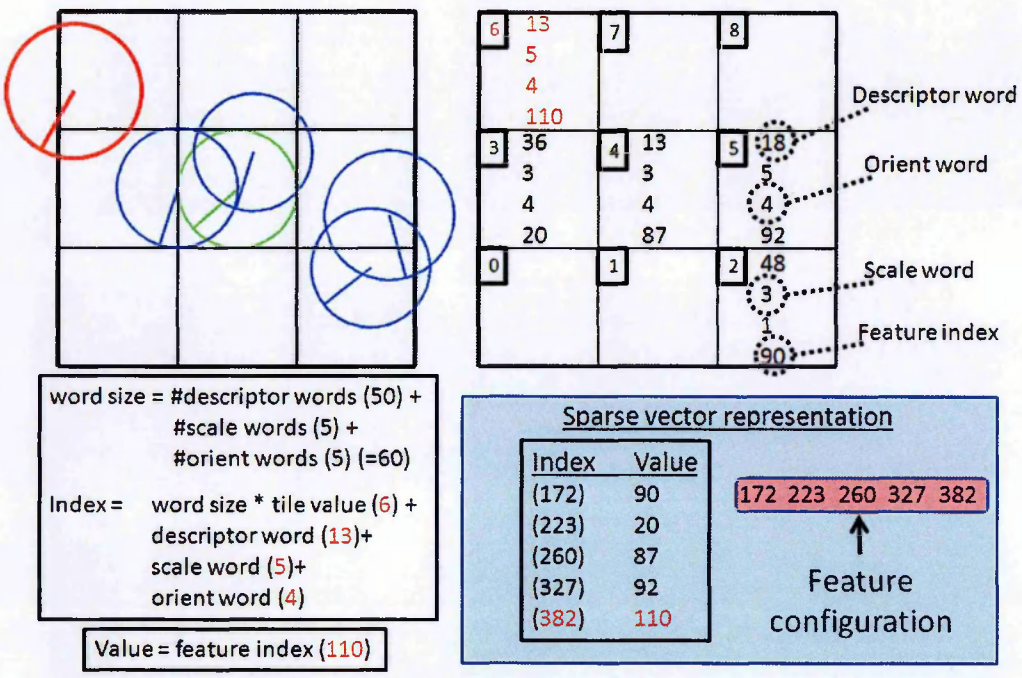


Figure 6.11: *Encoding configurations*: This figure illustrates how a configuration of features is encoded in a sparse vector representation, and how this sparse vector representation is used to build the feature configuration vectors used by the mining algorithm. The top left of the figure shows a configuration of features found around the central (green) feature. The top right of the figure illustrates how the feature configuration is represented as a configuration of visual codewords at quantised relative locations, scales and orientations. The bottom left part of the figure details how a particular feature (marked in red in the top left) is encoded in the sparse vector representation. The bottom right of the figure shows the sparse vector representation of the configuration. Also shown is the feature configuration vector that forms the percept part of the transaction vectors used in the data mining. The values of the non-zero indices of the sparse vectors are the feature indices that identify the feature in the image, these are used when mapping from feature pose to action parameters. Note that the center feature (green) is not represented.

Figure 6.11 illustrates the method used to encode the spatial configuration of the extracted SIFT features. A similar scheme was introduced in [62]. For every feature in an image (after filtering) a 3-by-3 grid is placed on the image, centered on the feature, and scaled proportionally to the feature scale. Any neighbouring features that fall into a tile of the grid are encoded as part of that feature configuration, the encoding reflects which tile the feature is in i.e. it's spatial relation, and the visual, scale and orientation words representing the feature. A sparse vector representation is employed for which the non-zero indices encode the configuration and the values store the feature index in the image, so that the continuous feature pose may be recalled for the P-A mappings. The feature configuration vector contains the indices of the non-zero elements of the sparse vector, and is used to represent the visual information in the data mining process.

Examples of feature configurations for two of the training images are shown in figure 6.12. As can be seen, some of the feature configurations lie on or partially on the target vehicle, whilst many lie on the background. The full set of configurations for an image (as illustrated in figure 6.10(d)) will contain considerable redundancy, where each local pairwise spatial relationship will be encoded a number of times within multiple feature configurations.

6.5.2 Percept-Action transaction database

Association rule mining is the process of finding association rules in a database $D = \{t_1, t_2, ..., t_m\}$ of transactions, where each transaction is a set of items, and I is the set of all items[1]. An association rule is an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

A Percept-Action (P-A) transaction represents a feature configuration coupled to the associated action-type. The action-type being the cluster label associated to the image from which the feature configuration is extracted.

[1]The terminology *transactions* and *items* comes from the data mining literature, reflecting the subjects origins in market basket analysis applications

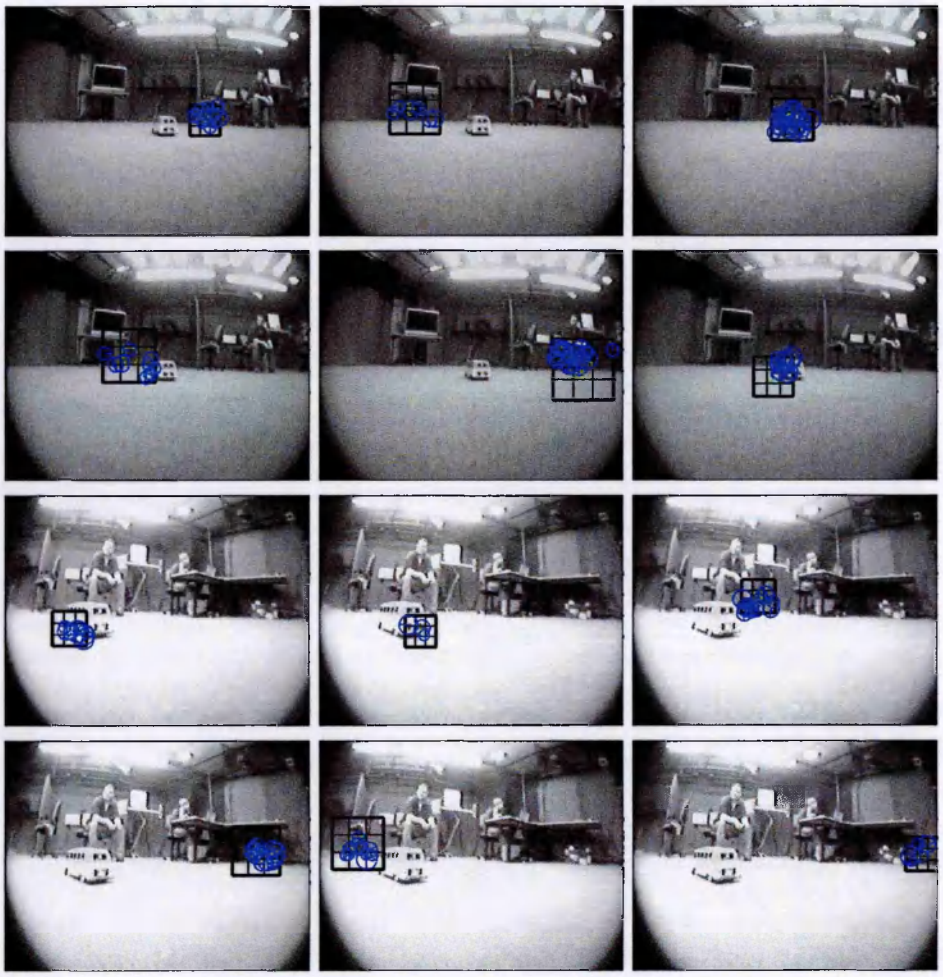


Figure 6.12: *Feature configurations*: Six examples of feature configurations for two frames are shown. Some of the configurations contain features on the target, some contain features only from the background.

The set of items is $I = \{\alpha_1, ..., \alpha_k, R_1, ...R_l\}$, where $\{\alpha_1, ..., \alpha_k\}$ are the $k = 6$ action-type items and $\{R_1, ...R_l\}$ are the $l = 540$ (9 tiles, 50 visual, 5 orientation and 5 scale words) unique spatial relationships that form the feature configurations. Each transaction vector is the concatenation of the action-type item with the items from the feature configuration vector, as illustrated in figure 6.13. Therefore each transaction contains a subset of I with one item always drawn from $\{\alpha_1, ..., \alpha_k\}$.

The transaction database $D = \{t_1, t_2, ..., t_m\}$ is assembled, as in figure 6.13, by collecting

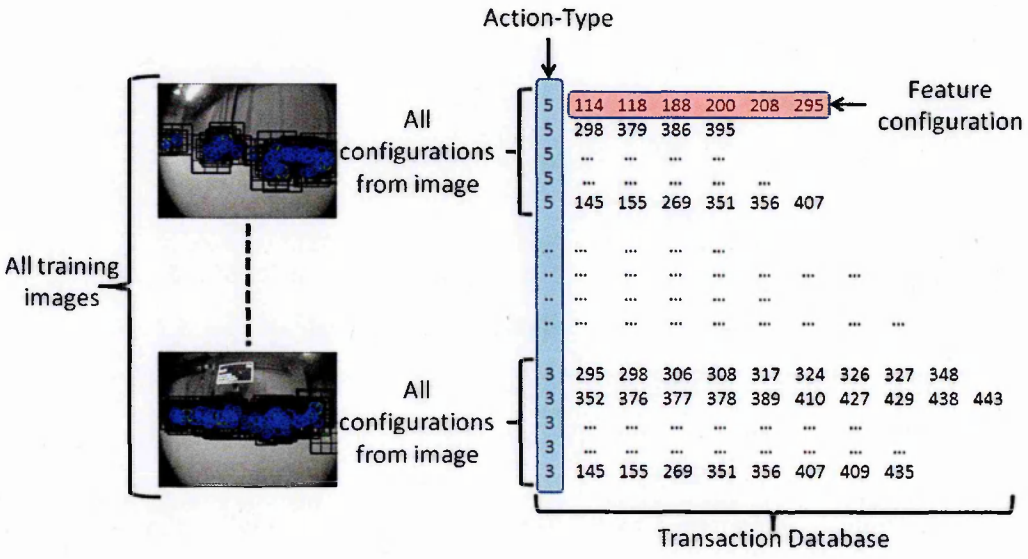


Figure 6.13: *Transaction database*: Each transaction is the concatenation of an action-type label (obtained by k-means clustering the action parameters) with a feature configuration (the indices of the non-zero elements of the sparse vector representation). The transaction database is the collection of all transactions from all training images.

together all P-A transactions drawn from all training data, $E = \{(\mathbf{p}^1, \mathbf{a}^1), ..., (\mathbf{p}^N, \mathbf{a}^N)\}$. In the experiments carried out in section 6.7, the total number of transactions in the database, $m = 88810$. This database is then processed using the Apriori [8] data mining algorithm, in order to find frequent and discriminative feature configurations for each action-type.

6.5.3 Mining P-A association rules

In association rule mining, rules are selected from the set of all possible rules based on constraints on measures of significance and interest. These constraints are thresholds on itemset *support* and rule *confidence*. The support, $supp(X)$, of an itemset X is defined as the proportion of transactions in the database which contain X. The confidence, $conf(X \Rightarrow Y)$ of a rule is defined:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{6.3}$$

The Apriori algorithm [3] employed here exploits the anti-monotonicity of the support threshold constraint - that a subset of a frequent itemset must also be a frequent itemset - to efficiently mine association rules. This work uses an efficient existing implementation of the Apriori algorithm [8].

Association rule mining is employed to mine the P-A transaction database, in order to discover feature configurations that frequently co-occur with a particular action-type, and not all other action-types. The algorithm finds subsets of items from the transaction vectors that are frequent and discriminative to a given action-type. The Apriori algorithm is ran once for each action-type, where it searches for rules including that action-type, and treats all other action-types as negative examples.

For the experiments carried out here, the support threshold $T_{Supp} = 0.02$ and confidence threshold $T_{Conf} = 99$ are used for all action-types and are selected by experimentation. These values are chosen as they provide an appropriate size set of rules to allow for real time rule matching in novel images (as detailed below in section 6.6.2). Between 400 and 500 rules are found for each action-type. The rules contain between 3 and 10 items (including the action-type item). An example of such a rule would be $\{slow\text{-}left \rightarrow 114, 188, 295\}$, meaning that a particular configuration of three features commonly occur and are associated with actions of the type 'slow-left'.

For the mining, the feature configurations are represented using the indices of the non-negative elements of the sparse vector representation, as illustrated in figures 6.11 and 6.13. However, when matching configurations found in an image to association rules, the sparse vector itself is used, rather than the non-negative indices. The sparse dot product is used to efficiently match rules to configurations found in an image. Examples of the mined association rules for each action-type are illustrated in figure 6.14.

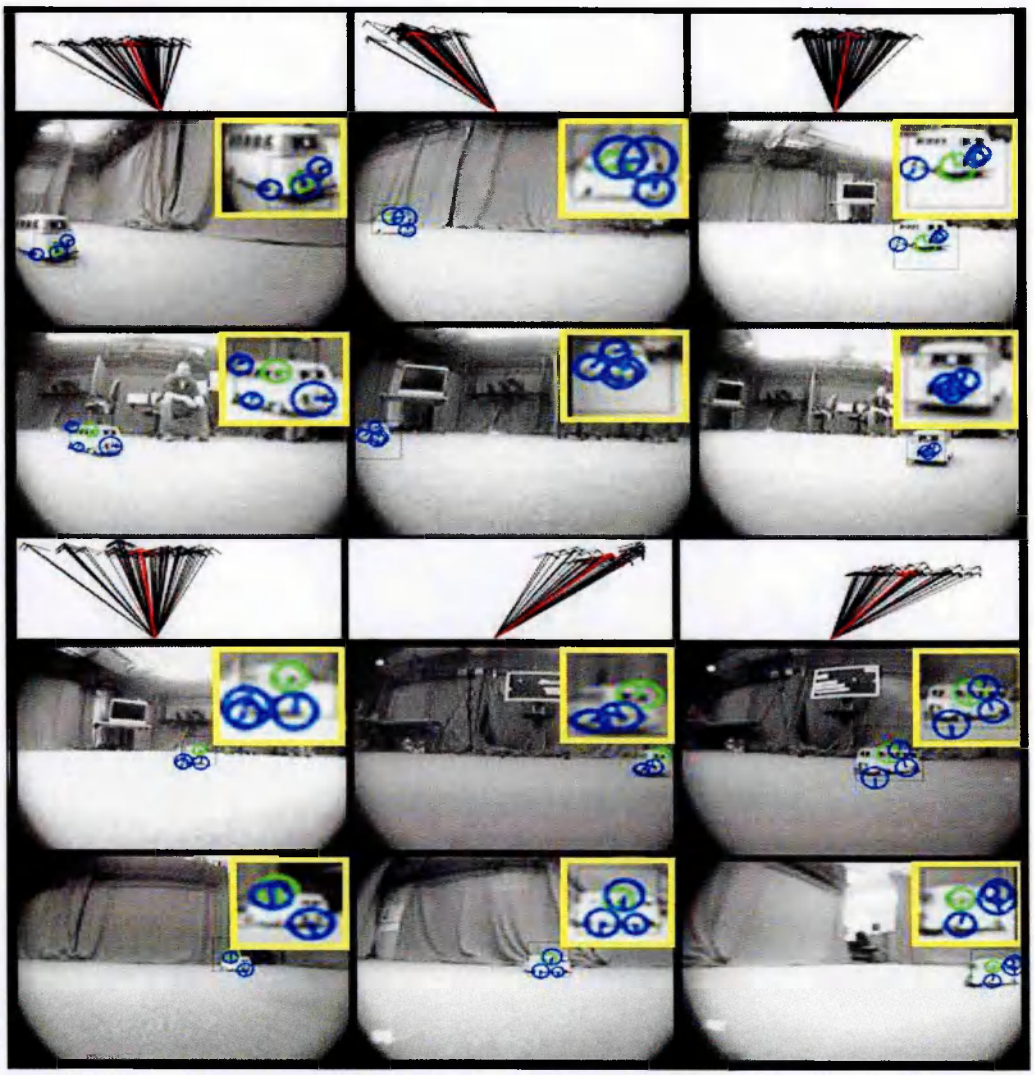


Figure 6.14: *Association rules*: Training vectors for six action-types (from left to right on top row: 'slow-left', 'fast-left', 'slow-straight', and on fourth row: 'fast-straight', 'fast-right', 'slow-right') are shown along with examples of associated configuration rules mined for each type. In general, if the lead vehicle is to the left/center/right, then the associated action is left/center/right. However sometimes the pose of the lead vehicle, rather than the position is used to associate to the action-type (e.g. far right on fifth row, and middle on bottom row).

6.6 Affordance Based Representation

This section details how the proposed system builds, in an unsupervised manner, an affordance based representation of the world, and how this representation is used to generate responses to novel percept data. This is achieved by attaching learnt mappings to each mined association rule. These mappings map from the pose (horizontal and vertical position, scale and orientation) of the features in the rules onto actions. Linear regression is used to learn linear mappings from pose space to action space. First the mapping learning process is detailed, then the mechanism developed to utilise the mappings is described.

6.6.1 Learning action-type specific P-A mappings

A linear percept-to-action (P-A) mapping, $\mathbf{H}_{P-A}$, is learnt for each association rule (mined configuration). $\mathbf{H}_{P-A}$ maps from $(C * 4)$-dimensional feature pose space, to 2-dimensional action space, $\Re^{C*4} \rightarrow \Re^2$, where C is the number of features that make up the rule. A bias term is included in the linear model. So an action, $\mathbf{a}$ is computed from a $(C * 4)$-dimensional pose vector, $\hat{\mathbf{p}}$, as in equation 6.4.

$$\mathbf{a} = \mathbf{H}_{P-A}\hat{\mathbf{p}} + b \tag{6.4}$$

In order to learn each $\mathbf{H}_{P-A}$, N training examples of $\{\mathbf{a}_i, \hat{\mathbf{p}}_i\}$ pairs, $(i \in [1, N])$ are required. The training set for each $\mathbf{H}_{P-A}$ is obtained by matching rules to configurations found in the training images. Whenever a configuration found in a training image is matched to a rule, the pose parameters of the features that make up that configuration form a new pose vector $\hat{\mathbf{p}}$. The value of the non-negative elements of the sparse vector provide the index to the matched configurations constituent features.

For each rule, all the matched configuration pose vectors, $\hat{\mathbf{p}}$, and the associated action vectors, $\mathbf{a}$, are stacked into the training matrices, $\mathbf{P}$ and $\mathbf{A}$ respectively. Least squares is then used to obtain $\mathbf{H}_{P-A}$, as shown previously.

6.6.2 Responding to novel data

A new input image is processed to generate a set of visual codeword feature configurations as detailed above. Configurations are then compared to all the mined action-type specific configurations (rules). Matching a configuration to a mined rule is achieved by computing the dot product of the two sparse vector representations. If the number of non-zero elements in the dot product is equal to the number of non-zero elements in the sparse vector representation of the association rule, then the rule is matched. If a match is found then an action prediction is made as in equation 6.4 using the $\mathbf{H}_{P-A}$ associated to the matched rule. Once all found configurations have been compared to all rules, the output action is computed as the median of all action predictions.

To speed up the generation of actions, only configurations within a search range of the previous target location are compared to the rules. The search range is proportional to median grid size of the configurations matched in the previous frame, and is centered at the median position of the previously matched configurations.

6.7 Evaluation

The two objectives of the work presented in this chapter - to discover the visual entities important to the task and to generate appropriate responses to novel data - are evaluated. This is achieved by using the supervised vehicle following dataset introduced in section 6.1.2. To recap, this dataset is obtained by learning (in a supervised manner) a detector for the lead vehicle. The detector is a Waldboost detector [70] trained on hand labeled examples - sufficient examples are used in training to provide a detector that achieves very high accuracy on the test dataset. The detected position of the lead vehicle is then used to evaluate how well the mined configurations relate to the lead vehicle. Additionally, the detected pose data is used as input to a supervised method for action generation, detailed in section 6.2, to compare to the proposed unsupervised approach.

Figure 6.14 shows examples of mined configurations that lie on the object of interest, the lead vehicle. Indeed the majority of mined configurations do lie on the lead vehicle,

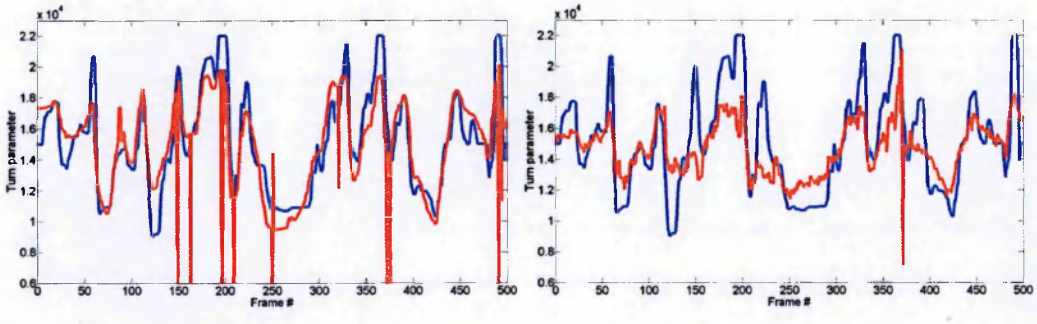
Table 6.1: Hit/miss ratio for mined configurations lying on the lead vehicle.

Action class	slow-left	fast-left	slow-straight	fast-straight	fast-right	slow-right
Hit/Miss ratio	0.95	0.78	0.83	0.74	0.92	0.87

implying that the proposed method has discovered the important visual entities. To quantitatively evaluate this, the hit/miss ratio is measured across a test set of unseen data. A hit is defined as when at least 50% of the features that make up a configuration lie within the bounding box obtained from the detector. Table 6.1 shows the hit/miss ratio for each action-type.

Figure 6.14 and table 6.1 both seem to indicate that the mined action-type specific feature configurations are highly correlated with the features that lie on the lead vehicle.

The action generation mechanism is evaluated by comparing the actions generated by the proposed system on unseen test data with actions generated by the supervised approach. The supervised approach maps from the target pose, obtained from the waldboost detector, to the action parameters using a single linear regression model, $\Re^4 \rightarrow \Re^2$.



(a) Supervised method (b) Proposed method

Figure 6.15: *Generated action signals*: The generated 'turn-control' action signals (red) are shown for the proposed method and a supervised method, along with the expected action signal (blue).

Comparing the action signals generated by the supervised and proposed (unsupervised)

approaches in figure 6.15, it can be seen that both methods approximately reproduce the control signal provided by the teacher. Note that the high accuracy of the supervised approach in parts of the signal, reflects the strongly linear relationship between target pose and action signals.

The large peaks in the signal generated by the supervised approach correspond to false detections. Although there are false detections (incorrect configuration matches) in the proposed system, these generally have a minimal effect on the output as the output is the median of a number of predictions, therefore these irregularities in the action signals are generally avoided.

Certain parts of the signal generated by the proposed approach do not exactly follow the expected signal (for example from frame 100 to 150). This is in some cases due to the fact that the expected signal, provided by the teacher, includes instances of oversteer and compensation, and is therefore not necessarily superior to the generated signal.

6.7.1 Behaviour evaluation

Figures 6.16 and 6.17 demonstrate the approach at imitating the desired behaviour. In figure 6.16 the target is placed at three stationary positions and the agent is shown to generate actions that drive toward the target. In figure 6.17 the lead vehicle is driven around and the agent is shown demonstrating the desired behaviour - following the lead vehicle.

Figure 6.16 shows how the agent drives toward the stationary lead vehicle. The correct turn parameter is produced in response to the position of the lead vehicle, with occasional oversteer being compensated for in the following time steps, although in these experiments oversteer was found to be rare. As the agent approaches the lead vehicle it slows down, indicating that the control signal is being determined in part by the scale parameters of the Sift features that make up the matched feature configurations. As a single linear mapping maps from $(C * 4)$-dimensional feature pose space (where C is the number of features that make up a rule) onto 2-dimensional action space, the correlations between the two control signals are maintained. For example, the same turn

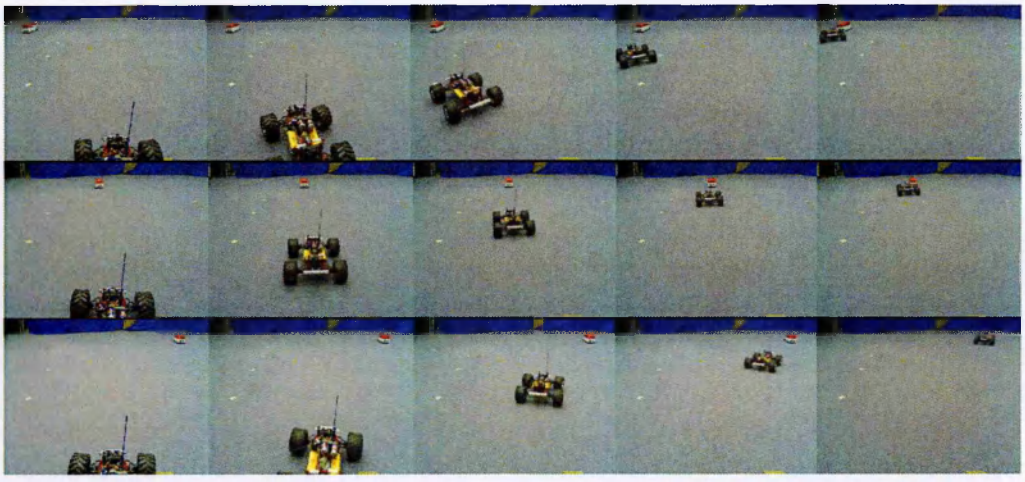


Figure 6.16: *Action generation results*: The agent is shown to demonstrate the appropriate actions, by driving (to left - top, straight - middle, to right - bottom) toward the target and then coming to stop.

control value will produce different turning angles at different speeds. This is captured by the model as both the control signals are predicted from all four pose parameters, rather than, as might be done in an engineered system, independently predicting turn and speed parameters from horizontal position and scale respectively.

In the experiment illustrated in figure 6.17 the lead vehicle is driven freely around the environment. As the agent follows the lead vehicle around it is presented with many aspects of the lead vehicle as it is viewed from many different directions. For example the lead vehicle sometimes appears side on, traveling from left to right (or right to left). In this case it was found that the correct turn control was generated, regardless of the horizontal position of the lead vehicle. This behaviour highlights the fact that the control signals are generated in response to a higher order pose space than the 4-dimensional (horizontal and vertical position, scale and orientation) feature pose space. This is achieved through the mined action-type specific feature configurations, where each configuration contains information regarding the pose of the lead vehicle, as each configuration is drawn from a particular aspect or view. Examples of such configurations can be seen in figure 6.14. Observe, in figure 6.14, the position and pose of the lead vehicle in the image in the second column on the bottom row. Although the

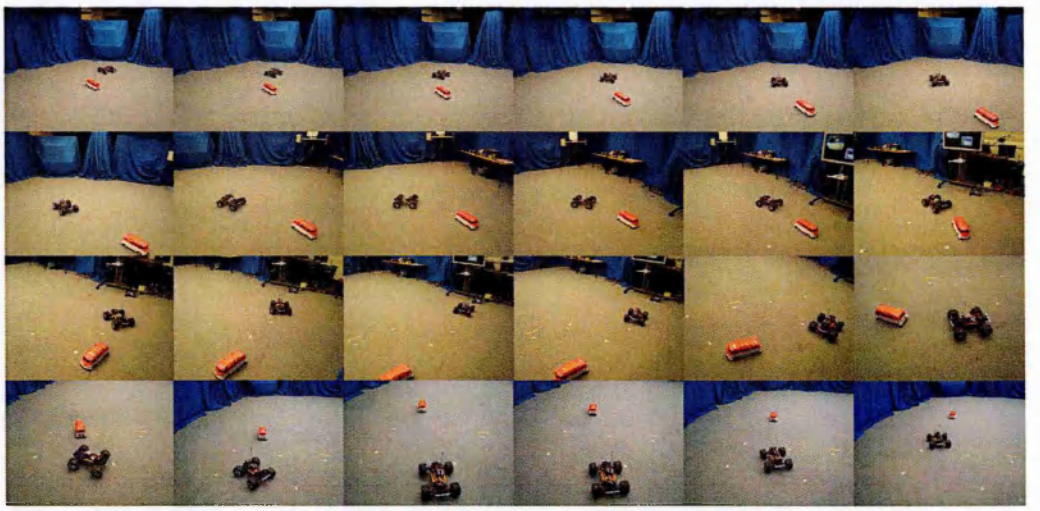


Figure 6.17: *Behaviour imitation*: The behaviour demonstrated by example is replicated by the agent, as it follows the lead vehicle. The sequence runs in raster scan starting at the top left and ending at the bottom right.

lead vehicle is to the center of the image, the configuration is associated to the 'fast-left' action-type due to the side on view. Matching to this configuration will result in the activation of the 'fast-right' linear p-a mapping.

In the experiment illustrated in figure 6.17 the lead vehicle occasionally left the view of the agent. A number of strategies were tested to cope with this occurrence. For the first and simplest strategy, if no configurations are matched in the image, the vehicle stops and waits until the vehicle is re-detected (through configuration matching). For the second strategy, the vehicle continues with the last generated action until re-detection. A third strategy involved the agent entering a 'search mode' in which it turned fully in the direction in which the lead vehicle had exited the view. Of the three strategies, the first was least problematic and resulted in the most acceptable driving behaviour. The second, although occasionally the correct strategy, often resulted in the agent driving off into a wall or obstacle, potentially at high speed. The third strategy was successful in some instances, but provided no guarantee of finding the lead vehicle and sometimes resulted in continuous turning and collision with walls and/or obstacles.

6.8 Conclusion and Discussion

This chapter presents a method for discovering the visual entities that are important to a given autonomous navigation task and utilising these perceptual representations to imitate the behaviour that is demonstrated by the teacher. The system requires no explicit definition of behaviour, uses no prior model of the objects of interest to the task and no supervision, other than the provision of input-output exemplars in the form of images and actions i.e. recorded experiences that exhibit the desired behaviour.

Partitioning the training exemplars using similarity of actions provides a means of organising the perceptual space of the agent in a way that is relevant to the problem domain. This allows for the discovery of perceptual representations that are specific to a particular class of actions. These representations are discovered using efficient association rule mining techniques. The representations are built on a spatially encoded visual word representation. The results shown in figure 6.14 and table 6.1 confirm that the visual entities discovered do in fact relate to the object in the scene that is important to the task.

By attaching action generation models (linear percept-to-action mappings) to each discovered visual entity, the system builds, in an entirely unsupervised manner, an affordance based representation of the world. This novel representation directly couples percepts to actions, resulting in a system that is able to respond to novel percepts in real time. The results presented in figures 6.15, 6.16 and 6.17 demonstrate that this novel affordance based representation generates the type of actions expected and allows the system to imitate the behaviour demonstrated by the teacher, when presented with new situations. This is achieved with no explicit definition of the behaviour.

Whilst the developed approaches performed well in the test scenarios, it is possible to construct a number of scenarios in which the approach my prove insufficient to allow full behaviour imitation. First, the use of a single rigid lead vehicle as the important visual cue is well suited to the feature configuration mining approach. The developed approach has not been shown to be capable of generalising to a class of lead vehicles, or perhaps other entities such as road lanes or people. In the case of road lanes, the mining process would be expected to identify lane markings or drivable surface regions

and to associate properties of these entities to actions e.g. road curvature should be associated to turning in a lane following task. In the case of people, the task may require the agent to follow people, or maybe to avoid collision with people. In either case, both the number and complexity of the mined association rules is likely to be considerably higher, due to the considerable variability of the appearance of people. One possible way to approach such problems would be to utilise a richer set of visual features, allowing for mining of a 'mixed bag' of features each representing a different facet of the visual environment.

The approach developed in this chapter does not model the dynamics of either the perceptual space nor the agents own action space. The approach produces a reactive stimulus-response system, in which responses are generated purely based on the current state of the discovered visual entities, without taking into account past world states or the current state of the agent. Including a representation of the dynamics of the discovered entities is likely to improve the quality of the generated actions. Not including any representation of the agents previous actions in the action generation process can result in large discontinuities in the generated control signals - the generated action could transition from a 'slow-left' type action to a 'fast-right' type action in a single time step. Some form of temporal smoothing or dynamic model would avoid the occurrence of this undesirable behaviour.

Taking the extremely complex environment of a real driving scenario on public roads as an example of a test scenario, it is apparent that the developed method would not be sufficient to imitate the desired driving behaviour. Although it may be possible to extend the approach to general lane following and perhaps obstacle avoidance (with the inclusion of additional feature descriptors and with sufficient training data), the approach can not be expected to deal with the complex interactions of multiple road entities and the behavioural rules associated with safe and legal driving. In these more complex environments, higher level logical reasoning approaches will likely be required to achieve fully autonomous behaviour.

Chapter 7

Concluding Discussion

7.1 Summary

This thesis has focused on solving the simultaneous modeling and tracking problem and generic problems in the percept action domain. Novel solutions to these problems have been developed by combining unsupervised learning methods with linear regression models. Whilst linear regression offers a computationally low cost solution to mapping from problem input to problem output spaces, unsupervised learning, which has a relatively high computational overhead, can be used to discover problem specific structure in problem input/output exemplars. This discovered structure is used to improve the modeling capabilities of the linear models. The approach of learning from example and of unsupervised learning is consistent with a fundamental motivation of the work in this thesis that is to remove the need for explicit definition of models or rules that determine the process of perception and system behaviour.

In chapter 2 relevant literature is reviewed and a discussion is made to provide background for, and give context to, the work that makes up this thesis. Particular attention is given to the changing research trends from the cognitivist to the emergent paradigm. The importance of embodiment to the process of cognition is developed with motivation and evidence drawn from philosophical, neuropsychological and AI research backgrounds. Conventional techniques and relevant literature are reviewed to the problem domains of visual tracking and visual servo control.

Chapters 3 and 4 are concerned with the development of fast visual feature tracking algorithms that utilise no prior model (hard coded or learned) of the target appearance. The approach developed operates at high frame rates, tracks fast moving objects and is adaptable to variations in appearance brought about by occlusions or changes in pose and lighting. This is achieved by employing a novel, flexible and adaptive object representation comprised of sets of spatially localised linear displacement estimators associated to various modes of a multi modal template based appearance model learnt on-the-fly. Chapter 3 focuses on the displacement estimation methods and chapter 4 develops the general Simultaneous Modeling and Tracking framework as well as methods for unsupervised learning of models of the appearance of the target object.

In chapter 3 the relationship between regression and registration techniques for displacement estimation is explored and the relative strengths of the two approaches are evaluated with illustrative results highlighting the pitfalls of the registration techniques in the presence of multiple local minima in the registration cost surface. A regression based displacement estimator, the *Linear Predictor* (LP), is developed that provides an efficient and powerful learning mechanism. LP learning methods that have low computational cost are employed, where costly predictor support selection and regression function estimation methods are avoided. Despite the non-optimal learning methods used to develop a cost-effective LP, the predictors are shown to have a well defined accuracy characteristic up to the range of displacements for which they are trained. The LP is found to allow the explicit trade off between predictor range, accuracy and computational cost.

Chapter 4 introduces a general Simultaneous Modeling and Tracking framework within which the displacement estimation process provides a mechanism for self supervision of the appearance model learning process and, in return, the appearance model provides information about the structure of the target appearance space that enables the tracker to cope with a high degree of variation in appearance. Due to the computationally low-cost, sub-optimal learning methods used for learning the LPs, it is essential that the use of the online-LP be combined with mechanisms to evaluate, weight, remove and relearn LPs online during tracking. This is achieved using an association matrix containing performance metrics for each LP within each appearance mode. Three

general components to the generic architecture are identified: an appearance model, an association matrix and a bank of displacement estimators.

Chapter 4 continues by investigating various configurations of the general tracking framework. A variation of Dowson & Bowden's SMAT algorithm is detailed and this model is incorporated into the proposed tracking framework to form the LP-SMAT tracker. The medoidshift nonparametric clustering method is also employed to model appearance variations and this model in incorporated into the proposed framework. Both the appearance models demonstrate the ability to adapt to large variations in appearance in order to manage flocks of online-LPs and to facilitate accurate and efficient tracking. Due to the flexibility of the medoidshift approach it is able to build more representative models of the object appearance space, regardless of the temporal evolution of the target appearance. Both the online-LP based methods, LP-SMAT and LP-MED, achieved good accuracy compared to other approaches, LK-SMAT and online-Boost and with considerable decrease in computational cost.

In chapter 5 a system is developed and demonstrated that is capable of learning to respond appropriately in an autonomous navigation problem, without the need for the explicit definition of behaviour. This is achieved by exploiting three major techniques. First, the imitation training framework in which domain specific embodied knowledge is imparted by allowing a teacher to demonstrate appropriate behaviour through the agents control hardware. Second the concept of modeling behaviour of the agent via the Perception Action Cycle (PAC) using a recall-reuse methodology based on learning from example. Third the proposed approach of using unsupervised learning that discovers structure in the action space in order to construct exploitable categories in the perceptual space. By facilitating a self-organizing process, this solution ensures that the perceptual categories formed by the agent are semantically grounded, where the meaning is the categories association with it's action models.

Chapter 6 develops a novel percept-action mining methodology, that is able to discover the visual entities that are important to a vision system given a specific problem and to map from these emergent perceptual entities, onto the agents action space. This is achieved by adopting many of the ideas developed throughout this thesis: imitation

learning, output space clustering and linear regression for input-output mapping. These approaches are combined with association rule mining algorithms and methods for modelling the spatial configuration of local image features. The system requires no explicit definition of behaviour, uses no prior model of the objects of interest to the task and no supervision, other than the provision of input-output exemplars in the form of images and actions i.e. recorded experiences that exhibit the desired behaviour.

In the wider context of this thesis, the processes described in chapters 3 and 4 can be viewed as the use of unsupervised learning to discover structure in a problems *input* space and to use that structure to partition the linear input-output mappings. By contrast, the mechanisms developed in chapters 5 and 6 use unsupervised learning to discover structure in a problems *output* space and to use that structure to partition the linear input-output mappings. This approach is dependent on the existence of input-output examples, a feature also of embodied systems which require interaction with the world in order to organise the cognitive process.

Two factors that strongly motivate the work in this thesis are the computational efficiency of linear mappings (both for learning and prediction), and the non-linearity of the relationships between percept and action spaces. This second factor being in part due to the generally significant non-linearity of percept spaces. The problem then, is to apply linear techniques to a non-linear problem. This is achieved here by developing novel piecewise linear mappings. A key factor in any piecewise model, is delimiting the range and domain of each linear region, or partitioning the mapping space. This is achieved in this thesis in two ways. First, in chapter 4 this is achieved by clustering the input space to partition the mapping space. In this way the boundaries of each linear region are determined by limits in the input space. Secondly, in chapter 6 the boundaries of each linear region are determined by limits in the output space. Both these approaches impose structure on both input and output space. An interesting result is that the structure imposed on the input space by output space clustering is generally more meaningful than the structure imposed on the output space by input clustering.

In fact the piecewise linear mappings developed in this thesis have another important

feature. Rather than each linear region being mapped by a single linear function, multiple linearly weighted linear functions are used within each region. These 'partitioned flocks', or constellations of linear mappings increases robustness of the overall mapping. Employing multiple regression functions also provides a means of iteratively updating the overall mapping by replacing poorly performing functions from within a constellation.

Within the strictest definition of embodiment the agent needs to be situated in the real physical world in order to develop cognitive capabilities. However a more relaxed definition might require only that the agent needs to be able to observe changes in input caused by it's output. In this sense the LP tracker achieves its mappings by simulating an embodied experience. By generating synthetic outputs (displacements) and observing the inputs (intensity differences) the tracker is able to learn a mapping between the two spaces.

7.2 Future Work

This thesis has focused on solving the simultaneous modeling and tracking problem as well as generic problems in the percept action domain. By developing generic solutions, the future application of the approaches to multiple domains is increased. The tracking approaches proposed can be applied to many tracking problems where no prior model of the target appearance is available and where an explicit trade off between computational cost and accuracy is desirable. In particular, the approach offers an alternative to the ubiquitous Lucas-Kanade tracker. Example applications include structure from motion and motion segmentation. Both these problems require many interest points to be tracked, with no a priori information about the features to be tracked.

The percept-action learning methods developed could also be extended to alternative domains. Current research is focused on using many of the components of these models to model the behaviour of drivers in a real driving scenario. By discovering and modeling structure in the drivers control space e.g. foot break usage, concepts and categories relating to the drivers perceptual space can be discovered, e.g. red traffic lights. Additionally, the percept-action data mining methodology is currently being employed for

learning visual servo control. In the context of a robot gardener, the approach is being used to learn which features of plant/leaf appearance can be used to guide the control of a robotic arm, in tasks such as pruning and inspection.

Although not investigated here, it is believed the exemplar hierarchy could have wider applications than the percept-action domain problem. The use of hierarchical output space clustering could be used in a number of applications, to discover the important variance and invariance of input parameters as well as provide general to specific generative output models. Such an application could be that of human pose estimation, where structure discovered in the pose space can be used to organise the input feature classification.

Another interesting aspect of the P-A hierarchy relates to the potential for generating multiple hypothesis from the probabilistic action models. By allowing the search to traverse multiple branches of the search tree, the system is able to generate a range of different solutions to the problem. Some work has been carried out to exploit this property of the hierarchy within a probabilistic multiple hypothesis exploration strategy, similar to a particle filter, where the hypothesis generated by the hierarchy can be propagated forward in time to discover optimal strategies.

There are many potential avenues of research that arise out of this thesis, but there are two specific application areas of interest. Visual tracking and autonomous robot control/navigation. Following are plans of future work for each of these two directions.

7.2.1 Future work: tracking

To further increase the range of applications of the proposed tracking approach, the incorporation of higher order transformations, e.g. affine deformation, into the linear prediction technique should be relatively straightforward. This could be achieved by either increasing the dimensionality of the linear mappings or alternatively by having flocks of LPs dedicated to each dimension of the transformation. An evaluation of the pros and cons of these two approaches would be of interest.

Another interesting extension of the tracking approach could be to apply the output space clustering approach to the problem. In the tracking domain, the output space

is the target pose or pose update parameters. As a first step, the current tracker (e.g. LP-MED) could be run on a number of training sequences, collecting the pose update data for each predictor at each frame. Clustering this data may reveal a distribution of updates, e.g. a cluster of small pose updates when the target is not moving, and another cluster of large updates, when the target is moving. Given higher dimensional pose updates, the clustering may reveal other pose update classes, relating to e.g. the object tilting or the object rotating. New linear predictors could then be trained to specifically handle each of the classes of pose update parameters i.e. one bank of LPs trained on going left quickely and another for rotating slowly, for example. The interesting problem will be projecting this structure back into the input (image) space. A discriminative model of the set of image transitions associated to a class of pose updates (e.g. rotating slowly), could be used to activate the appropriate subset of LP's, those trained on the 'rotating slowly' data. In this way, the output space clustering will have formed exploitable categories in the space of image transitions. A simple way to represent an image transition is as a vector of image pixel differences.

Continuing with the visual tracking problem, the shape of the target is restricted to a rectangle in the image plane under the current model. A dynamic region-growing approach based on the local and global coherence of linear displacement estimations across the image would enable researchers to model full scene dynamics without laborious model building or hand labeling of regions of interest. A starting point would be to initialise an LP-MED (or LP-SMAT) tracker on an object using the current method of identifying a rectangle on the object. Then allowing some LPs to be trained on regions outside the rectangle, and by modifying the boundaries of the rectangle to identify the convex hull of the position of LPs who are on the target. Whether an LP belongs to a target could be estimated based on a threshold of the flock agreement error developed in chapter 3. Ultimately, the research should aim to achieve full scene object and motion segmentation, using a combination LP partitioning and appearance modelling.

7.2.2 Future work: robot control

Taking the approaches of learning from example (from demonstration), output space clustering and percept-action association rule mining as a starting point, there are a number of potentially interesting avenues of future work. First, applying the approach developed in chapter 6 to alternative applications and scenarios will help delineate the scope of applications of the approach. Two application areas of interest are object manipulation with a robotic arm, and autonomous navigation. Within the autonomous navigation domain, it is not clear whether the the current visual information representation will be sufficient to represent the important characteristics to the task. For example, for a path/road/lane following task, the curvature of the lane boundary will be an important feature, but may not be captured by the local SIFT feature configurations of the existing approach. As the datamining approach allows for discovery of patterns of a 'mixed bag' of features, it could be well suited to the use of multiple feature representations. Incorporating local, global, texture, colour etc. features and feature configurations into the existing percept-action mining methodology should allow for a larger range of more complex entities to be discovered.

An important next step for the percept-action mining approach is to allow for temporal lag between percepts and actions. Sometimes actions are generated in response to percepts observed at some time in the past. Extending the data mining approach to find associations within a temporal window is therefore important. On way to achieve this may be to attach an additional symbol to the transaction vectors, obtained through quantisation of the temporal lag associated with the transaction vector.

It would also be worth investigating how the system could be applied to the task of object manipulation through a robotic arm. As a starting point, a teacher could demonstrate a simple task, e.g. grasping an object, by controlling the robotic arm. The image and control data could be processed as in chapter 6, to discover the object of interest and the associated affordances. Provided sufficient training data, the system should learn to grasp the object. Ultimately, the research should aim to achieve the development of robotic systems that could carry out complex manipulations of and interaction with complex object, by discovering the affordances associated to the important visual

entities.

Robotics is currently a technology that is largely restricted to certain manufacturing domains or specific scientific disciplines. Getting a robot to perform a particular task requires a significant effort on the part of highly specialist technical persons. In the future, it may be possible for any person to demonstrate a task to a robotic system, and for that system to then behave in the desired manner. For this to happen, the robotic system must be able to build perceptual representations that are meaningful to the given task and to learn both from experience and from knowledge imparted by a teacher. It is hoped that the flexible, generic and adaptive learning approaches developed within this thesis take steps toward such systems.

Bibliography

[1] A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59, March 1994.

[2] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–882–II–888 Vol.2, 2004.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[4] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, 2009.

[5] Simon Baker, Ralph Gross, and Iain Matthews. Lucas-kanade 20 years on: A unifying framework: Part 3. *International Journal of Computer Vision*, 56:221–255, 2002.

[6] D. Ballard and C. Brown. *Computer Vision.* Prentice-Hall, 1982.

[7] Jon L. Bentley. K-d trees for semidynamic point sets. In *SCG '90: Proceedings of the sixth annual symposium on Computational geometry*, pages 187–197, New York, NY, USA, 1990. ACM.

[8] Christian Borgelt. Efficient implementations of apriori and eclat. In *Proc. 1st*

IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL). CEUR Workshop Proceedings 90, page 90, 2003.

[9] Michelle Bovard, Matthew Gillette, Trishan de Lanerolle, Bozidar Marinkovic, Nhon H. Trinh, Peter Votto, David J. Ahlgren, Kundan Nepal, and Amir Tamrakar. Design evolution of the trinity college igvc robot alvin. *J. Robot. Syst.*, 21:461–469, September 2004.

[10] M. Brass, H. Bekkering, and W. Prinz. Movement observation affects movement execution in a simple response task. *Acta Psychol (Amst)*, 106(1-2):3–22, January 2001.

[11] Matthieu Bray, Pushmeet Kohli, and Philip H. S. Torr. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *European Conference on Computer Vision*, pages 642–655, 2006.

[12] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[13] R. A. Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

[14] R. A. Brooks. *Cambrian intelligence : the early history of the new AI.* MIT Press, Cambridge, Mass., 1999.

[15] G. Buccino, F. Binkofski, G. R. Fink, L. Fadiga, L. Fogassi, V. Gallese, R. J. Seitz, K. Zilles, G. Rizzolatti, and H. J. Freund. Action observation activates premotor and parietal areas in a somatotopic manner: an fmri study. *European Journal of Neuroscience*, 13:400–404, 2001.

[16] G. Buccino, F. Binkofski, and L. Riggio. The mirror neuron system and action recognition. *Brain Lang*, 89(2):370–376, May 2004.

[17] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proceedings*

of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil, 2007.

[18] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *European Conference on Computer Vision*, pages 484–498, 1998.

[19] G. di Pellegrino, L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Understanding motor events: a neurophysiological study. *Exp Brain Res*, 91(1):176–180, 1992.

[20] N.D.H. Dowson and R. Bowden. N-tier simultaneous modelling and tracking for arbitrary warps. In M. Chantler, R. Fisher, and M. Trucco, editors, *Proc. of the 17th British Machine Vision Conference*. British Machine Vision Association, 2006.

[21] Nicholas Dowson and Richard Bowden. Mutual information for lucas-kanade tracking (milk): An inverse compositional formulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):180–185, 2008.

[22] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, November 2000.

[23] L. Ellis, J.G. Matas, and R. Bowden. Online learning and partitioning of linear displacement predictors for tracking. *British Machine Vision Conference 08*, 1:33–43, 2008.

[24] B. Espiau, F. Chaumette, and P. Rives. In *Selected Papers from the Workshop on Geometric Reasoning for Perception and Action*, pages 106–136, London, UK, 1993. Springer-Verlag.

[25] P. Fitzpatrick. From first contact to close encounters: A developmentally deep perceptual system for a humanoid robot. *PhD thesis, MIT*, 2003.

[26] J. Randall Flanagan and Roland S. Johansson. Action plans used in action observation. *Nature*, 424(6950):769–771, August 2003.

[27] P. Forssen and G. Granlund. Robust multi-scale extraction of blob features. *Lecture Notes in Computer Science*, 2749:11–18, 2003.

[28] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[29] J. J. Gibson. *The Theory of Affordances.* Lawrence Erlbaum, 1977.

[30] Andrew Gilbert, John Illingworth, and Richard Bowden. Action recognition using mined hierarchical compound features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2010.

[31] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In M. Chantler, R. Fisher, and M. Trucco, editors, *Proc. of the 17th British Machine Vision Conference*, pages 47–56. British Machine Vision Association, 2006.

[32] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 234–247, Berlin, Heidelberg, 2008. Springer-Verlag.

[33] G. H. Granlund. The complexity of vision. *Signal Processing*, 74(1):101–126, April 1999. Invited paper.

[34] Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[35] Bruce B. Hansen and Bryan S. Morse. Multiscale image registration using scale trace correlation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2202, 1999.

[36] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.

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

[38] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.

[39] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[40] M. Jeannerod. *Motor Cognition: What Actions Tell the Self.* Oxford University Press, 2006.

[41] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. In *Conference on Computer Vision and Pattern Recognition*, pages 415–422, 2001.

[42] Sébastien Jodogne and Justus H. Piater. Interactive learning of mappings from visual percepts to actions. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 393–400, New York, NY, USA, 2005. ACM.

[43] Marggie Jones and David Vernon. Using neural networks to learn hand-eye coordination. *Neural Computing and Applications*, 2(1):2–12, 1994.

[44] Frederic Jurie and Michel Dhome. Real time robust template matching. In *in British Machine Vision Conference 2002*, pages 123–131, 2002.

[45] Z Kalal, J Matas, and K Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. *Conference on Computer Vision and Pattern Recognition*, 2010.

[46] J. Kolodner. *Case-based reasoning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[47] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[48] George Lakoff and Mark Johnson. *Philosophy in the Flesh : The Embodied Mind and Its Challenge to Western Thought.* Basic Books, December 1999.

[49] D. B. Leake. *Case-Based Reasoning: Experiences, Lessons and Future Directions.* MIT Press, Cambridge, MA, USA, 1996.

[50] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[51] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pages 674–679, 1981.

[52] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, volume 1, pages 281–297, 1967.

[53] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2d-3d model-based approach. In *International Conference on Computer Vision*, pages 262–268, 1999.

[54] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[55] J. Matas, K. Zimmermann, T. Svoboda, and A. Hilton. Learning efficient linear predictors for motion estimation. In *Proceedings of 5th Indian Conference on Computer Vision Graphics and Image Processing*, pages 445–456, 2006.

[56] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):810–815, 2004.

[57] R. Navaratnam, A. W. Fitzgibbon, and R. Cipolla. The joint manifold model for semi-supervised multi-valued regression. In *International Conference on Computer Vision*, pages 1–8, 2007.

[58] Nils J. Nilsson. *Introduction To Machine Learning.* http://ai.stanford.edu/people/nilsson/MLDraftBook/, 1996.

[59] Eng-Jon Ong and Richard Bowden. Robust facial feature tracking using shape-constrained multi-resolution selected linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2010.

[60] Lucas Paletta and Gerald Fritz. Reinforcement learning of predictive features in affordance perception. In *Proceedings of the 2006 international conference on Towards affordance-based robot control*, pages 77–90. Springer-Verlag, 2008.

[61] Dana Paquin, Doron Levy, Eduard Schreibmann, and Lei Xing. Multiscale image registration. *Math. Biosci. Eng.*, 3(2):389–418, 2006.

[62] T. Quack, V. Ferrari, B. Leibe, and L. Van Gool. Efficient mining of frequent and distinctive feature configurations. pages 1–8, 2007.

[63] G. Rizzolatti and L. Craighero. The mirror-neuron system. *Annual Review of Neuroscience*, 27(1):169–192, 2004.

[64] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[65] A. C. Sanderson and L. E. Weiss. Image based visual servo control using relational graph error signals. *Conference on Cybernetics and Society*, pages 1074–1077, 1980.

[66] Yaser Ajmal Sheikh, Erum Arif Khan, and Takeo Kanade. Mode-seeking by medoidshifts. In *Eleventh IEEE International Conference on Computer Vision (ICCV 2007)*, number 141, October 2007.

[67] Heung-Yeung Shum and Richard Szeliski. Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, February 2000.

[68] J.M. Siskind. Reconstructing force-dynamic models from video sequences. *Artificial Intelligence*, 151(1-2):91–154, 2003.

[69] A. Sloman and J. Chappell. The altricial-precocial spectrum for robots. In *International Joint Conferences on Artificial Intelligence*, pages 1187–1192, 2005.

[70] Jan Sochman and Jiri Matas. Waldboost: Learning for time constrained sequential detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 150–156, Washington, DC, USA, 2005. IEEE Computer Society.

[71] M. Sridhar, A. G. Cohn, and D. C. Hogg. Learning functional object-categories from a relational spatio-temporal representation. In *Proceeding of the 2008 confer-*

ence on ECAI 2008, pages 606–610, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[72] M. Taddeo and L. Floridi. Solving the symbol grounding problem: a critical review of fifteen years of research. *Journal of Experimental and Theoretical Artificial Intelligence*, 17:419–445, 2005.

[73] E. Thelen and L. B. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action (Cognitive Psychology)*. The MIT Press, January 1996.

[74] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge: Research articles. *J. Robot. Syst.*, 23:661–692, September 2006.

[75] T. van Gelder and R. F. Port. It's about time: an overview of the dynamical approach to cognition. pages 1–43, 1995.

[76] F. J. Varela, E. Thompson, and E. Rosch. *The Embodied Mind: Cognitive science and human experience*. MIT Press, 1991.

[77] D. Vernon. Cognitive vision: The case for embodied perception. *Image and Vision Computing*, 26(1):127–140, January 2008.

[78] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57:137–154, May 2004.

[79] L. Weiss, A. C. Sanderson, and C. P. Neuman. *Dynamic Sensor-Based Control of Robots with Visual Feedback*, RA-3(1), October 1987.

[80] S. D. Whitehead and D. H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83, 1991.

[81] Oliver M. C. Williams, Andrew Blake, and Roberto Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *International Conference on Computer Vision*, pages 353–361, 2003.

[82] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13, 2006.

[83] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):677–692, 2009.

[84] Karel Zimmermann. *Fast Learnable Methods for Object Tracking.* PhD thesis, Center for Machine Perception (CMP), Czech Technical University, Prague, Czech Republic, 2008.

Reproduced with permission of copyright owner. Further reproduction prohibited without permission.