Adaptive Supervision Online Learning
for Vision Based Autonomous Systems

Kristoffer Öfjäll
Adaptive Supervision Online Learning
for Vision Based Autonomous Systems

© 2016 Kristoffer Öljäll

Department of Electrical Engineering
Linköping University
SE-581 83 Linköping
Sweden

ISBN 978-91-7685-815-8 ISSN 0345-7524
Abstract

Driver assistance systems in modern cars now show clear steps towards autonomous driving and improvements are presented in a steady pace. The total number of sensors has also decreased from the vehicles of the initial DARPA challenge, more resembling a pile of sensors with a car underneath. Still, anyone driving a tele-operated toy using a video link is a demonstration that a single camera provides enough information about the surrounding world.

Most lane assist systems are developed for highway use and depend on visible lane markers. However, lane markers may not be visible due to snow or wear, and there are roads without lane markers. With a slightly different approach, autonomous road following can be obtained on almost any kind of road. Using real-time online machine learning, a human driver can demonstrate driving on a road type unknown to the system and after some training, the system can seamlessly take over. The demonstrator system presented in this work has shown capability of learning to follow different types of roads as well as learning to follow a person. The system is based solely on vision, mapping camera images directly to control signals.

Such systems need the ability to handle multiple-hypothesis outputs as there may be several plausible options in similar situations. If there is an obstacle in the middle of the road, the obstacle can be avoided by going on either side. However, the average action, going straight ahead, is not a viable option. Similarly, at an intersection, the system should follow one road, not the average of all roads.

To this end, an online machine learning framework is presented where inputs and outputs are represented using the channel representation. The learning system is structurally simple and computationally light, based on neuropsychological ideas presented by Donald Hebb over 60 years ago. Nonetheless the system has shown a capability to learn advanced tasks. Furthermore, the structure of the system permits a statistical interpretation where a non-parametric representation of the joint distribution of input and output is generated. Prediction generates the conditional distribution of the output, given the input.

The statistical interpretation motivates the introduction of priors. In cases with multiple options, such as at intersections, a prior can select one mode in the multimodal distribution of possible actions. In addition to the ability to learn from demonstration, a possibility for immediate reinforcement feedback is presented. This allows for a system where the teacher can choose the most appropriate way of training the system, at any time and at her own discretion.

The theoretical contributions include a deeper analysis of the channel representation. A geometrical analysis illustrates the cause of decoding bias commonly present in neurologically inspired representations, and measures to counteract it. Confidence values are analyzed and interpreted as evidence and coherence. Further, the use of the truncated cosine basis function is motivated.

Finally, a selection of applications is presented, such as autonomous road following by online learning and head pose estimation. A method founded on the same basic principles is used for visual tracking, where the probabilistic representation of target pixel values allows for changes in target appearance.
Populärvetenskaplig sammanfattning


I flera fall krävs att multipla hypoteser kan hanteras. Ifall det dyker upp en alg mitt på vägen finns möjlighet att styra undan antingen åt höger eller åt vänster. Däremot är medelstyrsignalen, det vill säga rakt fram, inte alls ett bra val. Även korsningar utgör ett liknande exempel där fordonet bör välja en väg, inte ett medelvärde av alla vägar.


Acknowledgments

How did I end up here? All alone at Linköping University late at night. The night before sending this manuscript off to the print shop. It is interesting how past and seemingly small decisions, with time can have such a great influence on the path of life.

Where would I otherwise be? Just as well I might have ended up in Östersund. Any one of these small decisions could easily have been different. Strangely enough, many of those who talked about moving somewhere else seem to have stayed, while many of us who didn’t mind staying seem to have ended up all over. Modern communication tools make these diverging paths so easy to follow.

Being here late at night is not unfamiliar. Somehow, no matter how early some manuscript is finished, there is always something more to do, some details to attend to. There may even be some time to run another experiment. It ain’t over until the submission deadline has passed.

Who would have guessed that working for a PhD in computer vision would bring you to places like the hybrid theater in Gothenburg, observing an open heart surgery, or to bird migration research in Lund, or a steel mill in Luleå, experiencing the heat of a passing bucket containing 120 metric tons of liquid steel. The second time of swimming in the just-above-freezing ocean outside Ystad I should have seen coming since I may have initiated the idea, but perhaps not the detour along some gravel road on the way back to Linköping.\footnote{Fortunately, most people in the mini-van were asleep at the time.} Who would have guessed that one would end up staying in a small village outside Jülich for a week, commuting by train and working with industrial robots, where any mistake would lead to more than just an error message on the screen.

Somehow you end up in strange theoretical places as well, getting lost in Minkowski space or experiencing the brain version of delayed onset muscle soreness after working in four- or five-dimensional spaces for some time. The traditional muscle version seems not as common as a result of research, even after chasing autonomous vehicles running amok. Some jumping-over-waves with a jet-ski in Florida during a conference break tend to do the trick on the other hand.

Somehow you tend to run into people inviting you to all sorts of adventures, and joining on expeditions to everything from the highest summits of northern Europe to caves and mines far below the surface, or a spontaneous drive across ten countries, or an evening by the campfire in the back yard.

One thing is very certain, I wouldn’t be the same without everyone I’ve met over the years. Many of you have had, and still have, a great influence on the path of my life, more or less intentionally, affecting these seemingly small decisions. Adventures seem to always bring along weather. I greatly appreciate everyone who have kept me company during the bad weather, and everyone who have shared the good weather, for what would any adventure be without anyone to share it with?

Now, as the nowadays quite familiar life as a PhD student is close to its end and as I enjoy the eleventh year (out of three) in Linköping, I’m looking forward to new adventures with many good old and new friends. As of now, the location of those adventures is quite unknown, and yet again, we seem to diverge all over the
planet. However, with great friends all over the world comes great opportunities for visiting unexpected places.

Returning to the text that awaits, some people have had a more direct influence on its existence. First to be mentioned is of course my supervisor Michael and everyone who has been at CVL over the last years and have created such an inspiring environment. With the people at TST, I even got the opportunity to experience the real world for a while. A great thanks goes to all current PhD students of CVL who have taken their time to find the many errors that once resided among these pages.

Finally, I would like to thank my family for unlimited support in any matter, and, Mikaela for joining me on the greatest adventure of all.

This work has been supported by EC’s 7th Framework Programme, grant agreement 247947 (GARNICS), by SSF through a grant for the project CUAS, by VR through grants for the projects VIDI and ETT, through the Strategic Areas for ICT research CADICS and ELLIIT.

Kristoffer Öljäll  May 2016

About the Cover

The cover is a photograph of petroglyphs from Östergötland, Sweden. The rock carvings are dated to the dawn of autonomous vehicles at around AD 2000. On the front page, there are three figures depicting a vehicle with a camera, a channel representation with two modes and an animal. Interpreting the represented distribution as predicted steering control, the two modes correspond to avoiding the animal by turning either left or right. The back cover contains an illustration of the geometry of four modular channels, a 3D simplex and the intersection curve with the surface of a sphere, see figure 4.2 on page 51.
Contents

1 Introduction 1
  1.1 Motivation ........................................... 2
  1.2 Thesis Overview ...................................... 3
  1.3 Supplementary Material .............................. 5
  1.4 List of Symbols and Notation ........................ 6
  1.5 Included Publications ................................. 7
  1.6 Related Publications .................................. 13
  1.7 Figures .................................................. 18

I Representation 21

2 Introduction to Representations 23
  2.1 Biological Visual Representation ..................... 25
  2.2 Visual Information Representation in Computer Systems 25

3 The Channel Representation 27
  3.1 Introduction to the Channel Representation .......... 28
  3.2 Basis Functions and Encoding ........................ 30
  3.3 Uniqueness of the Cosine Basis Function ............ 35
  3.4 Statistical Interpretation ............................. 39
  3.5 Summary and Contributions ............................ 43

4 Decoding the Channel Representation 45
  4.1 A Brief History of Channel Decoding .................. 45
  4.2 Geometric Analysis .................................... 50
  4.3 Evidence and Coherence ............................... 54
  4.4 Decoding Bias, Cause and Mitigation ................ 56
  4.5 Summary and Contributions ............................ 65
CONTENTS

II Learning Methods

5 Introduction to Machine Learning 69
  5.1 Properties of Learning Methods ..................... 70
  5.2 Learning Methods and Training Data .................. 72
  5.3 Locally Weighted Projection Regression ............... 75
  5.4 Random Forest Regression ......................... 76
  5.5 Hebbian Learning .................................. 77
  5.6 Associative Learning .............................. 79

6 Hebbian Associative Learning 83
  6.1 Introducing Hebbian Associative Learning .......... 83
  6.2 From Hebbian to qHebb Learning ................... 87
  6.3 Coherence in Hebbian Associative Learning .......... 89
  6.4 Incorporating Priors .............................. 91
  6.5 Learning from Performance Feedback ................. 93
  6.6 Models for Visual Object Tracking .................. 94
  6.7 Summary and Contributions .......................... 96

7 Factorization and Non-Linear Channels 99
  7.1 Subspace Factorization ............................. 99
  7.2 Time Logarithmic Channels ......................... 105
  7.3 Log-Polar Channels ............................... 107
  7.4 Summary and Contributions .......................... 112

III Applications 113

8 Online Learning of Autonomous Road Following 115
  8.1 Introduction and Related Work ....................... 115
  8.2 Evaluation System and Experimental Setup ........... 117
  8.3 Offline Learning of Steering ...................... 120
  8.4 Learning of Unimodal Steering Control .............. 121
  8.5 Learning of Multimodal Steering Control ............ 125
  8.6 Multimodal Steering and Throttle Control .......... 126
  8.7 Learning to Follow a Human ....................... 134
  8.8 Summary and Conclusion ............................ 135

9 Bird and Head Orientation Regression 137
  9.1 Head Pose Regression .............................. 137
  9.2 Bird Position and Pose Estimation .................. 140
  9.3 Conclusion .................................... 143
CONTENTS

10 Visual Object Tracking 145
  10.1 General Tracking Framework .......................... 145
  10.2 Experiments ............................................. 146
  10.3 Conclusion ................................................. 150

11 Projected Histogram Rail Curvature Estimation 151
  11.1 Introduction ............................................... 151
  11.2 Rail Geometry ............................................. 153
  11.3 Conclusion ................................................. 157

IV Concluding Remarks 159

12 Discussion on Past, Present and Future Work 161

13 Conclusion 165
This work presents a learning system for online multiple-hypothesis learning with visual applications in mind. The work mostly resides within machine learning and computer vision, with support from statistics and probability theory, and inputs from neuroscience. The applications bring connections to areas such as robotics and autonomous vehicles, train safety and thermal infrared vision, and biology with bird migration research.

The channel representation is central, and will follow along from the introduction of representations through learning and end up among the applications. The channel representation itself is also treated in this work, providing new theoretical results based on probabilistic and geometric interpretations. Some of the biologically inspired representations have issues with bias, that is, the results depend on the absolute positions of channel centers. A geometric analysis sheds light on this issue.

The learning system itself is quite simple. Most of its power stems from the representation. The Hebbian associative learning is first introduced by a series of figures, aiming for an easy to follow introduction with minimal prerequisites. A natural interpretation of the learning framework is obtained through a probabilistic view of the representation. This view also motivates additional extensions to the learning system.

A range of applications is presented, the most prominent is the online learning of purely vision-based online road following. The needs of this application have set the direction of research in the learning system. Emerging ideas can almost directly be transferred to visual object tracking. The channel representation also carries over to train safety applications, in the simplified form of histograms. In the bird tracking application, the log–polar arrangement of channels sees its first practical application.
Figure 1.1: Illustration of multiple hypotheses in inverse kinematics. There are several sets of joint positions that will solve the reaching problem. These different solutions need to be separated, as most linear combinations of solutions will most likely not be a valid solution.

1.1 Motivation

Mostly the text ahead will assume of the reader an already existing interest in the channel representation. The channel representation may not be widely known under that specific name, however similar ideas have been around under different names in different areas for a long time. The channel representation [38], along with the similar idea of population coding [23], emerged among systems using biomimetic representations. From the statistical side come histograms, especially soft histograms and kernel density estimation [84] with strong connections to the channel representation. The radial basis functions [14] appeared in machine learning. The most common functions used such as the Gaussian lack the norm constancy properties of the common channel representation basis function, however the similarities are striking. In computer vision, the SIFT descriptor [68] and the color names approach [111] implement the ideas of channel representations. The first is based on receptive fields and weighted histograms, the second describes colors in terms of basis functions in the color space, where the basis functions are placed at colors with a specific name in the English language.

The primary reason for using the channel representation in this work is the ability to represent probability density functions with multiple modes, that is, multiple hypotheses. Interpreting the channel representation in terms of density functions brings ideas from probability theory, providing ideas for new development and providing a framework for deeper analysis of the representation. Especially Bayesian ideas are of interest.

These properties of the representation are useful for constructing a learning system where ranges of input values are associated with ranges of output values. An online learning system is obtained by using the learning ideas of Hebb [44].
The motivation for Hebbian associative learning stems from one of the applications and the lack of learning systems fulfilling all requirements while still being computationally lightweight. When looking for online and real-time learning systems capable of learning many-to-many mappings, there are not many options left on the shelves. Furthermore, looking at applications with visual and thus high-dimensional data, some of the last remaining options drop out.

One particular application is online learning of visual autonomous road following. The use case is simple: in a not very distant future, the regular lane following system in the car shuts down when turning onto a smaller road or when snow appears on the road. The human driver then drives the car for a minute while the online learning road following system learns whereafter the car follows the road autonomously again, using only a single visual camera already present in the car, used by the regular active safety systems.

The need for online learning is obvious in this case. Online learning is also teacher-friendly as there is no waiting for the system to learn. Training directly affects system behavior, providing direct feedback to the teacher. In case there is a need for more training, this can be provided immediately when the need is identified and the system then seamlessly returns to autonomous operation.

Multiple-hypothesis outputs, or multimodal outputs are required when there are several different solutions to a problem, but where the mean of all solutions is not a valid solution. Typical examples include inverse kinematics, see figure 1.1, where each input, a desired pose of the end effector, has several possible joint configurations as output. Another example is for autonomous vehicles at intersections. Typical unimodal methods average outputs when training samples with similar inputs are seen. This reduces prediction noise in general, but averaging across different solutions will generate invalid predictions. There may also be a case where the current input is ambiguous but still provides some information – a multimodal representation can capture the essence of such a case. Later, additional information may resolve the ambiguity, see figure 1.2.

Hebbian associative learning on channel-represented inputs and outputs is fast and provides more precise predictions than a corresponding histogram representation. The system is interpretable in terms of distributions of the input and output, and the joint distribution connecting them. The associative linkage matrix directly illustrates which input ranges are associated with which output ranges. This allows advanced abilities to be learned using a simple algorithm.

1.2 Thesis Overview

This section shall, together with the table of contents, provide you with a mental map of the material ahead. This text briefly presents the theoretical sections and points out novel contributions. For those aiming directly for the applications, this section will help identifying the required theory.

This thesis contains three major parts: representation, learning methods and applications. Later parts build upon the contents of the former. The representation and learning parts contain contributions more on the theoretical side while
CHAPTER 1. INTRODUCTION

Figure 1.2: Illustration of multiple hypotheses, where additional information is required to tell the cases apart. Given only the middle image, it is not possible to infer the intended 3D structure of the illustration. The figure is repeated on either side, where additional information is provided to resolve the ambiguity.

the applications part obviously contains contributions of more practical nature.

The first chapter of each theoretical part, chapters 2 and 5, contain an introduction of more general character and are free from contributions. For most readers, their contents will already be familiar. However, section 5.6 introduces the notation and background for Hebbian associative learning, the foundation of qHebb learning, which is the major contribution to machine learning.

1.2.1 Representation

Most of the material is based on the channel representation. It is introduced in sections 3.1 and 3.2. These sections contain mostly previous work. For re-obtaining a conventional representation, section 4.1 presents the decoding procedure. These sections form the spine and are required to follow most of the remaining material. Additionally, section 4.1 contains some contributions such as a maximum likelihood decoding and a variation of the conventional decoding procedure, more suitable for the following geometrical interpretation.

The remainder of chapters 3 and 4 present contributions related to the channel representation. A quick summary of section 3.3 is: use the truncated cos² channels, not e.g. Gaussian or B-splines. Section 3.4 introduces a probabilistic interpretation of the representation. This is later used for exploring properties of the representation and for governing design decisions. A geometrical interpretation is presented in section 4.2. It is the foundation of further theoretical contributions in the chapter. The reader feeling more at home in Minkowski space will find the same material presented using group theory in the corresponding publication [33].

Section 4.3 expands the interpretation of certainty measures for the representation. As will be seen, such information is integrated in the representation itself. Finally, the main contribution of section 4.4 is an analysis of when decoding bias appear and how it can be reduced. Beware that a cousin of the channel represen-
1.3. SUPPLEMENTARY MATERIAL

A presentation will appear rather unexpectedly in this section, the population coding and the corresponding readout. The bias properties of the two are compared, but the focus then returns to the channel representation in time for the step to machine learning.

1.2.2 Learning Methods

Following the introduction chapter, the part on learning methods is mostly focused on Hebbian associative learning and the generalization qHebb. Section 6.1 presents a graphical introduction to associative learning on the channel representation, intended to be easy to follow. This provides an overview of the area where the contributions of chapter 6 are settled.

The main contribution of this part is the qHebb learning, presented in section 6.2. The remaining sections in this chapter contain related contributions. Coherence, a property of the channel representation, is transferred to associative learning in section 6.3. It is used for reducing the influence of noise. In section 6.4, priors are introduced for giving higher-level systems the possibility of affecting the associative learning system in an expressive way. Up to now, qHebb has been based on supervised learning. In section 6.5, steps are provided leading towards reinforcement learning. Finally, the ideas of qHebb are transferred to visual object tracking in section 6.6.

Chapter 7 contains a selection of promising contributions that have not yet reached common use in applications. Section 7.1 presents subspace factorization for reducing computational requirements. The possibilities of partially specified input data is related and also treated here. Some results regarding logarithmic channel placement is presented in section 7.2. Its main use is for representing events in time, where higher temporal resolution is required close to the present. One typical example is for pixel-based background models and change detection. The log-polar channel placement, presented in section 7.3, is the corresponding spatial idea. In biological systems, it is known as foveal vision.

1.3 Supplementary Material

An electronic copy is available at http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125916. The page also contains supplementary material such as videos.
1.4 List of Symbols and Notation

The following list contains symbols and notation commonly used throughout this work. In general, bold face lower case letters denote vectors and bold face upper case letters denote matrices. At some occasions, summation limits are left out. In such cases, the sum is over all elements of the corresponding vectors or matrices.

\begin{itemize}
  \item $i$ The imaginary unit, $i^2 = -1$.
  \item $b(\cdot)$ Basis function of a channel representation.
  \item $b_n(\cdot)$ Basis function centered around $n$.
  \item $w$ The width (support) of the basis function.
  \item $\mathbf{x}$ Channel vector with components $x_n$, usually a channel encoding of $\xi$.
  \item $\xi$ Scalar or vector to be channel encoded.
  \item $\mathcal{C}(\cdot)$ The channel encoding operator, $\mathbf{x} = \mathcal{C}(\xi)$.
  \item $\mathcal{C}^\dagger(\cdot)$ The channel decoding operator, $\hat{\xi} = \mathcal{C}^\dagger(\mathbf{x})$.
  \item $\hat{\cdot}$ In general, a prediction or estimate.
  \item $\cdot^T$ Transpose.
  \item $\otimes$ Kronecker product.
  \item $\odot$ Hadamard, element wise, matrix product.
  \item $|\cdot|$ Absolute value.
  \item $\text{arg}(\cdot)$ Argument of a complex number.
  \item $\|\cdot\|_p$ $L_p$ vector norm, $\left(\frac{1}{N} \sum_{n=1}^{N} |x_n|^p\right)^{\frac{1}{p}}$.
  \item $\text{diag}(\cdot)$ $N$ by $N$ matrix with the supplied vector along the diagonal.
  \item $[\cdot]_k$ The $k$th element of a vector.
  \item $[\cdot]_{kl}$ The element in the $k$th row and $l$th column of a matrix.
  \item $\mathbb{R}$ The set of real numbers.
  \item $\mathbb{Z}$ The set of integers.
  \item $\mathbb{Z}^+$ The set of strictly positive integers.
\end{itemize}
1.5 Included Publications

This section lists the publications on which this work is based, together with the contributions of the author in cases where there are multiple authors.

1.5.1 Visual Autonomous Road Following by Symbiotic Online Learning


Abstract:
Recent years have shown great progress in driving assistance systems, approaching autonomous driving step by step. Many approaches rely on lane markers however, which limits the system to larger paved roads and poses problems during winter. In this work we explore an alternative approach to visual road following based on online learning. The system learns the current visual appearance of the road while the vehicle is operated by a human. When driving onto a new type of road, the human driver will drive for a minute while the system learns. After training, the human driver can let go of the controls. The present work proposes a novel approach to online perception-action learning for the specific problem of road following, which makes interchangeably use of supervised learning (by demonstration), instantaneous reinforcement learning, and unsupervised learning (self-reinforcement learning). The proposed method, symbiotic online learning of associations and regression (SOLAR), extends previous work on qHebb-learning in three ways: priors are introduced to enforce mode selection and to drive learning towards particular goals, the qHebb-learning methods is complemented with a reinforcement variant, and a self-assessment method based on predictive coding is proposed. The SOLAR algorithm is compared to qHebb-learning and deep learning for the task of road following, implemented on a model RC-car. The system demonstrates an ability to learn to follow paved and gravel roads outdoors. Further, the system is evaluated in a controlled indoor environment which provides quantifiable results. The experiments show that the SOLAR algorithm results in autonomous capabilities that go beyond those of existing methods with respect to speed, accuracy, and functionality.

Contribution:
This work presents further analysis and extensions of the channel based Hebbian associative learning systems. The proposed methods are based on the probabilistic interpretation of the channel representation and enables new learning modalities in addition to the possibility of learning from demonstration. The author developed the ideas leading to this publication, implemented the associative learning systems, performed the online experiments and did the main part of the writing.
1.5.2 Emlen-funnel experiments revisited: methods update for studying compass orientation in songbirds


Abstract:
Migratory songbirds weighing only a few grams carry an inherited capacity to migrate several thousand kilometres each year crossing continental landmasses and barriers between distant breeding sites and wintering areas. How individual songbirds manage with extreme precision to find their way during the migratory journey is still largely unknown. The functional characteristics of biological compasses used by songbird migrants has mainly been investigated by recording the birds directed migratory activity in circular cages, so-called Emlen-funnels. The method to record songbird orientation is over 50 years old and has not received major updates over the past decades. The aim of this work is to combine the traditional Emlen-funnel method with novel digital-image analysis and compare the results from new digital methods with the established manual methods to evaluate songbird migratory activity and orientation in circular cages.

We performed orientation experiments in a modern orientation testing facility using the European robin (Erithacus rubecula) as our study species. We used circular modified Emlen-funnels equipped with thermal-paper and simultaneously recorded the songbird movements in the cages from above with a digital camera. We evaluated and compared the results obtained with five different methods. Two methods have been commonly used in songbirds’ orientation experiments; the other three methods were developed for this study and were based either on evaluation of the thermal-paper using automated image analysis, or on the analysis of videos recorded during the experiment. The video analyses were performed by both manual annotation and by a more sophisticated computer vision algorithm.

The three methods used to evaluate scratches produced by the claws of birds on the soft surface of the thermal-papers produced similar results, but presented some differences compared with the video analyses. These differences were caused mainly by differences in scatter, as any movement of the bird along the sloping walls of the funnel was recorded on the thermal-paper when the bird fluttered around the cage. The video evaluations allowed us to detect single take-off attempts by the birds, and to consider only this behaviour in the following orientation analyses. Computer vision also made it possible to identify and separately evaluate different behaviours, such as wing fluttering, distance displaced and body alignment, that was impossible to record by the thermal-paper, providing new insight in the level of behavioural complexity that songbirds’ express during periods of migratory restlessness.

The traditional Emlen-funnel is still the most used method to investigate compass orientation in songbirds under experimentally controlled conditions, both in the field and in laboratory. However, there is a need for more detailed information to unveil the relevance of specific behaviour during the birds’ migratory restless-
1.5. INCLUDED PUBLICATIONS

ness phase. Moreover, a more consistent procedure of analysis, possibly not user
biased, will allow for easier comparison of results from different types of experi-
ments. Although the use of thermal-paper as registration medium is still the most
used in Emlen-funnel studies, new numerical image analysis techniques currently
available provide effective tools to investigate in detail the songbirds’ migratory
behaviour. Thus, video analysis can be a stand-alone method or a complementary
method to the thermal-paper. By using automated video analysis, it is possible to
reach a much higher level of understanding of the behaviour of captive birds and
since computer vision is a constantly growing discipline, there will be an increas-
ing number of possibilities to evaluate and quantify specific behaviours as new
algorithms are developed in the future.

Contribution:
The author has contributed image processing routines for extracting bird position
and orientation from video. Furthermore, the author contributed the parts of the
manuscript related to image processing.

1.5.3 Unbiased Decoding of Biologically Motivated Visual
Feature Descriptors

Michael Felsberg, Kristoffer Öfjäll, and Reiner Lenz. Unbiased decoding
of biologically motivated visual feature descriptors. Frontiers in
Robotics and AI, 2(20), 2015.

Abstract:
Visual feature descriptors are essential elements in most computer and robot vi-
sion systems. They typically lead to an abstraction of the input data, images, or
video, for further processing, such as clustering and machine learning. In cluster-
ing applications, the cluster center represents the prototypical descriptor of the
cluster and estimates the corresponding signal value, such as color value or domi-
nating flow orientation, by decoding the prototypical descriptor. Machine learning
applications determine the relevance of respective descriptors and a visualization
of the corresponding decoded information is very useful for the analysis of the
learning algorithm. Thus decoding of feature descriptors is a relevant problem,
frequently addressed in recent work. Also, the human brain represents sensori-
motor information at a suitable abstraction level through varying activation of
neuron populations. In previous work, computational models have been derived
that agree with findings of neurophysiological experiments on the representation of
visual information by decoding the underlying signals. However, the represented
variables have a bias toward centers or boundaries of the tuning curves. Despite
the fact that feature descriptors in computer vision are motivated from neuro-
science, the respective decoding methods have been derived largely independent.
From first principles, we derive unbiased decoding schemes for biologically mo-
tivated feature descriptors with a minimum amount of redundancy and suitable
invariance properties. These descriptors establish a non-parametric density esti-
mation of the underlying stochastic process with a particular algebraic structure.
Based on the resulting algebraic constraints, we show formally how the decoding
problem is formulated as an unbiased maximum likelihood estimator and we derive a recurrent inverse diffusion scheme to infer the dominating mode of the distribution. These methods are evaluated in experiments, where stationary points and bias from noisy image data are compared to existing methods.

**Contribution:**
The author has contributed with ideas concerning decoding invariant processing, derived computational schemes, performed simulations, and contributed to the text.

### 1.5.4 Detecting Rails and Obstacles Using a Train-Mounted Thermal Camera


**Abstract:**
We propose a method for detecting obstacles on the railway in front of a moving train using a monocular thermal camera. The problem is motivated by the large number of collisions between trains and various obstacles, resulting in reduced safety and high costs. The proposed method includes a novel way of detecting the rails in the imagery, as well as a way to detect anomalies on the railway. While the problem at a first glance looks similar to road and lane detection, which in the past has been a popular research topic, a closer look reveals that the problem at hand is previously unaddressed. As a consequence, relevant datasets are missing as well, and thus our contribution is two-fold: We propose an approach to the novel problem of obstacle detection on railways and we describe the acquisition of a novel data set.

**Contribution:**
This work presents the first stages of a train safety system. The contributions of the author are the parts related to scene geometry and rail detection, where the author developed the ideas, implemented the subsystem and did the main part of the writing.

### 1.5.5 Online Learning of Vision-Based Robot Control during Autonomous Operation


**Abstract:**
Online learning of vision-based robot control requires appropriate activation strategies during operation. In this chapter we present such a learning approach with
applications to two areas of vision-based robot control. In the first setting, self-evaluation is possible for the learning system and the system autonomously switches to learning mode for producing the necessary training data by exploration. The other application is in a setting where external information is required for determining the correctness of an action. Therefore, an operator provides training data when required, leading to an automatic mode switch to online learning from demonstration. In experiments for the first setting, the system is able to autonomously learn the inverse kinematics of a robotic arm. We propose improvements producing more informative training data compared to random exploration. This reduces training time and limits learning to regions where the learnt mapping is used. The learnt region is extended autonomously on demand. In experiments for the second setting, we present an autonomous driving system learning a mapping from visual input to control signals, which is trained by manually steering the robot. After the initial training period, the system seamlessly continues autonomously. Manual control can be taken back at any time for providing additional training.

Contribution:
This work presents two learning robotic systems where both learning and operation is online. The primary advantage compared to the system in [27] is the possibility to seamlessly switch to training mode if the initial training is insufficient. The author developed the ideas leading to this publication, implemented the systems, performed the experiments and did the main part of the writing.

1.5.6 Weighted Update and Comparison for Channel-Based Distribution Field Tracking


Abstract:
There are three major issues for visual object trackers: model representation, search and model update. In this paper we address the last two issues for a specific model representation, grid based distribution models by means of channel-based distribution fields. Particularly we address the comparison part of searching. Previous work in the area has used standard methods for comparison and update, not exploiting all the possibilities of the representation. In this work we propose two comparison schemes and one update scheme adapted to the distribution model. The proposed schemes significantly improve the accuracy and robustness on the Visual Object Tracking (VOT) 2014 Challenge dataset.

Contribution:
This work builds upon the same foundation as the channel based learning systems. The work illustrates how this can be used to maintain a distribution field model for tracking. Furthermore, the statistical view of the channel representation is developed, deriving expressions for statistical moments of probability density functions.
CHAPTER 1. INTRODUCTION

represented by channels. The author developed the ideas leading to this publication, implemented the proposed schemes into the evaluation framework, performed the experiments and did the main part of the writing.

1.5.7 Biologically Inspired Online Learning of Visual Autonomous Driving


Abstract:
While autonomously driving systems accumulate more and more sensors as well as highly specialized visual features and engineered solutions, the human visual system provides evidence that visual input and simple low level image features are sufficient for successful driving. In this paper we propose extensions (non-linear update and coherence weighting) to one of the simplest biologically inspired learning schemes (Hebbian learning). We show that this is sufficient for online learning of visual autonomous driving, where the system learns to directly map low level image features to control signals. After the initial training period, the system seamlessly continues autonomously. This extended Hebbian algorithm, qHebb, has constant bounds on time and memory complexity for training and evaluation, independent of the number of training samples presented to the system. Further, the proposed algorithm compares favorably to state of the art engineered batch learning algorithms.

Contribution:
This work presents a novel online multimodal Hebbian associative learning scheme which retains properties of previous associative learning methods while improving performance such that the proposed method compares favorably to state of the art batch learning methods. The author developed the ideas and the extensions of Hebbian learning leading to this publication, implemented the demonstrator system, performed the experiments and did the main part of the writing.

1.5.8 Autonomous Navigation and Sign Detector Learning


Abstract:
This paper presents an autonomous robotic system that incorporates novel Computer Vision, Machine Learning and Data Mining algorithms in order to learn to navigate and discover important visual entities. This is achieved within a Learning from Demonstration (LfD) framework, where policies are derived from example state-to-action mappings. For autonomous navigation, a mapping is learnt from holistic image features (GIST) onto control parameters using Random Forest regression. Additionally, visual entities (road signs e.g. STOP sign) that are
strongly associated to autonomously discovered modes of action (e.g. stopping
behaviour) are discovered through a novel Percept-Action Mining methodology.
The resulting sign detector is learnt without any supervision (no image labeling
or bounding box annotations are used). The complete system is demonstrated
on a fully autonomous robotic platform, featuring a single camera mounted on a
standard remote control car. The robot carries a PC laptop, that performs all the
processing on board and in real-time.

Contribution:
This work presents an integrated system with three main components: learning
visual navigation, learning traffic signs and corresponding actions, and, obstacle
avoidance using monocular structure from motion. All processing is performed
on board on a laptop. The author’s main contributions include integrating the
systems on the intended platform and performing the experiments, the latter in
collaboration with Liam, Nicolas and Johan.

1.6 Related Publications

This section lists publications related to this work.

1.6.1 Online Learning of Autonomous Driving Using Channel Representations of Multi-Modal Joint Distributions

Kristoffer Öfjäll and Michael Felsberg. Online learning of autonomous
driving using channel representations of multi-modal joint distributions. In Proceedings of SSBA, Swedish Symposium on Image Analysis,
March 2015.

Abstract:
An online learning system for many-to-many mappings is presented, where a specific input may map to several different output values. This ability is critical in autonomous driving systems when an obstacle suddenly appears on the road. Given the two options, an evasive maneuver to the left or to the right, a conventional learning system would average these outputs and go straight into the obstacle. Intersections present similar demands and an online learning-from-demonstration autonomous vehicle is presented, capable of handling such scenarios. Learning is based on estimating a non-parametric, multi-modal representation of the joint input-output density. The representation allows real-time learning and prediction onboard the vehicle. Using visual input, most features are unrelated to the output. This reduces contrast in the output distribution and thus reduces system performance. We propose a measure of specificness of inputs with respect to the output, by which the influence of unrelated inputs can be reduced. The proposed learning system is further evaluated on a visual pose estimation dataset and compares favorably to state-of-the-art methods.

Contribution:
This work presents the channel based Hebbian associative learning systems and
its possibilities regarding multiple hypothesis predictions. This work was further extended into [80].

1.6.2 The Visual Object Tracking VOT2014 Challenge Results


Abstract:
The Visual Object Tracking challenge 2014, VOT2014, aims at comparing short-term single-object visual trackers that do not apply pre-learned models of object appearance. Results of 38 trackers are presented. The number of tested trackers makes VOT 2014 the largest benchmark on short-term tracking to date. For each participating tracker, a short description is provided in the appendix. Features of the VOT2014 challenge that go beyond its VOT2013 predecessor are introduced: (i) a new VOT2014 dataset with full annotation of targets by rotated bounding boxes and per-frame attribute, (ii) extensions of the VOT2013 evaluation methodology, (iii) a new unit for tracking speed assessment less dependent on the hardware and (iv) the VOT2014 evaluation toolkit that significantly speeds up execution of experiments. The dataset, the evaluation kit as well as the results are publicly available at the challenge website1.

Contribution:
The author contributed with an improved channel based distribution field tracker [79].

---

1http://votchallenge.net
1.6. RELATED PUBLICATIONS

1.6.3 Online Learning and Mode Switching for Autonomous Driving from Demonstration


Abstract:
Most approaches to learning of vision based autonomous driving either contain predefined parts such as lane marker detectors or require offline processing of training data before the system can run autonomously. Information regarding typical driving behaviour such as staying on roads must be transferred to the system since exploration approaches would be very time and car consuming. We present an autonomous driving system learning a mapping from visual input to control signals, which is trained online by manually steering the robot. After the initial training period, the system seamlessly continues autonomously. Manual control can be taken back at any time for providing additional training.

Contribution:
This publication presents material later combined with [74] to form the book chapter [77].

1.6.4 Rapid Explorative Direct Inverse Kinematics Learning of Relevant Locations for Active Vision


Abstract:
An online method for rapidly learning the inverse kinematics of a redundant robotic arm is presented addressing the special requirements of active vision for visual inspection tasks. The system is initialized with a model covering a small area around the starting position, which is then incrementally extended by exploration. The number of motions during this process is minimized by only exploring configurations required for successful completion of the task at hand. The explored area is automatically extended online and on demand. To achieve this, state of the art methods for learning and numerical optimization are combined in a tight implementation where parts of the learned model, the Jacobians, are used during optimization, resulting in significant synergy effects. In a series of standard experiments, we show that the integrated method performs better than using both methods sequentially.

Contribution:
This work explores the possibilities of using numerical optimization for directing exploration in self-learning systems for inverse kinematics. The work was later extended and combined with related work on learning from demonstration and autonomous driving, forming the book chapter [77].
1.6.5 Integrating Learning and Optimization for Active Vision Inverse Kinematics


Abstract:
An online method for rapidly learning the inverse kinematics of a redundant robotic arm is presented addressing the special requirements of active vision for visual inspection tasks. The system is initialized with a model covering a small area around the starting position, which is then incrementally extended by exploration. The number of motions during this process is minimized by only exploring configurations required for successful completion of the task at hand. The explored area is automatically extended online and on demand. To achieve this, state of the art methods for learning and numerical optimization are combined in a tight implementation where parts of the learned model, the Jacobians, are used during optimization, resulting in significant synergy effects. In a series of standard experiments, we show that the integrated method performs better than using both methods sequentially.

Contribution:
This work was later developed further into [74].

1.6.6 Combining Vision, Machine Learning and Automatic Control to Play the Labyrinth Game


Abstract:
The labyrinth game is a simple yet challenging platform, not only for humans but also for control algorithms and systems. The game is easy to understand but still very hard to master. From a system point of view, the ball behavior is in general easy to model but close to the obstacles there are severe non-linearities. Additionally, the far from flat surface on which the ball rolls provides for changing dynamics depending on the ball position.

The general dynamics of the system can easily be handled by traditional automatic control methods. Taking the obstacles and uneven surface into account would require very detailed models of the system. A simple deterministic control algorithm is combined with a learning control method. The simple control method provides initial training data. As the learning method is trained, the system can learn from the results of its own actions and the performance improves well beyond the performance of the initial controller.

A vision system and image analysis is used to estimate the ball position while a combination of a PID controller and a learning controller based on LWPR is used to learn to steer the ball through the maze.
1.6. RELATED PUBLICATIONS

**Contribution:**
This work presents an evaluation system for control algorithms. A novel learning controller based on LWPR is evaluated and it is shown that the performance of the learning controller can improve beyond the performance of the teacher. The author initiated and developed the ideas leading to this publication, implemented the demonstrator system, performed the experiments and did the main part of the writing.
CHAPTER 1. INTRODUCTION

1.7 Figures

Previously published figures and tables are reproduced with permission of the respective copyright holder. For unlisted figures, either the copyright remains with the author (BMVC and SSBA) or the figures are made specifically for this publication.

1.7.1 Springer

The copyright of the figures and tables from the following publications resides with Springer and are reproduced with permission of Springer.


1.7.2 IEEE

The copyright of the figures and tables from the following publications resides with IEEE and are reproduced with permission of IEEE.


1.7. FIGURES

1.7.3 Frontiers

Figures and tables from the following publications are distributed with a Creative Commons attribution license.

Part I

Representation
Chapter 2

Introduction to Representations

Operating on anything else than binary digits, representation is a fundamental issue in every case of information processing using digital computers. Signals cannot be directly acted upon by such devices but must be represented using a set of binary digits. For fundamental mathematical objects such as integers and real numbers, there are widely adopted standard representations such as IEEE 754 for floating point numbers. However, already at basic geometric objects, such as lines, there are several different representations in common use.

Most representations are interchangeable in the sense of the original object being reconstructable from each representation. Although two different representations may contain the same information regarding the represented object, there are differences in other aspects, such as suitability and ease of use in operations on the represented objects (e.g., computational and memory complexity).

Consider the line example, for drawing, a representation using a point and a direction is preferred [13]. On the other hand, for finding the intersection with another line, a representing equation is preferred [42], an equation whose solutions provide points on the line. Using a geometric algebra, a line is represented by a single entity [51], similar to the use of quaternions for representing rotations in 3D.

There are also representations from which the original object cannot be reconstructed. Such representations may be useful where the lost information is of no use to the system and thus the remaining entity has a higher concentration of useful information, in the best case in a more easily accessible form. The representation or system is said to be invariant to such changes that constitute the lost information. A useful representation should retain as much of the desired information as possible while being invariant to noise and undesired information. Furthermore, the representation should be computationally easy to handle.

It is clear that the suitability of any particular representation is dependent on the task at hand, and, most problems can be made easier (or harder) by choosing a suitable way of representing information. In cases where there exists a perfect
representation, machine learning can be as simple as finding a threshold. If the task of the system is to control the temperature of a room, such a representation would be to represent the current room state with a number proportionally related to the room temperature.

For an autonomously driving vehicle, most information in the camera sensor data is irrelevant for the operation of the system. Only the road and objects on and close to the road are of interest. As of today, the author is not aware of any driver assistance system notifying the passengers e.g. when passing by an exceptionally beautiful landscape.

For robotics and vision, the objects to represent usually originate from the surrounding world and have to be measured. This includes almost any imaginable entity such as the local air temperature or the incoming radiation within the visible spectrum from different directions. Within computer vision, properties of objects are sought by observing solely the radiation reflected off the object towards a single point, where the camera is located.

For certain physical entities, there are common sensors for measuring these. Such as using a thermometer to measure temperature. However, there are also reasons for determining such entities using different types of sensors. Such sensors may be cheaper or already available, or there may be an effort to reduce the total number of sensors in a system.

Although seeming a bit strange, the computer vision community will not be surprised of hearing about any attempt to determine room temperature by visually observing appearance and behavior of people in the room. After all, humans tend to be able to estimate the outdoor temperature by looking out a window. Furthermore, a control system using such an approach for measuring room temperature would be able to directly keep the temperature at a comfortable level for the room occupants, not requiring the occupants to set an absolute desired temperature.

For the representation, the measured or estimated values themselves may not be sufficient, often, an estimate of the certainty of the estimated value is valuable. A conventional thermometer may provide more accurate temperature measurements than the vision-based thermometer described above. Any book on estimation theory, e.g. [107], points out the importance of affixing to any estimate of an entity, an estimate of the confidence of said estimate. One example of such a value-confidence pair is the local image orientation representation by complex numbers, where the argument represents the orientation and the magnitude represents the confidence.

Taking this one step further, the entity may be represented by a posterior distribution: an estimation of the probability of different values of the entity, given one or more measurements and possibly also taking into account other measurements providing indirect information regarding the sought entity. This requires an efficient representation of density functions. Such a representation is presented in chapter 3.
2.1 Biological Visual Representation

In biological vision systems of mammals, light intensities from different directions are measured using photosensitive cells in the retina. However, a different representation is used when transferring image information over the optic nerve [60]. The information represented in the nerve is just a small fraction of what is available at the retina. Nonetheless, it is apparently sufficient for the needs of the animal.

David Hubel and Torsten Wiesel explored the information processing in the first stages of the visual cortex. They recorded signals from single neurons in the visual cortex of cats and macaques among other animals [46]. Cells were found reacting to visual structures such as edges. The response depended on the orientation of the edge relative to the so called preferred orientation of each such cell. Also, the response faded gradually as the edge orientation was rotated further away from the preferred orientation of the cell.

This smoothly fading response, the tuning curve, can be utilized such that given a few cells with different preferred orientations, the true orientation of the edge can be recovered with greater precision than the number of cells, by comparing relative response strengths of the different cells. Within computational neuroscience, such a representation is known as population coding. One approach for decoding population codes with equally spaced preferred directions was presented by Denève et al. [23]. However, an implementation of their proposed scheme has shown to bias the decoded value towards preferred orientations or orientations between two preferred orientations [33]. The channel representation presented in chapter 3 can be interpreted as population coding, however, the channel representation is based on a more solid mathematical foundation which allows for a deeper analysis of representation properties.

2.2 Visual Information Representation in Computer Systems

The conventional representation of visual information, camera images, is an orthogonal grid of intensity measurements. These stem from a regular grid of light sensitive sensors in the camera. While this representation is tightly connected to the measurement hardware and suitable for reconstructing the light pattern on a computer screen, it may not be the best representation for extracting whatever desired information from the image that is required by a computer vision system.

The suitability of any image representation depends on what information is to be extracted, which depends on the application. This has led to the development of a large collection of different feature extractors and descriptors such as SIFT [69]), HOG [19], COSFIRE [2], and deep features [63] to name a few.

In addition to the different image representations, there are also different camera sensors emerging, such as the event camera. Instead of providing an absolute measure of light intensity in each pixel at a rate of 25 to 30 Hz, these sensors send an impulse whenever the intensity at a certain pixel have increased or decreased a
certain amount since the last impulse triggered by that pixel \cite{91, 53}. These impulses are sent with significantly better time resolution compared to conventional cameras.

Non-orthogonal sampling patterns are also considered. Mammal retinas have a higher density of light sensitive cells at the center of the visual field, the fovea. Similar ideas are used in e.g. object recognition where interesting (salient) areas are detected in a low resolution image and a high-resolution camera is directed towards the salient areas \cite{110}. Other approaches are using different sampling patterns which are still regular, such as hexagonal grids \cite{67}.

There are also examples of performing image processing directly on light itself, such as generating light using a projector and re-capturing the processed light with a camera. A simple case is setting camera focus a bit off instead of an initial software low pass filter in the processing chain \cite{72}.

Features and representations can also be learned. Using principal component analysis for an initial reduction of dimensionality has been popular \cite{105, 109}. In recent years deep learning, and especially convolutional neural networks \cite{61}, have regained much attention. In this layered approach, the first layers can be interpreted as feature extractors and thus changing the representation of the image information prior to the higher levels. The higher levels constitute a conventional artificial neural network which performs the specific task such as classification or regression.

Training a full convolutional neural network is resource demanding, requiring several days of computation time on GPU clusters \cite{59}. Pre-trained lower levels are available, trained for different purposes \cite{48, 103}, again separating calculating a new representation and using the new representation of the image to solve a particular task. Examining these lower levels of the convolutional networks, similarities to the first levels of biological vision systems have been found \cite{112}. 
Chapter 3

The Channel Representation

In this chapter the channel representation is introduced. The first section presents different basis functions and the encoding operation. This sets the stage for the later sections which present contributions to the field. The main part of this chapter is based on material from two publications [33, 79].

Figure 3.1: Examples of basis functions for the channel representation with width and overlap 3, that is, for each value \( \xi \), three basis functions are non-zero. (a) A single basis function. (b) Eight basis functions for representing values in the interval \( 0.5 \leq \xi \leq 6.5 \). The first and the last basis functions have their strongest response outside the interval.

The channel representation stems from a biologically inspired information representation [37, 101]. A statistical view and with that, a more solid mathematical
The central idea is to represent a value by its activation pattern of different channels placed along the possible range of values. The strength of the response of each channel is determined by the distance from the preferred value of the channel and a function describing the response decay for values not exactly at the preferred value of the channel. This function is known as the channel basis function, tuning curve or sometimes kernel. See figure 3.1 for one example of basis functions with overlap three.\footnote{Each value $\xi$ within the representable range is within the support of three basis functions, except at channel bounds where two basis functions are non-zero and equal.}

Several overlapping basis functions are distributed in the space of values. A specific value will activate some of these channels when encoded. The channel vector is the concatenated list of the activations for each channel. If rectangular, non-overlapping basis functions are used, the resulting channel vector is a histogram. However, using certain smooth and overlapping basis functions, a single encoded value can be recovered with arbitrarily high precision. For histograms, the precision is limited by the bin width.

An overview of other applications of channel-based image processing is provided in a survey \cite{28}. The channel representation shares some similarities with averaged shifted histograms \cite{98} and distribution fields \cite{100,79}.

In order to enable application of discrete signal processing methods, the basis functions are usually placed in a regular grid. In this chapter, channels are assumed to be centered on integer positions along the real number line. This can be assumed without loss of generality as any finite range of input values can be scaled and translated to the representable range of the channel representation. Removing the scaling and translation parameters from all expressions clarifies the presentation.

3.1 Introduction to the Channel Representation

Before going into all mathematical details of the channel representation, a short intuitive introduction is presented here. Consider the basis functions in figure 3.1b. The eight basis functions are centered at integers from 0 to 7. The width of the support of each basis function is three. Given a value $\xi$ within the representable range from 0.5 to 6.5, the corresponding channel vector $x$ is obtained as a list of the function values of all the basis functions.

Let $\xi = 2$, the corresponding channel vector is illustrated in figure 3.2a as scaled basis functions. Their point-wise sum is plotted as a dashed line with a maximum at $\xi = 2$, indicated by a vertical dashed line. Since the width of the support of the basis functions is three, at most three elements in the channel vector are non-zero. In this case, using un-normalized $\cos^2$ channels, $x = [0 \ 0.25 \ 1.00 \ 0.25 \ 0 \ 0 \ 0 \ 0]^T$.

Let $\xi = 2 + \pi$, a value neither on a channel center nor right between two. The corresponding channel vector $x \approx [0 \ 0 \ 0 \ 0.13 \ 0.98 \ 0.39 \ 0]^T$ is illustrated in figure 3.2b. Note that the point-wise sum of scaled basis functions has its
3.1. INTRODUCTION TO THE CHANNEL REPRESENTATION

In both cases, the sum of the elements of the channel vector is 1.5. This is not a coincidence but a property of certain basis functions: the sum of all channel coefficients is independent of the encoded value, as long as the value is within the representable range. For such representations, the certainty or evidence of a specific value may be encoded by scaling the channel vector. That is, both value and certainty is contained in the same entity. At the extreme, an all-zero channel vector indicates that no information is available.

The representation allows multiple values, by adding the two channel vectors, \( \mathbf{x} \approx [0 \ 0.25 \ 1.00 \ 0.25 \ 0.13 \ 0.98 \ 0.39 \ 0]^T \) is obtained. It is illustrated in figure 3.2c, scaled by 0.5 to maintain the sum at 1.5. The same vector is illustrated as a stem plot in figure 3.2d. This vector represents both values 2 and \( 2 + \pi \). If one value is more certain or common than the other, a weighted combination can be used. Note the difference from representing the two values using e.g. their mean.

Certainly, more vectors can be combined. In figure 3.3, samples are drawn from two normal distributions with equal chance of selecting either distribution. Such samples are channel encoded and the mean channel vector is obtained. The vector is illustrated as the stem plot in figure 3.3, the solid line is the density function of the distribution of the samples and the dashed line is the weighted point-wise sum of the basis functions. In this case, normalized basis functions are used, such that they represent valid density functions. The channel vector is related to the sampled kernel density estimate of the distribution.

![Figure 3.2: Illustrations of channel representations using weighted basis functions or a stem plot. In (a) and (b), a single value, 2 and \( 2 + \pi \) respectively, is encoded. In (c), both values are represented in the same channel vector. Plot (d) is another illustration of the same combined channel vector.](image-url)
3.2 Basis Functions and Encoding

Several different shapes of the basis function have been proposed and evaluated [35]. All proposed basis functions and tuning curves have a bell shape, matching the results of measurements on biological systems [46]. However, measurement noise does not allow measuring the exact shape of the biological basis functions. Fortunately, the precise shape of the basis function does not notably affect system performance when used solely as a biologically inspired representation [23].

For other purposes, such as decoding and representing distributions, the shape of the basis function does matter. In such cases, a suitable shape can be found by postulating a series of mathematical requirements on the basis function. The decoding procedure depends on the precise shape and is presented in chapter 4.

Fundamental requirements on the basis functions \( b(\xi) \) are

\[
\begin{align*}
  b(\xi) &\geq 0 \quad \xi \in \mathbb{R} \quad (3.1) \\
  b(-\xi) &= b(\xi) \quad \xi \in \mathbb{R} \quad (3.2) \\
  \exists K > 0 \text{ such that } |b(\xi) - b(\xi + \epsilon)| &\leq K|\epsilon| \quad \xi, \epsilon \in \mathbb{R} \quad (3.3) \\
  b(\xi) &= 0 \quad |\xi| \geq w/2 \quad (3.4) \\
  b(\xi) &> b(\xi + \epsilon) \quad 0 < \xi < w/2, \quad \epsilon > 0 \quad (3.5) \\
  \text{arg} ((\mathcal{F}b)(\theta)) &= 0 \quad \theta \in \mathbb{R} \quad (3.6) \\
  \|C(\xi)\|_1 &= K_1 \quad \xi \in \mathbb{A} \quad (3.7) \\
  \|C(\xi)\|_2 &= K_2 \quad \xi \in \mathbb{A} \quad , \quad (3.8)
\end{align*}
\]

where \( w \) is the width of the basis function and \( \mathbb{A} \) is the representable range. The last two requirements, (3.7) and (3.8), are included only for the sake of completeness as the channel encoding operator \( C(\cdot) \) will not be introduced until section 3.2.1.
Furthermore, these requirements are not fulfilled by all proposed basis functions. Nonetheless, the requirements indicate that the basis function shall be such that the channel encodings of a value $\xi$ should have constant $L_1$ and $L_2$ norms for all values within the representable range $A$.

Among the direct requirements on the basis function, non-negativity (3.1), symmetry (3.2) and Lipschitz continuity (3.3) stems from the biological background. Cells cannot respond with less than no activity. However, other mechanisms may allow cell activity to inhibit activity in other cells. Compact support (3.4) is required for computational efficiency, which is related to energy consumption in biological systems. Strict monotonicity (3.5) on $0 < \xi < w/2$ is required for unambiguous decoding.

The basis function is not allowed to introduce spatial translations. This is expressed by interpreting the basis function as the impulse response of a zero phase linear filter, leading to requirement (3.6), where $\mathcal{F}$ denotes the Fourier transform. Later the relation of the channel representation to kernel density estimation will be explored. The channel vector can then be interpreted as a sampled, filtered density function. The basis function will be the impulse response of the filter. The list of requirements is not minimal, e.g. symmetry (3.2) follows from zero phase (3.6) and e.g. non-negativity (3.1) follows from (3.3), (3.4) and (3.5).

The number of simultaneously active channels, $w$, can be chosen arbitrarily from an encoding point of view, however, three overlapping channels has been a de-facto standard choice [49], which is motivated by the decoding [35]. From the assumption of integer channel centers follows $w = 3$. Section 3.3 presents a proof regarding minimal channel overlap.

### 3.2.1 Channel Encoding

The general encoding procedure is independent of the shape of the basis function. First, channel encoding of scalar values $\xi$ is presented. Encoding of vectors is primarily the matter of combining the channel vectors of each encoded element in the vector to be encoded.

Using $N$ channels, the encoding of a scalar $\xi$ is a vector of channel coefficients

$$x = C(\xi) = [b(\xi - 1), b(\xi - 2), b(\xi - 3), \ldots, b(\xi - N)]^T,$$

(3.9)

that is, each component of the channel vector is $x_n = b_n(\xi)$, using the shifted basis function notation $b_n(\xi) = b(\xi - n)$ for $n = 1, 2, \ldots, N$. The valid representable range, where there are three overlapping channels, is

$$A = \{\xi \in \mathbb{R} : 1.5 < \xi < N - 0.5\}.$$  

(3.10)

Within this range, an encoded value can be perfectly recovered in the noise free case. For any other choice of overlap $w$, the representable range is different. Note that the basis functions are required to be non-negative and thus all channel coefficients in $x$ will be non-negative.

In this case, and in most cases hereafter, the first channel is centered around 1 as this simplifies notation. The choice is arbitrary of course and without loss of
Figure 3.4: (a) Illustration of a channel vector as weighted basis functions. The encoded value, 3.2, is marked with a dashed line. (b) Illustration of eight basis functions on a modular domain, a circle in this case. One basis function is marked with a bold line.

generality. In figure 3.1, the first basis function is centered around 0 which would introduce a constant offset in notation: \( b_n(\xi) = b(\xi - (n - 1)) \).

For representing modular domains, such as orientations, the edge channels are wrapped around such that the first two channels are the same as the last two. On a modular domain, the choice of the first channel is of course arbitrary. Figure 3.4 illustrates a single channel encoded value and channels on a modular domain. In applications, the number of channels, \( N \), is first selected depending on the expected noise and characteristics of the values, \( \xi \), to be encoded. The domain of \( \xi \) is then transformed to the valid encoder range.

3.2.2 Channel Encoding Vectors

Two different approaches are available for encoding multi-dimensional data. Both approaches encode each dimension separately. The difference lies in how these separate channel vectors are combined [39].

Consider a two-dimensional point \( \xi = [\xi_1, \xi_2]^T \) and the channel vector of each encoded element \( x_1 = C(\xi_1) \) and \( x_2 = C(\xi_2) \) using \( N_1 \) and \( N_2 \) channels respectively. If the separate dimensions are independent, the two channel vectors are concatenated

\[
x = C(\xi) = [C(\xi_1)^T \ C(\xi_2)^T]^T = [x_1^T \ x_2^T]^T , \tag{3.11}
\]

resulting in a \( N_1 + N_2 \) dimensional channel vector \( x \). If independency cannot be
assumed, a Kronecker product is used,
\[ x = C(\xi) = C(\xi_1) \otimes C(\xi_2) = x_1 \otimes x_2 , \]  
resulting in a \( N_1 N_2 \)-dimensional channel vector. The Kronecker product corresponds to an outer product followed by vectorization, \( x = \text{vec}(x_2 x_1^T) \). This corresponds to placing multi-dimensional channels in an axis aligned grid.

This is motivated by a probabilistic view on the channel representation. Consider \( \xi = [\xi_1, \xi_2]^T \) as a sample drawn from a distribution with joint density \( f_{\xi_1,\xi_2}(\xi_1, \xi_2) \) and marginal densities \( f_{\xi_1}(\xi_1) \) and \( f_{\xi_2}(\xi_2) \). If the two dimensions are independent, the joint distribution is separable and can be obtained from the marginal distributions, \( f_{\xi_1,\xi_2}(\xi_1, \xi_2) = f_{\xi_1}(\xi_1)f_{\xi_2}(\xi_2) \). Similarly, it is sufficient to store the separate channel vectors \( x_1 \) and \( x_2 \).

On the other hand, if there is a dependency, the joint distribution cannot be obtained from the marginal distributions and this is reflected in the channel encoding as a need to store the full Kronecker product (or vectorized outer product). There is a relation between sums of outer products of channel vectors \( x_2 x_1^T \) and joint distributions. This is discussed further in section 3.4 on the statistical interpretation of the channel representation.

For higher-dimensional data, both approaches can obviously be extended forming longer channel vectors \( x = [x_1^T, x_2^T, \ldots, x_n^T]^T \) or \( x = x_1 \otimes x_2 \otimes \ldots \otimes x_n \), encoding \( n \)-dimensional data. Furthermore, dimensions can be grouped and combined using both methods in any combination that fits the problem at hand. Assume two dimensions to be dependent and the remaining dimensions mutually independent and independent of this two-dimensional subspace. A suitable encoding is to combine the two mutually dependent dimensions using the vectorized outer product and concatenating the independently encoded channel vectors of the remaining dimensions with the channel vector of the two-dimensional space.

The drawback of the joint channel representation is the exponential growth of the number of coefficients with the number of dimensions. This is mitigated by sparse data structures and the finite support of the channel basis function, as the number of non-zero elements are at most three to the power of the dimensionality of \( \xi \).

Encoding and decoding of single values is of no greater use. Only when several channel vectors are combined does the advantages of the representation appear. An example is presented in section 3.4. First, a closer look on different basis functions and their properties follows.

### 3.2.3 Basis Functions and Channel Vector Invariances

Different basis functions have been evaluated such as \( \cos^2 \), B-splines and Gaussian kernels [35]. The Gaussian kernel however, lacks compact support, leading to dense channel vectors.

Basis functions has been evaluated with respect to properties such as encoding and decoding complexity. One significant set of properties are the (in)variances of the channel vector with respect to the relative positions of encoded values to the channel centers. Forssén performed a series of experiments along these lines for
different basis functions \[35\]. For the present work, invariances of the \(L_1\) and \(L_2\) norms of the channel vector have had most impact.

The lack of this type of analysis is the main deficiency of population coding frameworks stemming solely from a neuroscientific background. The dependency on the relative position of encoded values with respect to basis function centers leads to biased results where the bias is dependent on the channel grid \[33\].

In the present work, truncated \(\cos^2\) basis functions

\[
b(\xi) = \begin{cases} 
\cos^2\left(\frac{\pi \xi}{w}\right) & |\xi| < \frac{w}{2} \\
0 & |\xi| \geq \frac{w}{2} 
\end{cases}
\]  

(3.13)

are used, with \(w = 3\), assuming integer spacing and three overlapping basis functions. The channel representation then obtains both \(L_1\) and \(L_2\) invariance. That is \(\|C(\xi)\|_1 = \frac{3}{2}\) and \(\|C(\xi)\|_2 = \frac{3}{\sqrt{2}}\) for all \(\xi\) within the representable range. An example of \(\cos^2\) basis functions is presented in Fig. 3.1.

Channel representations based on Gaussian basis functions, such as is used in the well know radial basis functions, lacks compact support as well as both types of norm invariance properties. Using B-spline basis functions, the channel representation has \(L_1\) invariance but not \(L_2\) invariance \[35\]. The invariants are essential for determining properties of sums of channel vectors. The \(L_1\) norm is related to evidence, that is, the total number of samples in the sum or within a particular range. The \(L_2\) norm is related to coherence among the samples in a mode. This is further analyzed in section 4.3 with respect to the decoding of the channel representation.

In this work, two differently scaled versions of the \(\cos^2\) basis functions will appear. By including constant scale factors in the definition of the basis function, expressions and proofs containing these will be more clear. The latter, mass preserving basis function will be the most commonly used.

The unscaled \(\cos^2\) basis function (3.13) is

\[
b(\xi) = \begin{cases} 
\cos^2\left(\frac{\pi \xi}{3}\right) & |\xi| < \frac{3}{2} \\
0 & |\xi| \geq \frac{3}{2} 
\end{cases}
\]  

(3.14)

For the mass-preserving \(\cos^2\) basis function

\[
b(\xi) = \frac{2}{3} \begin{cases} 
\cos^2\left(\frac{\pi \xi}{3}\right) & |\xi| < \frac{3}{2} \\
0 & |\xi| \geq \frac{3}{2} 
\end{cases}
\]  

(3.15)

the channel representation obtains the properties

\[
\|C(\xi)\|_1 = 1 \tag{3.16}
\]

and

\[
\|C(\xi)\|_2 = \frac{1}{\sqrt{2}} \tag{3.17}
\]

for all \(\xi\) within the representable range. Furthermore,

\[
\int_{-\infty}^{\infty} b(\xi) \, d\xi = 1 \tag{3.18}
\]
For future reference, the derivative of the basis function is
\[
\frac{db}{d\xi} = \begin{cases} 
-\frac{2\pi}{9}\sin\left(\frac{2\pi \xi}{3}\right) & |\xi| < \frac{3}{4} \\
0 & |\xi| \geq \frac{3}{4}
\end{cases}.
\] (3.19)

### 3.3 Uniqueness of the Cosine Basis Function

This section presents recent theoretical results regarding the \(\cos^2\) channel representation, providing a theoretical motivation for the previously de-facto choice of three overlapping channels. Furthermore, the uniqueness of the \(\cos^2\) basis function is shown. The main part of this section is an adaptation of parts of the journal article by Felsberg, Öfjäll and Lenz [33].

![Figure 3.5: (a) Three mass-preserving \(\cos^2\) channel basis functions on a modular domain of length 3. (b) The basis functions on a circle.](image)

Consider a channel representation using three channels on a modular domain as in figure 3.5 and the corresponding channel vector interpreted as a point in \(\mathbb{R}^3\),
\[
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{x} = \mathcal{C}(\xi)
\] (3.20)

while \(\xi\) moves along all representable values. The constant \(L_1\) norm property indicates that all points lie in the plane \(x_1 + x_2 + x_3 = 1\) and corresponds to the idea of constant probability mass, i.e., each observation provides the same amount of evidence, independently of the specific value.

The constant \(L_2\) norm property indicates that all points lie on the sphere \(x_1^2 + x_2^2 + x_3^2 = \frac{1}{2}\), i.e. all points lie on a circle in the plane \(x_1 + x_2 + x_3 = 1\) with
center in $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$. Furthermore, it can be shown that the length of the tangent vector $\|dC(\xi)\|_2$ is constant. Together this corresponds to the idea of isometry, i.e. scalar products $C(\xi_1)^T C(\xi_2)$ depend only on the relative angle between $\xi_1$ and $\xi_2$, not on their absolute angles within the reference coordinate system.

For B-spline and Gaussian basis functions, the geometric result is not a circle and thus the scalar product depends on the placement of the channel centers. Channel representations using more than three channels still possess the $L_1$ and $L_2$ norm constancy properties, however, the geometry gets more involved. That is the subject of section 4.2 on geometric interpretations of channel decoding.

3.3.1 Proof of the Minimal Overlap Case

The cos$^2$ basis functions with the choice of $w = 3$, have an overlap of three. Thus, for a single observation $\xi$, the channel vector $x$ has at most three non-zero coefficients

$$x = C(\xi) = (\ldots, 0, b(\xi - \lfloor \xi \rfloor + 1), b(\xi - \lfloor \xi \rfloor), b(\xi - \lfloor \xi \rfloor - 1), 0, \ldots)^T,$$

where $\lfloor \xi \rfloor$ is the closest integer to $\xi$. Denote the three nonzero coefficients $x_1, x_2$ and $x_3$. Together with the requirements of kernel functions, (non-negativity, compactness, and Lipschitz-continuity), it can be shown that no kernels exist with overlap smaller than three for which the corresponding channel representation possesses both norm invariance properties:

Assume overlap 1, i.e. disjoint support of the basis functions. $\|C(\xi)\|_1 = 1$ implies constant activation in the support of the basis function, thus it is not Lipschitz continuous at the boundary.

Assume overlap 2. Let, without loss of generality, the second coefficient be $0 < x_2 \leq 1$. Overlap 2 implies that at most one of $x_1$ and $x_3$ is non-zero. Assume, without loss of generality, $x_1 = 0$ and thus $\|C(\xi)\|_1 = 1$ implies $x_3 = 1 - x_2$. For constant $L_2$ norm $\mu$,

$$\mu^2 = \|C(\xi)\|_2^2 = x_2^2 + x_3^2 = x_2^2 + (1 - 2x_2)^2 = 2x_2^2 + 1 - 2x_2$$

(3.22)
is obtained and therefore $x_2 = \frac{1}{2} \pm \frac{1}{2} \sqrt{2\mu^2 - 1}$ attains either of two discrete values and the basis function is therefore not Lipschitz continuous at the boundary.

Assume overlap 3. Without loss of generality, $0 < x_1, x_2, x_3 \leq 1$. From the two norm constraints

$$x_1^2 + x_2^2 + (1 - x_1 - x_2)^2 = \mu^2$$

(3.23)
is obtained and thus

$$x_2 = \frac{1}{2} (1 - x_1 \pm \sqrt{2\mu^2 - 1 + 2x_1 - 3x_1^2})$$

(3.24)

Note that $x_3$ takes the respectively other solution. Lipschitz continuity at the interval boundary requires that one of $x_1, x_2, x_3$ tends to zero. In that case, the
3.3. UNIQUENESS OF THE COSINE BASIS FUNCTION

sum of the remaining two coefficients tends to one and the larger coefficient is at least \( \frac{1}{2} \). Requiring \( x_2 \geq 0 \) for \( x_1 = 0 \) implies \( \mu^2 \geq \frac{1}{2} \). Requiring \( 0 \leq x_2 \leq (1 - \frac{1}{2}) \) for \( x_1 = \frac{1}{2} \) implies \( \mu^2 \leq \frac{1}{2} \). Thus \( \mu^2 = \frac{1}{2} \) and

\[
x_2 = \frac{1}{2}(1 - x_1 \pm \sqrt{2x_1 - 3x_1^2}) \quad x_3 = \frac{1}{2}(1 - x_1 \mp \sqrt{2x_1 - 3x_1^2})
\]

(3.25)

which establish implicitly continuous real mappings \( \xi \mapsto (x_1, x_2, x_3) \in [0, \frac{2}{3}]^3 \).

Thus, the minimal overlap of any basis function is three if the requirements and norm invariance properties shall be fulfilled.

3.3.2 Proof of the Uniqueness of the \( \cos^2 \) Basis Functions

From section 3.3.1 it is known that the minimum overlap is three. This implies that for integer sampling, the support of the basis function is \( \xi \in [-\frac{3}{2}, \frac{3}{2}] \). Since the basis function is Lipschitz continuous, the Fourier series on \([-\frac{3}{2}, \frac{3}{2}]\) converges uniformly. Thus

\[
b(\xi) = \sum_{k=-\infty}^{\infty} B_k \exp(ik\xi \frac{2\pi}{3})
\]

(3.26)

for suitable \( B_k \). Since the basis function is real, \( B_{-k} = B_k^* \) and combined with \( B_k \geq 0 \), i.e. (3.6), it follows that \( B_{-k} = B_k \). Thus, \( b \) is a cosine series with coefficients \( 2B_k, k > 0 \)

\[
b(\xi) = B_0 + \sum_{k=1}^{\infty} 2B_k \cos(k\xi \frac{2\pi}{3})
\]

(3.27)

Consider the \( L_1 \) norm of a decoding window \(-\frac{1}{2} \leq \xi \leq \frac{1}{2}\), since \( b(\xi) \geq 0 \) for all \( \xi \),

\[
\left\| \begin{pmatrix} b(\xi + 1) \\ b(\xi) \\ b(\xi - 1) \end{pmatrix} \right\|_1 = b(\xi + 1) + b(\xi) + b(\xi - 1) = 3B_0 + \sum_{k=1}^{\infty} 2B_k \sum_{n=-1}^{1} \cos(k(\xi + n) \frac{2\pi}{3})
\]

(3.28)

where the inner sum is zero for \( k = 1, 2, 4, 5, 7, \ldots \) due to the uniform spacing of the points around the circle. However, for \( k = 3m \) for \( m \in \mathbb{Z}^+ \), the inner sum is

\[
\sum_{n=-1}^{1} \cos(2\pi m\xi + 2\pi mn) = 3 \cos(2\pi m\xi)
\]

(3.29)

which is dependent on \( \xi \). The cosine functions \( \cos(2\pi m\xi) \) are linearly independent\(^2\) and thus \( B_{3m} = 0 \) for all \( m \in \mathbb{Z}^+ \) is required for the \( L_1 \) norm to be constant. The \( L_1 \) norm is thus \( 3B_0 \).

\(^2\)The cosine functions are even orthogonal, i.e. \( \int_{-\frac{3}{2}}^{\frac{3}{2}} \cos(2\pi m\xi) \cos(2\pi n\xi) \, d\xi = 0 \) for all positive integers \( m \neq n \).
Consider the squared $L_2$ norm

$$b(\xi + 1)^2 + b(\xi)^2 + b(\xi - 1)^2 = \sum_{n=-1}^{1} \left[ B_0 + \sum_{k=1}^{\infty} 2B_k \cos(k(\xi + n)\frac{2\pi}{3}) \right]^2 =$$

$$= 3B_0^2 + \sum_{n=-1}^{1} 4B_0 \sum_{k=1}^{\infty} B_k \cos(k(\xi + n)\frac{2\pi}{3}) +$$

$$+ \sum_{k=1}^{\infty} 4B_k^2 \sum_{n=-1}^{1} \cos^2(k(\xi + n)\frac{2\pi}{3}) +$$

$$+ \sum_{l=1}^{\infty} \sum_{k=1}^{l-1} \sum_{n=-1}^{1} 8B_k B_l \cos(k(\xi + n)\frac{2\pi}{3}) \cos(l(\xi + n)\frac{2\pi}{3}) \quad (3.30)$$

The first term $3B_0^2$ is constant. The second term is similar to the expression for the $L_1$ norm and will be zero for all cases when the $L_1$ norm is constant. The inner sum of the third term

$$\frac{1}{2} \sum_{n=-1}^{1} \cos^2(k(\xi + n)\frac{2\pi}{3}) = \frac{1}{2} \sum_{n=-1}^{1} \left[ \cos(0) + \cos(2k(\xi + n)\frac{2\pi}{3}) \right] =$$

$$= \frac{3}{2} + \frac{1}{2} \sum_{n=-1}^{1} \cos(2k(\xi + n)\frac{2\pi}{3}) \quad (3.31)$$

is, similar to the $L_1$ norm case, constant for $2k \neq 3m$ for $m \in \mathbb{Z}^+$. Furthermore, there are integers $m_1$, $m_2$ such that $2k = 3m_1 \Leftrightarrow k = 3m_2$ for all $k \in \mathbb{Z}^+$. I.e. all non-constant factors correspond to zero coefficients $B_{3m}$ and thus vanish. The third term of (3.30) thus reduces to

$$\sum_{k=1}^{\infty} 6B_k^2 \quad (3.32)$$

All terms except the last one are thus constant with respect to $\xi$.

The inner sum of the last term of (3.30) can be reformulated into

$$\sum_{n=-1}^{1} 8B_k B_l \cos(k(\xi + n)\frac{2\pi}{3}) \cos(l(\xi + n)\frac{2\pi}{3}) =$$

$$= 4B_k B_l \left[ \sum_{n=-1}^{1} \cos((k - l)(\xi + n)\frac{2\pi}{3}) + \sum_{n=-1}^{1} \cos((k + l)(\xi + n)\frac{2\pi}{3}) \right] \quad (3.33)$$

The structure from (3.28) reappears again, in this case however, the first sum vanishes when $k - l \neq 3m$ and the second sum when $k + l \neq 3m$ for $m \in \mathbb{Z}^+$. Note that the indexing in (3.30) guarantees $k > l$. Furthermore, all coefficients $B_k$ and
3.4. STATISTICAL INTERPRETATION

$B_l$, where $k$ or $l$ is a multiple of three, are required to be zero, and thus any such term vanishes. This leaves two sets of expressions where either $k - l$ or $k + l$ is a multiple of three and where $l$ and $k$ are not multiples of three. Rearrange the non-vanishing terms of the last term of (3.30) by $k - l = 3m$ and $k + l = 3m$ into

$$
\sum_{m=1}^{\infty} \left[ \sum_{l \neq 3, 6, 9, \ldots}^{\infty} 4B_l B_{l+3m} \sum_{n=-1}^{1} \cos(3m(\xi + n)\frac{2\pi}{3}) + \sum_{l \neq 3, 6, 9, \ldots}^{3m-1} 4B_l B_{3m-l} \sum_{n=-1}^{1} \cos(3m(\xi + n)\frac{2\pi}{3}) \right] = 
$$

$$
= \sum_{m=1}^{\infty} \left[ \sum_{l \neq 3, 6, 9, \ldots}^{\infty} 4B_l B_{l+3m} 3 \cos(2\pi m\xi) + \sum_{l \neq 3, 6, 9, \ldots}^{3m-1} 4B_l B_{3m-l} 3 \cos(2\pi m\xi) \right] = 
$$

$$
= \sum_{m=1}^{\infty} 12 \left[ \sum_{l \neq 3, 6, 9, \ldots}^{\infty} B_l B_{l+3m} + \sum_{l \neq 3, 6, 9, \ldots}^{3m-1} B_l B_{3m-l} \right] \cos(2\pi m\xi) \quad (3.34)
$$

Thus, for constant $L_2$ norm, the expression needs to be constant. Due to the linear independence of the $\cos(2\pi m\xi)$ functions for different $m$, the constancy requirement reduces to a set of equations

$$
\sum_{l \neq 3, 6, 9, \ldots}^{\infty} B_l B_{l+3m} + \sum_{l \neq 3, 6, 9, \ldots}^{3m-1} B_l B_{3m-l} = 0 \quad (3.35)
$$

for $m = 1, 2, \ldots$. Every possible pair $B_l B_k$ appears exactly once in these equations, the appearing terms are illustrated in figure 3.6. Since $B_k \geq 0$ at most one $B_k$, $k > 0$ can be non-zero. Furthermore, only $B_1$ corresponds to a monotonically decreasing function on the interval $0 < \xi < \frac{3}{2}$. The requirements of limited support and Lipschitz continuity at the support boundaries determines the ratio of $B_0$ and $B_1$. The remaining degree of freedom is the overall scaling of the basis function.

### 3.4 Statistical Interpretation

The stricter requirements on the basis functions allows for a more mathematical treatment of the channel representation and channel encoding. Statistical applications are close at hand and in this section such results are presented. The core results of this section are adapted and expanded from Öfjäll and Felsberg [79].
Figure 3.6: Coefficient pairs appearing in each equation (3.35) for the eight first possibly non-zero coefficients $B_k$. The numbers in the table correspond to the equation number $m$. For a finite number of coefficients, the non-negativity constraint is not necessary, the structure of the equations then allow only one non-zero $B_k$.

3.4.1 Channel Encoding as a Kernel Density Estimate

Until now, only single, but possibly multi-dimensional, values have been encoded. However, the full benefits of the channel representation only appear when combining multiple encoded values of the same entity. The combination operation is usually the mean vector of several encoded measurements $C(\xi_1), \ldots, C(\xi_I)$:

$$\bar{x} = \frac{1}{I} \sum_{i=1}^{I} x_i = \frac{1}{I} \sum_{i=1}^{I} C(\xi_i). \quad (3.36)$$

This entity $\bar{x}$ can be interpreted as a soft histogram. Using the mass-preserving basis functions and removing the normalization $\frac{1}{I}$, each term contributes one unit to the $L_1$ norm of the total channel vector.

For a soft histogram, each sample is distributed in several bins, the distribution of the sample depends on a kernel or basis function. Using this interpretation, each channel coefficient corresponds to one bin. The advantage is a possibility to determine properties, such as the mean of a mode, with higher precision than the channel spacing. This is similar to the advantages of the Averaged Shifted Histogram proposed by Scott [98]. The aim of Scott was to generate better distribution estimations than those obtained from histograms without the computational demands of kernel density estimators. For comparison, a regular histogram is obtained if a box basis function is used, which is one between $-0.5$ and $0.5$ and zero everywhere else.

A sampled kernel density estimate is obtained by placing a kernel function at each sample $\xi_i$ and adding the values of this kernel function evaluated at all grid points. However, assuming a symmetric kernel function, the same result is obtained by centering kernel functions at each grid point and evaluating these at each sample point $\xi_i$.

Assume that the samples $\xi_i$ are drawn from a distribution with density function $f_\xi(\xi)$, then it can be shown that the expected values of the channel coefficients
3.4. STATISTICAL INTERPRETATION

\( \bar{x} \) are \( f_\xi \) convolved with the channel basis function, evaluated at each respective channel center \([49, 35]\), which is similar to kernel density estimation.

3.4.2 Representing Probability Density Functions

Consider the mass-preserving channel basis function \( b(\xi) \) (3.15). The basis function fulfills all requirements of a probability density function, and so does a linear combination of shifted basis functions where the coefficients are non-negative and sum to one. Since channel vector coefficients are non-negative and can be normalized such that \( \|\bar{x}\|_1 = 1 \), a channel vector weighted combination of basis functions

\[
\sum_{n=1}^{N} x_n b(\xi - n)
\]

(3.37)

is a non-parametric representation of a density function. Furthermore, it will be shown that statistical moments and expected values of functions of the random variable, \( g(X) \), in general can be computed through scalar products of the channel vector \( \bar{x} \) and a constant vector. The elements of the constant vector only depend on the function \( g \) and the choice of basis functions. Not on the channel vector which defines the distribution. Specifically, expressions for the mean and variance will be derived.

The results presented here are not limited to \( \cos^2 \) basis functions but hold for any basis function which is in itself a valid probability density function. Furthermore, integer spacing of the basis functions is not assumed in general but it will be presented as special cases. The shifted versions of the basis functions will be denoted \( b_n(\xi) \). The main material in this section was published in [79].

Thus, let \( b_n(\xi) \geq 0 \quad \forall \xi \quad \text{for} \quad n = 1, \ldots, N \) be a set of channel basis functions normalized such that \( \int_{-\infty}^{\infty} b_n(\xi) \, d\xi = 1 \) and let \( \bar{x} = [x_1 \ x_2 \ \cdots \ x_N]^T, \ x_n \geq 0 \) be the channel coefficients representing a distribution density function

\[
f_X(\xi) = \sum_{n=1}^{N} x_n b_n(\xi)
\]

(3.38)

Let the coefficients be normalized such that \( \|\bar{x}\|_1 = 1 \). Then \( f_X(\xi) \geq 0 \ \forall \xi \) and \( \int_{-\infty}^{\infty} f_X(\xi) \, d\xi = 1 \) follow.

Consider the random variable

\[
X : P(X < z) = \int_{-\infty}^{z} f_X(\xi) \, d\xi
\]

(3.39)

Expectations of functions \( g(X) \) of this variable become scalar products with the channel coefficient vector since

\[
E[g(X)] = \int_{-\infty}^{\infty} g(\xi)f_X(\xi) \, d\xi = \sum_{n=1}^{N} x_n \int_{-\infty}^{\infty} g(\xi)b_n(\xi) \, d\xi = \sum_{n=1}^{N} x_n g b_n = \bar{x}^T g
\]

(3.40)
where $g_{bn} = \int_{-\infty}^{\infty} g(\xi) b_n(\xi) \, d\xi$ and $g = [g_{b_1} \, g_{b_2} \, \cdots \, g_{b_{-n}}]^T$. Note that the elements of $g$ are independent of the channel coefficients $x$.

Let $\mu$ and $\sigma^2$ denote the mean and variance of the represented distribution $f_X$. The mean is obtained by letting $g(X) = X$ in (3.40):

$$
\mu = E[X] = \sum_{n=1}^{N} x_n \mu_{bn}, \tag{3.41}
$$

where basis function means are represented by $\mu_{bn} = \int_{-\infty}^{\infty} \xi b_n(\xi) \, d\xi$. For symmetric basis functions, such as $\cos^2$ basis functions, these coincide with channel centers. Furthermore, if the channels are placed at integer positions starting at one, the expression reduces to

$$
\mu = \sum_{n=1}^{N} nx_n. \tag{3.42}
$$

For the variance, $g(X) = (X - \mu)^2 = X^2 - 2\mu X + \mu^2$ and

$$
\sigma^2 = E[(X - \mu)^2] = \sum_{n=1}^{N} x_n \int_{-\infty}^{\infty} (\xi^2 - 2\mu \xi + \mu^2) b_n(\xi) \, d\xi =
$$

$$
= \sum_{n=1}^{N} x_n \left( \int_{-\infty}^{\infty} \xi^2 b_n(\xi) \, d\xi - 2\mu \int_{-\infty}^{\infty} \xi b_n(\xi) \, d\xi + \mu^2 \int_{-\infty}^{\infty} b_n(\xi) \, d\xi \right) =
$$

$$
= \sigma_b^2 \sum_{n=1}^{N} x_n + \sum_{n=1}^{N} x_n \mu_{bn}^2 - 2\mu \sum_{n=1}^{N} x_n \mu_{bn} + \mu^2 \sum_{n=1}^{N} x_n =
$$

$$
= \sigma_b^2 - \mu^2 + \sum_{n=1}^{N} x_n \mu_{bn}^2
$$

where $\sigma_b^2 = \int_{-\infty}^{\infty} (\xi - \mu_{bn})^2 b_n(\xi) \, d\xi$ is the centered second order moment for the basis function, constant for all $n = 1, 2, \cdots, N$ under the assumption of a shifted set of basis functions. Again, in the special case of symmetric basis functions and $b_n(\xi) = b(\xi - n)$, this reduces to

$$
\sigma^2 = \sum_{n=1}^{N} n^2 x_n - \left( \sum_{n=1}^{N} nx_n \right)^2 + \sigma_b^2. \tag{3.43}
$$

For $\cos^2$ basis functions with overlap 3 and integer spacing, $\sigma_b^2 = \frac{3}{4} - \frac{9}{2\pi^2}$.

Hence the mean and variance (and thus the standard deviation) of a channel represented distribution can be obtained through scalar products of channel coefficients and constant weight vectors.
3.5 Summary and Contributions

This chapter has introduced the channel representation and collected a set of requirements on the basis function. These have appeared only implicitly or sporadically earlier. The constant $L_1$ and $L_2$ norm properties of representations based on certain basis functions are pointed out. These have been noted in previous work but in this work their importance is more significant.

The main contributions of this chapter are the results on basis function uniqueness and the analysis of relations to distributions. Previously the use of the truncated squared cosine basis function and three overlapping channels has been a bit ad hoc. Now there are more mathematically solid motivations.

Geometrically the requirements of constant $L_1$ and $L_2$ norms limit the channel vector curve $C(\xi)$ to the intersections of the surfaces of a $(N-1)$-simplex and a $(N-1)$-dimensional sphere. Furthermore, requiring constant speed, $\|\frac{d}{d\xi}C(\xi)\|_2$, along the curve, the only remaining basis function is the truncated $\cos^2$.

In the uniqueness proof, the requirement of non-negative Fourier coefficients of the basis function can be replaced by a requirement on a finite number of non-zero coefficients. Using neither requirement, there is a family of solutions on the form $b(\xi) = \frac{2}{3} \cos^2 \left( (\xi + \sin 2\pi \xi) \frac{2}{3} \right)$. These functions have an infinite number of non-zero Fourier coefficients, some of which are negative, and gives a non-constant speed along the channel vector curve. The three different requirements are thus similar in the sense that any one will limit the allowable basis functions to the truncated $\cos^2$.

The use of channel representations as kernel density estimators are further analyzed and expressions of expectation values and statistical moments of the represented distributions are found, expressed directly in the channel coefficients. For integer spacing of the basis functions, these expressions are reduced to a very simple form.
Chapter 4

Decoding the Channel Representation

Decoding of a channel vector is an operation which maps a channel vector to a single value. For representations where only one value is encoded, the decoding result should be this encoded value. In cases where the representation contains more than one mode, decoding can either provide the value of the strongest mode or several values, representing the different modes.

In this chapter, previous decoding methods for channel representations and population codes are presented. The latter is known as readout. However, the readout of population codes introduces bias in the estimate. The contributions in this chapter include an analysis of the readout method as well as a geometrical analysis of the channel representation and decoding. From this analysis, requirements for an unbiased decoding emerges. Furthermore, a decoding method based on maximum likelihood is presented and shown to differ from the previous channel decoding method only with respect to decoding interval selection.

4.1 A Brief History of Channel Decoding

Forssén provides general channel decoding frameworks for Gaussian, B-spline and \( \cos^2 \) channels, as well as for different overlaps [35]. Here, the \( \cos^2 \) channel decoding will be presented as a foundation for the remainder of the chapter. First the general version for integer overlap and later the specialized version for overlap three. Channel centers are assumed to be placed at integer positions and thus the integer channel width \( w \) coincide with the basis function overlap.

In section 3.4, it was shown how to obtain the mean of a channel representation in a distribution sense. This is usually not desired in cases where there are multiple modes in the represented distribution. The channel decoding presented by Forssén is instead based on decoding intervals or decoding windows. Using overlap \( w \), a single encoded value will activate \( w \) channels and thus \( w \) consecutive channels are considered when decoding.
Every unit length decoding interval within the representable interval will be mapped to such a set of $w$ active channels. If the value obtained after decoding does not belong in the decoding interval, another set of coefficients corresponding to another decoding interval is evaluated. Forssén proposed using the decoding interval with the greatest sum of the corresponding channel coefficients. Weaker modes could be obtained by selecting the interval corresponding to the second greatest sum and so on.

Denote the corresponding channel coefficients of the decoding interval $x_l$, $x_{l+1}$, $\ldots$, $x_{l+w-1}$, i.e. the index $l$ determines the position of the decoding interval in the channel vector. Assuming a single encoded value $\xi$, each coefficient provides an equation

$$x_{l+d} = r b_{l+d}(\xi) = r \cos^2((\xi - l - d)\pi/w) \quad \text{for } d = 0, 1, \ldots, w - 1,$$

where $b_{l+d}$ is the basis function centered at $l + d$ and $r$ is a general scaling factor. The expression can be reformulated \[35\] into

$$x_{l+d} = \begin{pmatrix} 1/2 \cos(2\pi d/w) & 1/2 \sin(2\pi d/w) & 1/2 \end{pmatrix} \begin{pmatrix} r \cos(2\pi(\xi - l)/w) \\ r \sin(2\pi(\xi - l)/w) \\ r \end{pmatrix}$$

(4.2)

where the sought $\xi$ only appears in the last vector. By stacking $w$ equations, the system

$$\begin{pmatrix} x_l \\ x_{l+1} \\ \vdots \\ x_{l+w-1} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \cos(2\pi 0/w) & \sin(2\pi 0/w) & 1 \\ \cos(2\pi 1/w) & \sin(2\pi 1/w) & 1 \\ \vdots & \vdots & \vdots \\ \cos\left(\frac{2\pi (w-1)}{w}\right) & \sin\left(\frac{2\pi (w-1)}{w}\right) & 1 \end{pmatrix} \mathbf{p}$$

(4.3)

is obtained, where

$$\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} := \begin{pmatrix} r_1 \cos(2\pi(\xi - l)/w) \\ r_1 \sin(2\pi(\xi - l)/w) \\ r_2 \end{pmatrix}.$$  

(4.4)

The equation system can be solved in a least squares sense for $w \geq 3$ to obtain $\mathbf{p}$. Since the coefficient matrix is independent of the channel vector, the pseudo inverse can be calculated in advance. Finally, $\xi$, $r_1$ and $r_2$ can be obtained from $\mathbf{p}$ as $r_2 = p_3$, $r_1 = |p_1 + ip_2|$ and

$$\hat{\xi} = \frac{3}{2\pi} \arg(p_1 + ip_2) + l.$$  

(4.5)

Note that the scale parameter $r$ is split into two parameters $r_1$ and $r_2$ as the assumption of a single encoded value does not hold in general. The relation between $r_1$ and $r_2$ will be used as a measure of closeness to the ideal case.

In the special case $w = 3$, the coefficient matrix becomes

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 1 \\ \frac{1}{2} & 0 & 1 \\ \frac{1}{2} & 1/2 \sqrt{3}/2 & 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 0 & 1 \\ -1/2 & \sqrt{3}/2 & 1 \\ -1/2 & -\sqrt{3}/2 & 1 \end{pmatrix},$$  

(4.6)

i.e. square and invertible.
4.1.1 Orthonormal Decoding

For notational convenience, let $x_l$ be the part of the channel vector $x$ corresponding to the current decoding window, $x_l = [x_l, x_{l+1}, x_{l+2}]^T$. Further investigation of the coefficient matrix (4.6) reveals its orthogonal column vectors, and that the two first column vectors (corresponding to $r_1$) have the same $L_2$ norm. Thus by redefining $r_1$ and $r_2$, the decoding equation system can be reformulated using an orthonormal coefficient matrix \[ x_l = \frac{1}{\sqrt{3}} \begin{pmatrix} \sqrt{2} \cos(0\pi/3) & \sqrt{2} \sin(0\pi/3) & 1 \\ \sqrt{2} \cos(2\pi/3) & \sqrt{2} \sin(2\pi/3) & 1 \\ \sqrt{2} \cos(4\pi/3) & \sqrt{2} \sin(4\pi/3) & 1 \end{pmatrix} \begin{pmatrix} r_1 \cos(2\pi(\xi - l)/3) \\ r_1 \sin(2\pi(\xi - l)/3) \\ r_2 \end{pmatrix} \] (4.7)
or, inverting the expression

$$p = Wx_l = \frac{1}{\sqrt{3}} \begin{pmatrix} \sqrt{2} & -1/\sqrt{2} & 1/\sqrt{2} \\ 0 & 3/\sqrt{2} & -3/\sqrt{2} \\ 1 & 1 & 1 \end{pmatrix} x_l$$

(4.8)

where

$$p = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} := \begin{pmatrix} r_1 \cos(2\pi(\xi - l)/3) \\ r_1 \sin(2\pi(\xi - l)/3) \\ r_2 \end{pmatrix}.$$  

(4.9)

The matrix $W$ is the inverted (transposed) orthonormal coefficient matrix, a rotation matrix aligning $p_3$ with the scaling direction of the channel vector, $[1, 1, 1]^T$. The decoded value is obtained by

$$\hat{\xi} = \frac{3}{2\pi} \arg(p_1 + ip_2) + l,$$

(4.10)

similar to earlier results.

Using the mass preserving basis functions and the norm invariant property of the channel encoding, it is clear that for single encoded values within the decoding window

$$r_2 = p_3 = \frac{1}{\sqrt{3}}\|x_l\|_1 = \frac{1}{\sqrt{3}}$$

(4.11)

and

$$\|p\|_2 = \|Wx_l\|_2 = \|x_l\|_2 = \frac{1}{\sqrt{2}}$$

(4.12)

since rotations preserve $L_2$ distances. Furthermore,

$$r_1 = \sqrt{p_1^2 + p_2^2} = \sqrt{\|p\|_2^2 - p_3^2} = \frac{1}{\sqrt{6}}$$

(4.13)

which leads to the conclusion that for single encoded values within the decoding window

$$r_2 = \sqrt{2} r_1.$$  

(4.14)
Figure 4.1: Channel vector curve $\mathbf{x} = \mathcal{C}(\xi)$ for $\xi$ obtaining all values within the same decoding window (solid line). First value marked with a circle, last value marked with a cross. Dashed: the directions of $p_1$, $p_2$ and $p_3$. Note that the points of $\mathbf{x}_t$ trace out a third of a circle in the plane spanned by the directions of $p_1$ and $p_2$. By also scaling the channel vector, the surface of a cone is obtained.
This holds also for arbitrary scalings of the channel vector $\alpha x$, $\alpha > 0$, since (4.11) through (4.13) are homogeneous with respect to scaling of the channel vector.

Figure 4.1 illustrates the geometry. Consider scaled channel vectors $\alpha C(\xi)$ and let the encoded value $\xi$ vary within a decoding interval while the scale $\alpha > 0$ varies. The obtained points in channel vector space will be on the surface of a cone with angle $2 \tan^{-1}(1/\sqrt{2})$ (approximately 70.5 degrees) and symmetry axis $[1, 1, 1]^T$. Using the un-normalized decoding (4.6), $r_1$ and $r_2$ are equal for single encoded values, the cone gets wider (90 degrees) when solving the equation system. Still, the cone in the channel vector space remains at approximately 70.5 degrees.

If a set of measurements is represented using channels, the resulting channel vector $x$ is a convex combination of points on the surface of this cone, i.e., a point in the interior of the cone, and

$$r_1 \leq \frac{r_2}{\sqrt{2}} \quad \text{(4.15)}$$

in general. Decoding this channel vector is then related to finding the closest point on the cone surface. The exact definition of closest in this context is dependent on the transformation group considered. Most decoding options, as will be shown, will select a point on the cone somewhere along the intersection line with the half-plane defined by the cone symmetry axis and the interior point. All those points correspond to the same decoded value. Any differences lie in the selection of the decoding window. Using the greatest sum criterion for decoding window selection corresponds to maximizing $r_2$.

### 4.1.2 Maximum Likelihood Decoding

In this section, the maximum likelihood decoding approach of Felsberg, Öfjäll and Lenz [33] is presented. The approach leads to the same solution as previous decoding, assuming the same decoding window selection. However, the window selection criterion will be different, maximizing $r_1 + \sqrt{2}r_2$.

Here $x_l$ denotes the three-element vector corresponding to a decoding window $l$, and $C_l(\xi)$ the corresponding three elements of a single encoded value $\xi$. Furthermore, it is assumed that $l$ is chosen (dependent on $\xi$) such that $C_l(\xi)$ contains all non-zero values of $C(\xi)$. Thus,

$$x^T C(\xi) = x_l^T C_l(\xi) \quad . \quad \text{(4.16)}$$

Assume the coefficients of the channel vector to be decoded, $x$, are disturbed
by independent Gaussian noise, then the ML estimate is obtained as

\[
\hat{\xi} = \arg \max_{\xi} P(\xi|x) = \arg \max_{\xi} \exp\left(-\frac{1}{2} \|C(\xi) - x\|^2 \right) \quad (4.17)
\]

\[
= \arg \min_{\xi} \|C(\xi) - x\|^2 \quad (4.18)
\]

\[
= \arg \min_{\xi} C(\xi)^T C(\xi) - 2x^T C(\xi) + x^T x \quad (4.19)
\]

\[
= \arg \max_{\xi} \frac{1}{2} x^T C(\xi) \quad (4.20)
\]

\[
= \arg \max_{\xi} x^T C_l(\xi) \quad (4.21)
\]

\[
= \arg \max_{\xi} x^T l^T W^T C_l(\xi) \quad (4.22)
\]

\[
= \arg \max_{\xi} \left( Wx_l \right)^T W C_l(\xi) \quad (4.23)
\]

\[
= \arg \max_{\xi} \begin{bmatrix} r_1 \cos \beta & r_2 \cos \gamma(\xi) + \sin \beta \sin \gamma(\xi) \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} \cos(2\pi(\xi - \ell)/3) \\ \frac{1}{\sqrt{6}} \sin(2\pi(\xi - \ell)/3) \\ \frac{1}{\sqrt{3}} \end{bmatrix} \quad (4.24)
\]

\[
= \arg \max_{\xi} \begin{bmatrix} r_1 \cos \beta \\ r_1 \sin \beta \\ r_2 \end{bmatrix} = \frac{1}{\sqrt{3}} \begin{bmatrix} r_1 \sqrt{2} + r_2 \end{bmatrix} \quad (4.25)
\]

where \( r_1, r_2 \) and \( \beta \) are dependent on the channel vector \( x \) and the decoding window selection \( l \). At this point, the optimization is split into a continuous problem and a discrete problem. Given a selection of decoding window, only \( \gamma(\xi) := 2\pi(\xi - \ell)/3 \) is non-constant and the optimal choice is obviously \( \gamma(\xi) = \beta \), i.e.

\[
\hat{\xi} = \frac{3}{2\pi} \arg(\cos \beta + i \sin \beta) + \ell \quad (4.26)
\]

the same solution as (4.5). The choice \( \gamma(\xi) = \beta \) reduces (4.26) to

\[
\arg \max_{\xi} \frac{1}{\sqrt{3}} \left( \frac{r_1}{\sqrt{2}} + r_2 \right) \quad (4.27)
\]

leaving the discrete decoding window selection problem with the solution of choosing decoding window such that \( r_1 + \sqrt{2} r_2 \) is maximized.

The result shows that the iso-decoding window weight lines have the direction \( [1, -\sqrt{2}]^T \) in \( r_1, r_2 \)-space. Compare this to the expression for the surface of the cone of channel coefficients (4.14), \( \sqrt{2} r_1 = r_2 \Leftrightarrow \sqrt{2} r_1 - r_2 = 0 \), with direction \( [\sqrt{2}, 1]^T \). That is, for maximum likelihood decoding, the iso-decoding window weight lines are orthogonal to the surface of the cone.

4.2 Geometric Analysis

For mass preserving \( \cos^2 \) channels, the sum of channel coefficients (\( L_1 \)-norm) is constant 1 for all \( \xi \) in the representable area. Thus, the channel vector lies in the
4.2. GEOMETRIC ANALYSIS

$(N-1)$ simplex. The $L_2$ norm is constant $1/\sqrt{2}$ such that the channel vector also lies on the hypersphere with radius $1/\sqrt{2}$. Furthermore, projecting the channel vector onto the subspace orthogonal to $[1, \ldots, 1]^T$, the $L_2$ norm is still invariant with respect to the encoded value, see section 4.2.2. For increasing $\xi$, the $N$-dimensional channel vector projected onto this $(N-1)$-dimensional hyperplane will generate a 1D curve along the intersection of the $(N-1)$-simplex and the surface of the $(N-1)$-dimensional hypersphere with radius $\sqrt{1/2 - 1/N}$. See figure 4.2.

Figure 4.2: Channel vector curve $x_l = C(\xi)$ for $\xi$ obtaining all values within a modular representable range. Parts of the curve belonging to different decoding windows have different color. The constant direction $[1, \ldots, 1]^T$ is projected away and the $(N-1)$-simplex is drawn. The curve and simplex are rotated and then projected orthogonally to 2D. From left to right, $N = 3$, $N = 4$ and $N = 5$. See the supplementary material at http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125916 for video illustrations.

Averaging channel representations over a set of measurements results in a convex combination of points on the curve. Furthermore, since any valid channel vector is a convex combination of singular events on the input space, all valid channel vectors lie within the convex hull of the curve generated from Diracs in the input space. Introducing a non-negative scaling of the channel vector, the curve extends to sections of cones, each interval covering one third of a circle. For three modular channels, a full cone is generated in 3D, see figure 4.3.

4.2.1 Iso-decoding Operations

Since there are three channel coefficients corresponding to each decoding interval, there is a 2D space of changes to these three elements which do not alter the decoded value. Those are changes which lie in the plane orthogonal to

$$\frac{dC(\xi)}{d\xi} = \begin{pmatrix}
-\frac{2\pi}{9} \sin\left(\frac{2\pi(\xi - l - 0)}{3}\right) \\
-\frac{2\pi}{9} \sin\left(\frac{2\pi(\xi - l - 1)}{3}\right) \\
-\frac{2\pi}{9} \sin\left(\frac{2\pi(\xi - l - 2)}{3}\right)
\end{pmatrix}, \quad (4.29)$$
Figure 4.3: Cone generated by three modular channels and channel vector scaling. The circle marks the encoded value 0.65 with three lines illustrating scaling (maintains decoded value), $\frac{d\mathcal{C}(\xi)}{d\xi}$ (changes decoded value) and the direction orthogonal to the cone surface into the cone (maintains decoded value but decreases coherence). The three different colors of the mesh illustrate the different decoding windows.
where \( l \) determines the decoding interval. The basis function derivative was presented in 3.19.

For a general channel vector, the above expression would require decoding the channel vector to obtain \( \xi \). Instead, a normal to the iso-decoding plane can be obtained directly from \( x_l \), the coefficients of the channel vector corresponding to the decoding interval. From the channel decoding (section 4.1) it is clear that iso-decoding is obtained in a half-plane containing the origin, \( x_l \) and \([1, 1, 1]^T\), interpreted as points. Thus the vectors \( x_l \) and \([1, 1, 1]^T\) span this space. A normal to the iso-decoding space can thus be obtained by the cross product

\[
\begin{pmatrix}
1 \\
1 \\
1
\end{pmatrix} \times x_l = \begin{pmatrix}
0 & -1 & 1 \\
1 & 0 & -1 \\
-1 & 1 & 0
\end{pmatrix} x_l.
\]

That is, any additive change \( \Delta x_l \) to the channel vector fulfilling

\[
\Delta x_l^T \begin{pmatrix}
0 & -1 & 1 \\
1 & 0 & -1 \\
-1 & 1 & 0
\end{pmatrix} x_l = 0
\]

will not alter the decoded value in the current decoding interval. Such changes include scaling of the channel vector \( \Delta x_l = \alpha x_l \) and an operation related to sharpening or smoothing of the peak. The latter is related to coherence, the topic of section 4.3. Figure 4.3 illustrates the iso-decoding space in relation to the cone. Figure 4.4 illustrates the effect on the channel vector of moving along the two vectors spanning the two-dimensional iso-decoding space.

Note that any such change does not alter the iso-decoding space, this is clear both geometrically and since \( \Delta x_l^T (a \times \Delta x_l) = 0 \) for arbitrary vectors \( \Delta x_l \) and \( a \). However, smoothing operations on the channel vector does not in general belong to the iso-decoding operations.

The same results can be derived through transformation group theory, compare (4.31) with equation (20) in the work of Felsberg, Öfjäll and Lenz [33]. The cone structure of the channel representation is then related to the \((2 + 1)D\) Minkowski space \( \mathbb{R}^{2,1} \) and Lorentz transformations known from relativity theory.

### 4.2.2 The \((N - 1)\)-Dimensional Hypersphere

Consider the \( \cos^2 \) channel vector \( C(\xi) \) with its constant \( L_1 \) and \( L_2 \) norm properties, and a normalized basis vector for the space orthogonal to the \((N - 1)\) simplex

\[
v = \frac{1}{\sqrt{N}} [1, \ldots, 1]^T.
\]

The projection matrix

\[
P = I - vv^T.
\]

\(^1\)Unless \( x_l \) is parallel to \([1, 1, 1]^T\), in which case the decoding of \( x \) is ambiguous anyway.
Figure 4.4: Iso-decoding operations on a channel vector. The first row illustrates general scaling (moving along the surface of the cone) and the second row is a change orthogonal both to scaling and to $\frac{dc(\xi)}{d\xi}$, i.e. orthogonal to the cone surface. Note that the first operation leaves the decoding for all decoding windows constant while the second will alter the decoding in nearby decoding windows.

projects the channel vector onto the subspace orthogonal to $v$ and thus from orthogonality, $(PC(\xi))^T(vv^Tc(\xi)) = 0$, follows

$$\|PC(\xi)\|_2 = \sqrt{\|C(\xi)\|_2^2 - \|vv^Tc(\xi)\|_2^2} = \sqrt{\frac{1}{2} - (v^Tc(\xi))^2} = \sqrt{\frac{1}{2} - \left(\frac{1}{\sqrt{N}}\|C(\xi)\|_1\right)^2} = \sqrt{\frac{1}{2} - \frac{1}{N}}, \quad (4.34)$$

that is, $PC(\xi)$ lies on the $(N-1)$-dimensional hypersphere with radius $\sqrt{1/2 - 1/N}$.

As apparent, both constant $L_1$ and $L_2$ norms are required for this result to hold. Thus, other basis functions such as B-spline or Gaussian will not show this type of symmetry. The corresponding leftmost plot in figure 4.2 will not be a circle.

### 4.3 Evidence and Coherence

Consider a sum of channel vectors of encoded values

$$\sum_{k=1}^{K} C(\xi_k), \quad (4.35)$$

the parameters $r_1$ and $r_2$ of the channel decoding are of interest. Together with the decoded value $\hat{\xi}$, these form a value/evidence/coherence triplet. There will
4.3. EVIDENCE AND COHERENCE

be reasons to return to these concepts in the light of machine learning applications where there are reasons to reduce the influence of inputs with strong evidence but low coherence. Another example of the three-folded representation by value/coherence/evidence is the structure tensor [7, 36], where the sum of eigenvalues is a measure for the evidence, the difference of eigenvalues (divided by their sum) represents the coherence, and the orientation of the eigenvectors determines the value. It has been pointed out in that the structure tensor results in a 3D cone [31]. The contents of this section is primarily adapted from [75].

Evidence is related to the number of samples that support a mode, i.e. \( r_2 \), which is the \( L_1 \) norm of the decoding window coefficients. It is related to the distance from the origin of the cone, and constant on planes orthogonal to the symmetry axis of the cone. Considering a full channel vector, the evidence is the sum of all channel coefficients, the \( L_1 \) norm, and if mass preserving basis functions are used, it is equal to the number of samples \( K \).

Coherence is defined as \( r_1^2 / r_2^2 \). It is a measure of the consistency of samples which makes up a mode, i.e. related to the closeness to the surface of the cone, but compensated for evidence. Coherence is defined as a property related to a specific decoding window, the definition is extended to a full channel vector as the evidence of the strongest mode. The strongest mode is either the mode with largest evidence [35] or the mode where \( r_1 + \sqrt{2} r_2 \) is maximal (maximum likelihood decoding).

The final component of the triplet, value, is the value obtained from the channel decoding. It is related to rotations of the channel vector around the symmetry axis of the cone.

4.3.1 Coherence

The evidence, \( r_2 \), is directly obtained as the sum of channel coefficients. As shown later in this section, the coherence \( \text{coh}(x_l) \) can be calculated as a ratio of quadratic forms

\[
\text{coh}(x_l) = \frac{r_1^2}{r_2^2} = \frac{1}{x_l^T \mathbf{1} x_l} x_l^T \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} x_l , \tag{4.36}
\]

where \( \mathbf{1} = (1,1,1)^T \) and \( x_l \) consists of the three elements from the decoding window of the channel vector \( x \). It follows that when all elements in \( x_l \) are equal, \( r_1 = 0 \) and decoding is ambiguous. When the values within the decoding window are such that \( r_1 \) is large, i.e. the channel coefficients are close to the surface of the cone, the estimate of \( \xi \) is less dependent on small perturbations of the channel coefficients. However, the absolute value of \( r_1 \) varies with the scaling of the channel coefficients.

The coherence attains values \( 0 \leq \text{coh}(x_l) \leq \frac{1}{2} \). Coherence zero indicates ambiguous decoding while coherence \( \frac{1}{2} \) indicates that \( x_l \) lies on the surface of the cone, either since only a single value is encoded or since all encoded samples correspond to the same value. Specifically, \( \text{coh}(x_l) = \text{coh}(\alpha x_l) \), i.e. the coherence is independent of scale (\( \alpha > 0 \)) and \( \text{coh}(x_l) \) decreases with a wider distribution of
For single-mode distributions, the coherence is inversely related to the variance of the distribution where wide distributions generate low coherence and vice versa. However, for multimodal distributions, variance is generally large as it is a global property of the distribution. On the contrary, coherence may still be large (corresponding to low variance of the strongest mode) as it is a local property of the individual mode.

The coherence quadratic form expression (4.36) is derived from channel decoding. As previously stated, the decoding of a channel representation is obtained by solving for $\xi$ in

$$
\begin{pmatrix}
  r_1 \cos \left( \frac{2\pi}{3} (\xi - l) \right) \\
  r_1 \sin \left( \frac{2\pi}{3} (\xi - l) \right) \\
  r_2
\end{pmatrix} = \frac{1}{\sqrt{3}} \begin{pmatrix}
  \sqrt{2} & -1/\sqrt{2} & 1/\sqrt{2} \\
  0 & \sqrt{3}/2 & -\sqrt{3}/2 \\
  1 & 1 & 1
\end{pmatrix} x_l
$$

(4.37)

where $l$ selects the decoding window and $x_l$ is the corresponding three elements from the channel vector $x$ to be decoded. Evidence is obtained as

$$
r_2 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} W x_l = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} x_l
$$

(4.38)

and its square

$$
r_2^2 = \frac{1}{3} x_l^T \text{11}^T x_l = \frac{1}{3} x_l^T \begin{pmatrix} 1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix} x_l
$$

(4.39)

follows, with $1 = (1,1,1)^T$. Similarly, $r_1^2$ is obtained as

$$
r_1^2 = x_l^T W^T \begin{pmatrix} 1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix} W x_l = \frac{1}{3} x_l^T \begin{pmatrix} 2 & -1 & -1 \\
-1 & 2 & -1 \\
-1 & -1 & 2
\end{pmatrix} x_l
$$

(4.40)

since

$$
\begin{pmatrix} 1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix} W x_l = \begin{pmatrix} r_1 \cos (2\pi (\xi - l)/3) \\
 r_1 \sin (2\pi (\xi - l)/3)
\end{pmatrix}
$$

(4.41)

and

$$
\begin{pmatrix} r_1 \cos (2\pi (\xi - l)/3) \\
 r_1 \sin (2\pi (\xi - l)/3)
\end{pmatrix}^T \begin{pmatrix} r_1 \cos (2\pi (\xi - l)/3) \\
 r_1 \sin (2\pi (\xi - l)/3)
\end{pmatrix} = r_1^2 \left( \cos^2(\cdot) + \sin^2(\cdot) \right) = r_1^2
$$

(4.42)

### 4.4 Decoding Bias, Cause and Mitigation

Imagine drawing samples $\xi_k$ from a symmetric unimodal distribution and sum the channel vectors of these $x = \sum_{k=1}^K C(\xi_k)$. One would expect the decoded value $\hat{x}_i = C^T(x)$ to be an estimate of the distribution mean. Using three channels on
a modular domain, this is the case, the decoded value is an unbiased estimate of the distribution mean. This is illustrated in figure 4.5.

Using four or more channels, the encoded values lie in different planes if the distribution crosses a decoding interval limit. This offsets the mean channel vector, generating a systematic estimation bias dependent on the absolute position of the true mean with respect to channel centers. Figure 4.5 illustrates this.

If the distribution only crosses one decoding interval limit and the correct decoding interval is known beforehand, the bias can be removed by rotating one half-space such that the curve sections of the two intervals lie in the same plane before combining the channel vectors of the samples. Similar actions can be taken if all encoded vectors are available at decode time. However, both cases are rare.

In this section, we consider an iterative approach to reduce decoding bias by letting activation flow between channel coefficients in the vector prior to decoding. The idea is to sharpen the activations in the channel vector, taking into consideration values outside the coefficients belonging to the decoding interval. Since there is no bias in the three modular channel case, such an approach should not introduce any bias when applied to three-dimensional channel vectors. Furthermore, a well-known iterative scheme from computational neuroscience regarding readout of population codes is analyzed [23]. The main part of this section is based on [33].

4.4.1 Analysis of Population Code Readout

Population coding originates from the computational neuroscience field. The underlying idea is similar to the biological interpretation of the channel representation, tuning curves represent the average activation of neurons by a certain stimulus. One example is the response of certain neurons to edges with different orientations [46].

From a population of such neurons with different preferred orientations, an unbiased estimate of the true orientation is desired. The general idea applies to any other estimate of an entity represented by the mean activation of a population of neurons, population coding [113, 88]. One such method is claimed to be the "read out [of] these noisy hills" [23], represented by a population vector, much similar to a channel vector. A selection of decoding procedures for channel vectors has been presented earlier in this chapter. In contrast, the population vector readout uses a recurrent procedure to approximate a maximum likelihood estimation.

The readout by means of the population vector estimator is unbiased [23], however, the combination of the recurrent procedure and the estimator becomes biased if the recurrent equation is iterated until convergence. This effect is reduced if the tuning functions overlap extensively and if the recurrent equation is only iterated a few (2-3) times. Both approaches are proposed for readout [23]. The bias of the recurrent procedure was shown in [33]. Here, the paths to the stationary points will be illustrated.
Figure 4.5: Mean channel vector (red cross) in a channel representation with three channels on a modular domain. 50 of 5000 encoded samples from a Gaussian distribution are shown with blue crosses. The red line indicates the iso-decoding plane of the true mean value of the distribution. Since all encoded values lie in the same plane, decoding the mean channel vector will generate an unbiased estimate of the distribution mean. Note that coherence will not be maximal as the mean channel is not on the circle, indicating a wide distribution of samples that generated the mode.
4.4. DECODING BIAS, CAUSE AND MITIGATION

Figure 4.6: Mean channel vector (red cross) in a channel representation with four channels. 50 of 5000 encoded samples from a Gaussian distribution are shown with blue crosses. The red line indicates the position of the iso-decoding plane of the true mean value of the distribution. All points are projected onto the 3D space orthogonal to \([1, 1, 1, 1]^T\). Since the two curve segments corresponding to different decoding intervals do not lie in the same plane, the mean channel vector is shifted away from the correct iso-decoding plane, introducing a bias in the estimated mean. Furthermore, this bias is dependent on the position relative to channel centers. (a) shows a 3D view, the two curve segments lie in the planes of two sides in the 3-simplex. (b) shows the orthogonal projection onto the plane of the first curve segment.
The recurrent procedure proposed by Deneve is \[23\]

\[
\begin{align*}
  u_j(t + 1) &= \sum_k w_{jk} o_k(t) \quad (4.43) \\
  o_j(t + 1) &= \frac{u_j(t + 1)^2}{S + \mu \sum_k u_k(t + 1)^2} \quad , \quad (4.44)
\end{align*}
\]

where \(o_j(t)\) is the time evolution of the activations of each neuron and \(w_{jk}\), \(S\), \(\mu\) are parameters. Collect the \(o_j\) in the vector \(\mathbf{o}\), the \(u_j\) in a vector \(\mathbf{u}\) and the parameters \(w_{jk}\) in a matrix \(\mathbf{A}\). Thus, the procedure can be written

\[
\begin{align*}
  \mathbf{u}(t + 1) &= \mathbf{A} \mathbf{o}(t) \quad , \quad (4.45) \\
  o_j(t + 1) &= \frac{u_j(t + 1)^2}{S + \mu \|\mathbf{u}\|^2} \quad . \quad (4.46)
\end{align*}
\]

In the following, a population of three is considered. When the procedure has converged

\[
o(t + 1) = \mathbf{o}(t). \quad (4.47)
\]

If \(\mathbf{o}\) is a stationary point, \(\mathbf{u} = \mathbf{A} \mathbf{o}\) is constant over time and so is also the normalization factor \(Q := S + \mu \|\mathbf{u}\|^2\). For the procedure to be independent of the arbitrary indexing of neurons (i.e. selecting the first neuron on a modular domain and in which direction they are numbered), the matrix \(\mathbf{A}\) must be circulant and symmetric, i.e.

\[
\mathbf{A} = \begin{pmatrix}
  \alpha & \beta & \beta \\
  \beta & \alpha & \beta \\
  \beta & \beta & \alpha
\end{pmatrix} \quad (4.48)
\]

for suitable \(\alpha\) and \(\beta\). For a stationary solution \(o_j = \frac{1}{Q} (\mathbf{A} \mathbf{o})_j\) is obtained (where \((\cdot)_j\) selects the \(j\)th element of a vector) and therefore

\[
\begin{align*}
  o_1 &= \frac{1}{Q} (\alpha o_1 + \beta o_2 + \beta o_3)^2 \\
  o_2 &= \frac{1}{Q} (\beta o_1 + \alpha o_2 + \beta o_3)^2 \\
  o_3 &= \frac{1}{Q} (\beta o_1 + \beta o_2 + \alpha o_3)^2
\end{align*} \quad . \quad (4.49)
\]

It is clear that \(Q\) only affects absolute scale of stationary points, as does simultaneous scaling of \(\alpha\) and \(\beta\). Thus only one degree of freedom remains for changing the stationary points: the quotient of \(\beta/\alpha\).

Without loss of generality, let \(S = 0.1\), \(\mu = 0.9\), and \(\alpha = 1\). For different choices of \(\beta\), the trajectories of different initial points are plotted in figure 4.7. For comparison, see figure 4.8, illustrating the paths of an unbiased recurrent procedure.
4.4. DECODING BIAS, CAUSE AND MITIGATION

Figure 4.7: Iteration paths of the recurrent procedure for readout, for different $\beta$ values. Initial values are marked with crosses and final values with circles. Points in 3D are projected onto the plane with normal $[1, 1, 1]^T$, the corners of the triangle correspond to $[1, 0, 0]^T$, $[0, 1, 0]^T$ and $[0, 0, 1]^T$ respectively. For low values of $\beta$, there are stationary points close to the corners of the triangle, corresponding to three different decoded values. For larger values of $\beta$, the stationary points approach the center of the figure, corresponding to ambiguous decoding.
Figure 4.8: Iteration paths of the proposed recurrent procedure for channel mode
sharpening [33]. Initial values are marked with crosses and final values with circles.
Points in 3D are projected onto the plane with normal $[1, 1, 1]^T$, the corners of
the triangle correspond to $[1, 0, 0]^T$, $[0, 1, 0]^T$ and $[0, 0, 1]^T$ respectively. Since
points stay on the initial angular position with respect to the circle, the procedure
is unbiased.
4.4. DECODING BIAS, CAUSE AND MITIGATION

4.4.2 Reducing Decoding Bias

By deriving a different recurrent procedure, it is possible to reduce bias seen for channel vectors with more than three coefficients. The starting point of such a scheme is the requirement of decoding invariance for three channel cases, i.e. the update \( \Delta x \) should fulfill (4.31)

\[
\Delta^T_{x_l} \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} x_l = 0.
\]

(4.50)

Furthermore, the iteration should leave evidence unchanged, i.e.

\[
\Delta^T_{x_l} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 0.
\]

(4.51)

The solution is thus obtained by forming the cross product operator matrix

\[
\Delta_{x_l} = -\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \times \left( \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} x_l \right) = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} x.
\]

(4.52)

The sign is chosen such that the step length in the iterative scheme is positive. The operator contains cyclic permutations of the negative discrete Laplacian kernel \([-1, 2, -1]^T\) and the iterations can be expressed as

\[
x_+ = x + \alpha \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} x
\]

(4.53)

for a suitable \( \alpha \). This is iterated until \( \sqrt{2} r_1 = r_2 \), i.e. the channel vector reaches the surface of the cone. This recurrent operation on three-dimensional vectors is illustrated in figure 4.8. As expected, the projected points move radially towards the circle.

For the three channel case, a solution can be found algebraically. The iterative approach is required for higher-dimensional channel vectors where the algebraic structure becomes more complicated. Points are no longer projected onto a circle in a plane but onto a more general 1D curve in high-dimensional space. The approach for three channels is generalized by splitting the Laplacian operator and defining a flow vector \( d \), where each coefficient \( d_k \) determines how much of the channel coefficient \( x_k \) is moved to \( x_{k+1} \), i.e. \( x_{k+1} = x_k - d_k + d_{k-1} \).

For the three-dimensional case \( d_k = \alpha(x_{k+1} - x_k) \) using modular indexing, i.e.

\[
x_+ = \begin{pmatrix} x_{1+} \\ x_{2+} \\ x_{3+} \end{pmatrix} = \begin{pmatrix} x_1 - d_1 + d_3 \\ x_2 - d_2 + d_1 \\ x_3 - d_3 + d_2 \end{pmatrix} = \begin{pmatrix} x_1 + \alpha(x_1 - x_2 - x_3 + x_1) \\ x_2 + \alpha(x_2 - x_3 - x_1 + x_2) \\ x_3 + \alpha(x_3 - x_1 - x_2 + x_3) \end{pmatrix}
\]

(4.54)

which is the same relation as (4.53). Using more than three channels, the flow depends non-linearly on the channel coefficients \( d_k = f(x_k, x_{k+1}) \). Assuming that
the function $f$ is known and $\alpha$ being the update step length, the following iterative algorithm is obtained, where any index that goes outside the vector is wrapped around:

```plaintext
while $\sqrt{2}r_1 < r_2$ do
    for all $k$ do
        $d_k \leftarrow f(c_k, c_{k+1})$
    end for
    for all $k$ do
        $c_k \leftarrow c_k + \alpha(d_{k-1} - d_k)$
    end for
end while
```

The function $f$, illustrated in figure 4.9, is hard to determine analytically but can be numerically sampled by estimating channel vectors based on distributions with varying mean and variance. Note that the mean channel vector of an infinite number of samples from a certain distribution can be obtained by the convolution of the basis function and the probability density function, see section 3.4.

![Plot of the function for determining the flow between two channels $d_k = f(x_k, x_{k+1})$.](image_url)
4.4.3 Bias Reduction Experiment

For the effects of the bias reduction iterations to appear, broad distributions of the initial samples are required. To illustrate the effect, a synthetic orientation estimation experiment is set up. Individual orientation estimates (in the range from $-\pi$ to $\pi$) are sampled from a known distribution with variable properties. The samples are channel encoded and then averaged. The mean channel vector is treated using one of three different options: applying the derived bias reduction iterations, using the recurrent procedure proposed by Deneve or leaving the channel vector unaltered. It is then decoded and the obtained estimate of the mean orientation is compared to the true mean from the known distribution. This is repeated for several positions of the true mean relative to channel centers. A channel representation using seven channels is used.

The results of these trials are presented in figure 4.10. In both cases, applying 10 iterations of Deneve’s method increases the bias. The plot converges towards a step function as the iterations pushes towards channel centers, see also figure 4.7.

The direct decoding cases, (a) and (b) in figure 4.10, where the average channel vector is directly decoded, shows a difference depending on the width of the distribution. For a narrow distribution of samples, in this case a standard deviation of half the decoding interval length, the estimates are unbiased. The plots of true and estimated mean form a noisy diagonal line, but without structure in the noise, see (a). For the wider distribution, with standard deviation 1.5 times the decoding interval length, the line has visible waves, see (b). The seven waves correspond to the seven channels, and thus there is a systematic bias in the estimates. Although the resolution of the figure does not directly allow this, it is worth noting that due to symmetry, there is no estimation bias if the true mode is located on a channel center or exactly between two channel centers.

For the derived bias reducing iterations, the results are displayed in (e) and (f) of 4.10. In the narrow distribution case (e), there is no difference compared to the direct decoding (a). Thus the iterations do not introduce any bias. In the case of wide distribution, the bias is reduced by iterating the proposed scheme, compare (f) with (b). Examining the results carefully, there is still some remains of the bias, however, it is significantly reduced. The marks in the upper left and lower right corners of all plots are due to the modularity of the orientation.

4.5 Summary and Contributions

This chapter presented the decoding of channel representations. A new decoding approach was presented based on maximum likelihood. The new approach was shown to coincide with the previous decoding except with respect to the decoding interval selection criterion. A geometrical view of the channel representation was presented which allowed for analysis of iso-decoding spaces and iso-decoding operations. Furthermore, the new view provided insight into cases when decoding is biased and approaches for reducing bias. The certainty measures of the channel decoding were arranged into measures of evidence and coherence.
Figure 4.10: Comparison of recurrent procedures for reducing bias in mode estimation by channel representations. A number of samples from a Gaussian distribution is channel encoded using a representation with seven channels, and the mean channel vector is obtained. The channel vector is processed by a recurrent procedure and then decoded. Experiments are repeated for 400 different positions of the Gaussian (true mode). For each trial, the estimated mode is marked at the x-position of the corresponding true mode, i.e. the ideal result is a diagonal line. The left column uses 200 samples and a standard deviation of the Gaussian 0.5 times the decoding interval length. The right column uses 1800 samples and standard deviation 1.5 times the decoding interval. The top row is direct decoding of the mean channel vector, the mid row uses 10 Deneve iterations before decoding and in the bottom row, the proposed recurrent procedure is iterated until convergence before decoding.
Part II

Learning Methods
Chapter 5

Introduction to Machine Learning

This chapter provides a short introduction to machine learning concepts which will return in the remainder of this work. The machine learning field in general is significantly broader and out of scope. A more general overview is provided in popular sources such as the books by Simon Haykin [43] and Christopher Bishop [8]. This chapter is primarily adapted from the work of Öjäll [71].

Machine learning relates to everything from finding parameters in pre-defined mathematical models, through curve fitting and general subspace fitting, to optimization or finding structure in data sets. The latter, finding structure in often high-dimensional data is often referred to as unsupervised learning. This can be used e.g. to reduce dimensionality while minimizing information loss.

Where machine learning appears in this work however, the purpose is often to train a system to produce a certain output for each input provided to the system. The desired output is not in general directly related to the input such as a lower-dimensional representation of the same, but the desired output is something externally defined. An example is when driving and encountering a red octagonal piece of metal, the action of stopping has nothing to do with the metal object itself (the stop sign) but it is a convention (traffic rule) agreed upon externally which the learning system (driver) has to learn. Depending on the type of training, this is often referred to as supervised learning or reinforcement learning.

Using supervised learning, pairs of inputs and the corresponding desired outputs, i.e. training data, are provided to the system, related to learning from demonstration. Using reinforcement learning on the other hand, correct outputs are not provided to the learning system but should be explored. Training is provided by means of performance feedback such that the learning system should alter its predictions to increase performance. Often, performance feedback is not available to the system after each prediction but only after a sequence of predictions.

The representation of inputs and outputs is related to the machine learning problem. Using representations suitable to the problem at hand may simplify the learning problem. In deep learning approaches, the representation is learned
In biological systems, learning is common on several levels. Some studies have become very well known. On an individual level, Pavlov studied conditioning in dogs [85] where a response of one stimuli is transferred to another, unrelated stimuli. By repeatedly combining the unrelated stimuli with presenting food to the dogs, the salivation reaction, originally only seen as a reaction to food, was after some training possible to activate by only using the unrelated stimuli.

On a higher level, the work of Darwin [22] is full of examples where a population of individuals may learn by the appearance of new individuals, different from the previous individuals in the population, rather than changes of the individuals themselves. Similar ideas are applied quite directly in optimization by genetic algorithms, where a pool of slightly different algorithms are generated and each is evaluated on a particular task. The best algorithms are kept and combined to generate new algorithms for refilling the pool. For enabling escape from local optima, a slight amount of random changes is applied, corresponding to exploration in machine learning and to the natural variation of Darwin. One example of the use of genetic algorithms in machine learning was for generating coefficients and structure for an Othello playing artificial neural network [70]. By evaluating an evolving pool of these networks against increasingly better hand constructed computer program opponents, the learning pool of networks developed increasingly complex game strategies which could be identified with the strategy development in a human player.

The studies of Pavlov and Darwin on learning in biological systems were more concerned with answering questions such as if there is learning at a certain level and what the effect of this learning is. For constructing artificial learning systems, theories on how learning can be implemented biologically are of higher interest. One such idea was presented by Donald Hebb in his 1959 book The Organization of Behavior [44]. The idea, now known as Hebbian learning, was only a small part of the book but had a strong impact on ideas for creating artificial learning systems. Very simplified, the idea is that neurons which are often simultaneously activated get stronger connections.

5.1 Properties of Learning Methods

There are several different ways of classifying learning methods, one of the more popular is by the type of training, such as supervised or unsupervised learning. Here, a selection of different properties is presented. Learning systems can be classified as having a particular property or not. The presentation here is primarily considering supervised learning systems, although these properties can be applied to unsupervised methods also. The properties presented here are not mutually exclusive, a learning system may possess several properties.

5.1.1 Online Learning

An online learning system is constructed in such a way that new training data can be incorporated without retraining the full system [92]. In literature, the
5.1. PROPERTIES OF LEARNING METHODS

precise requirements for a system to be an online learning system are varying. In this work, an online learning system is required to fulfill the following set of requirements, both during training and prediction:

- The method should be incremental, that is, the system should be able to incorporate a new training sample without access to previous or later training samples.

- The computational demand of incorporating a new training sample or making a single prediction should be bounded by a finite bound, especially, the bound shall be independent of the number of training samples already presented to the system.

- The memory requirement of the internal model should be bounded by a finite bound, especially, the bound shall be independent of the number of training samples presented to the system.

The first requirement is common among all definitions of learning systems. The last two requirements vary in their formulation. Some sets of requirements put no constraint on memory or computational complexity, others allow a logarithmic increase in complexity with more training data.

At the other end of the scale are the batch learning systems, where all training data are collected first. The system is then trained using the full batch of training data. There are also learning systems in between these two extremes, such as systems where training data are processed sequentially, but with many passes over the data, possibly with randomly permuted order.

The main advantage of an online learning system is the possibility to provide new training data without a full retraining of the system. If the switch between training and prediction mode is fast enough, new training data can be provided as soon as such a need is identified. When the additional training data collection is finished, the system can immediately proceed with new predictions under the influence of the new training data.

5.1.2 Active Learning

With active learning, the learning system is able to affect the generation or selection of training data in some way. There is no common and precise definition of active learning but more of a general idea with different interpretations. Settles has composed a survey [99], illuminating different aspects of the subject.

The primary case of active learning in the present work is the learning of inverse kinematics [77]. Every predicted output from the learning system, a vector of joint angles, will provide a pose vector of the robotic arm. That pose vector is the corresponding input to the learning system which the system should map to the joint angle vector just considered. The two vectors thus constitute a training data point. By exploring the space of joint angles, the system can affect the generation of training data.

Learning for controlling a labyrinth game presents similar challenges [72]. In both cases there is the choice of how to carry out the exploration. The simplest
approach is using random exploration. By exploiting the partially trained learning system, more directed exploration can be performed [93, 3, 18]. In the labyrinth game, exploration is performed by using a PID-controller. For the inverse kinematics framework, exploration is obtained by numerically minimizing the pose error using the partially trained model for estimating derivatives.

5.1.3 Multiple-Hypothesis Learning

A learning system with multiple-hypothesis output is able to learn one-to-many and many-to-many mappings. Many learning systems are constructed and limited to produce one single unimodal answer [32], i.e. learning one-to-one or many-to-one mappings. If training data exist where the input is the same but the output is different, some combination, such as the average, is generated in response to the specific input.

In some cases, the multiple hypotheses or modes should be separated. At a T-junction where there is possible to go either left or right, the two modes should be separated for the system not to generate the average of the two possibilities: straight ahead. The multiple-hypothesis ability of the learning system is dependent on the representation of the output.

Figure 6.10 on page 97 illustrates the response of several machine learning systems after training using multimodal training data. For each input value, there are two possible output values, e.g. an input of 0.5 maps either to output 0.25 or output 0.5. Training data points are indicated by crosses, for each input value, one mode is stronger than the other, i.e. more training samples are drawn from it. The strongest mode changes at an input value of 0.5.

Five learning systems are trained using these samples and the outputs generated by the system in response to inputs between 0 and 1 are plotted in the figure. The systems are linear regression, Support Vector Regression (SVR) [16], Real-time Overlapping Gaussian Expert Regression (ROGER) [40], Random Forest Regression (RFR) [66] and qHebb [75].

The unimodal methods (linear regression and RFR) tend to generate averages between the two modes (linear regression) or discontinuous jumps between the modes (RFR). The multimodal capable methods (SVR, ROGER and qHebb) selects the stronger mode in each region. Support vector regression is however a batch learning method. ROGER is slightly affected by the weaker mode and slows down with increasing number of training samples. In general, there is a shortage of fully online multimodal learning methods.

5.2 Learning Methods and Training Data

The learning method is related to the type of training data available. Preferably, a data set should be available with pairs of inputs and the corresponding correct (or desired) outputs. The set should cover all possible variations of all interesting cases the system may ever encounter and the number of examples of each case should be balanced. This is not always the case, the selection of training data
may be biased towards more common cases and there may be no examples of rare cases. Furthermore, the correct output may not be available at all.

If the desired output cannot be presented to the system, a reinforcement learning approach may be taken if at least some sort of correctness feedback of the predicted output can be provided. It is clear that a data set with desired outputs contains more information than performance feedback since a performance feedback value can be generated by comparing the actual output to the desired output, but not the other way around. Reinforcement learning is often applied in an active learning scenario, where old predictions affect future input to the system. In such cases, a pre-collected data set may not contain examples of the new input.

In some cases, where reinforcement learning is applicable, the performance feedback or error is easily obtained when an output prediction is available. If the feedback calculations can be incorporated into the learning system, a self-supervised system is possible. One such example is inverse kinematics learning. When the joint angles have been predicted, the error feedback is easily found as the difference between the resulting end effector pose and its desired pose.

On the other hand, for the task of autonomous road following, determining the correctness of a predicted action is comparable in difficulty to the task itself. The evaluation system should determine how well the road is followed unless a very coarse discrete feedback is used. Information regarding if the vehicle has crashed or not can be obtained from present on board car systems, however, a slightly more fine-grained feedback may be desirable.

5.2.1 Explorative Learning

Explorative learning, or self-supervised learning, is related to reinforcement learning for cases where the evaluation of a prediction from the learning system can be performed easily in comparison to the task to be learned. In such cases, the learning system may explore the space of actions itself. A related example, closer to traditional supervised learning, is an experiment where a robot has learned visual tracking of objects and uses this in combination with a laser scanner to learn object following using only the new sensor [21].

Self-supervised learning differs from unsupervised learning [43] in that unsupervised learning usually concerns dimensionality reduction, finding structure in the input data. There is no output or action to be predicted. The output from an unsupervised learning system is a new, often lower-dimensional, representation of the input space.

There has been a range of exploration strategies proposed, fully random approaches such as motor babbling can successfully explore the action space. If there is a specific target to be reached, fully directed exploration can be performed using e.g. gradient descent [74], or a combination with local random exploration such as directed motor babbling [93].

Explorative learning has successfully been used in applications such as learning inverse kinematics [74], where the feedback signal is the distance from the current pose of the end effector to the desired pose. Explorative learning has also been applied for a learning controller for a labyrinth game [72].
A directed method may approach the goal quicker, but alternative solutions may be missed. This is related to the exploration versus exploitation dilemma of reinforcement learning. The best area in action space to explore next may also be unclear, or as expressed by Schaal: falling down might not tell us much about the forces needed in walking [95].

5.2.2 Learning from Demonstration

Learning from demonstration is one variant of supervised learning. The term is more commonly used in situations where the learning system is supposed to learn to operate another system, such as learning how to drive a car along a road. Training data is collected while the teacher performs the task to be learned. The teacher is typically a human operator and while the demonstrated actions can be expected to be correct, the actions may not be optimal, disturbed by noise and there may be different solutions in similar situations.

For the autonomous road following example, determining the correctness of a predicted control command is comparable in difficulty to the task of road following itself. By considering the distribution of actions demonstrated in response to a certain input, something can be said regarding possible solutions. Under the assumption of locally unbiased demonstration, action predictions can be generated as local decodings of each mode in the predicted distribution of actions. Since there is no external evaluation of performance, the learning system will pick up particularities in the driving style of the demonstrator.

Also the labyrinth game controller is able to learn from demonstration in addition to the explorative learning touched upon earlier [72]. In that case the system is able to determine the correctness also of the demonstrated actions in terms of how these actions alter the ball trajectory with respect to the desired trajectory. Thus, demonstrating bad actions will not have negative impact on the performance of the learning system while even short sequences of good actions provides for faster learning.

Learning from demonstration in these types of systems tends to generate training data with certain properties to which some learning methods may be susceptible. First, the training data is typically temporally correlated. For autonomous driving, there may be several frames of straight road before the first example of a corner is produced. For online learning algorithms operating on batch data, a random permutation of the training data has shown to produce better learning results [43]. This is however not possible in a true online learning setting.

Second, the training data is biased. While there is a vast amount of training data available from common situations, there may only be a few examples of rare situations. However, these rare situations are at least equally important as the regular situations for successful autonomous driving. Thus, the relative occurrence of certain perceptions and actions in the training data does not reflect their importance. For ALVINN, special algorithms had to be developed to reduce the training set bias by removing common training data before the learning system [4].

For offline learning systems, where training data is collected first and then the system is trained, there is an issue for systems deviating slightly from the
5.3. LOCALLY WEIGHTED PROJECTION REGRESSION

desired path. Usually, the training data only contain examples of the vehicle driving successfully on the road. However, due to various reasons, the vehicle may obtain a state slightly offset from the correct path where there are no training examples demonstrating a recovering maneuver back onto the correct path. This is an advantage for online learning methods as if, during autonomous operation, the vehicle deviates from the correct path, manual control can be re-acquired and a corrective maneuver can be carried out, while at the same time providing the learning system with training data demonstrating this correction [75]. Similarly, this is an underlying reason why correct offline predictions on test data does not necessarily mean that the learning system will be able to successfully complete the task in a real scenario.

5.2.3 Reinforcement Learning

Reinforcement learning [43] is a special type of learning scenario where the teacher only provides performance feedback to the learning system, not full input-output examples as for the learning from demonstration case. Generally, the teacher is seen as a component external to the learning system, in contrast to the self-supervised learning. Learning is in general slower as the learning system may have to try several actions before making progress. After a solution is found, there is a trade-off between using the found solution and searching for a better solution, known as the exploration-exploitation dilemma.

For the road following system presented among the applications, an immediate version of reinforcement learning can be used to increase the performance of the system. Still, initial training is performed using learning from demonstration to quickly obtain base level performance. The immediate feedback mode can then be used to obtain performance beyond the abilities of the teacher providing demonstration. If demonstration was available at the higher performance level, using only learning from demonstration would be preferable from a learning time point of view.

5.3 Locally Weighted Projection Regression

Locally weighted projection regression [108], LWPR, is a unimodal online learning method specially developed for applications where the output is dependent on low-dimensional inputs embedded in a high-dimensional space. LWPR is an extension of locally weighted regression [96]. The general idea is to use the output from several local linear models weighted together to form the output. Some applications presented later use this learning system, however despite the projection operation, there are issues with image feature spaces with thousands of dimensions or more.

The output $y_{dk}$ for each local model $k$ for dimension $d$ consists of $r_k$ linear regressors

$$y_{dk} = \beta_{dk}^0 + \sum_{i=1}^{r_k} \beta_{dk_i}^u u_{dk_i}^T (x_{dk_i} - x_{dk}^0) \quad (5.1)$$

along different directions $u_{dk_i}$ in the input space. Each projection direction and
corresponding regression parameter $\beta_{dki}$ and bias $\beta_{0dk}$ are adjusted online using partial least squares. Variations in the input explained by each regression $i$ is removed from the input $x$ generating the input to the next regressor $x_{dk(i+1)}$.

The total prediction $\hat{y}_d$ in one output dimension $d$

$$\hat{y}_d = \frac{\sum_{k=1}^{K} w_{dk} y_{dk}}{\sum_{k=1}^{K} w_{dk}}$$  (5.2)

depends on the distance from the center $c_{dk}$ of each of the local models. Normally a Gaussian kernel is used, generating the weights

$$w_{dk} = \exp \left( -\frac{1}{2}(x - c_{dk})^T D_{dk} (x - c_{dk}) \right)$$  (5.3)

where the metric $D_{dk}$ is updated using stochastic gradient descent on the prediction error of each new training data point. The model centers $c_{dk}$ remain constant. New models are created when the weights of all previous models for a new training sample are below a fixed threshold. The new model is centered on the new training sample. The distance matrix for the new model is initialized to a fixed matrix, which is a user selectable parameter of the method.

### 5.3.1 High-Dimensional Issues

Although the input is projected onto a few dimensions, the distances for the weights still live in the full input space. The online property of the method depends on convergence to a finite number of local models and of a limited number of projection directions within each model. In the experiment presented in [108], a one-dimensional output was predicted from a 50-dimensional space where the output depended on a two-dimensional subspace.

Using full 2048-dimensional feature vectors as input, each local model required some hundred megabytes of primary memory for the autonomous vehicle. Increasing the size of the initial local models by setting smaller entries in the initial $D$ parameter reduced the problem, however, for longer training times the dimensionality of the input space had to be reduced before using LWPR [77]. Furthermore, the method is unimodal such that actions corresponding to the same perception are averaged. This is an issue when the autonomous vehicle encounters an obstacle straight ahead and the training data contain examples of evasive maneuvers both to the left and to the right.

### 5.4 Random Forest Regression

The first version of the autonomous road following vehicle used random forest regression for learning. A forest [12] is a collection of decision trees where the output of the forest is taken to be the average over all trees. A decision tree is a, usually binary, tree where each inner node contains a test on input data. The tests decide a path through the tree. Each leaf contains either a class label for
classification trees or a regression model for regression trees. As the model in each leaf only has to be valid for input data related to that leaf, the model can in general be significantly simpler than a model supposed to be valid in the full domain. In [27], a model of order zero was used, the mean value of all training data ending up in the leaf in question.

There is a large collection of approaches for building trees and selecting split criteria [94]. For a forest, Breiman noted that best performance was obtained for uncorrelated trees, however, building several trees from the same training data tend to generate dependencies between the trees.

In 1996 bagging was proposed as an attempt to reduce inter-tree dependencies [11]. The idea is to use a random subset of the training data to build each tree in the forest. Later, this was generalized to form random forests, where a random parameter vector $\Theta_k$ is generated for each tree $k$ and governs its construction [12]. All $\Theta_k$ are independent and identically distributed. For bagging, $\Theta_k$ is a random vector with binary entries and as many elements as there are training data entries. Each element in $\Theta_k$ determines if the corresponding training sample is to be used for building tree $k$. Also split criteria and thresholds may be randomly selected, usually after normalizing the input data by removing the mean and scaling by the inverse standard deviation in each dimension [12].

In the original formulation, the output from the whole forest was taken as the mean output of all trees in the forest. Later, using the median was proposed [89], which was shown to increase regression accuracy for regression onto steering control signals [27]. Using the mean over the forest tended to generate under-estimated steering predictions.

Furthermore, the random forests described so far are not able to handle multimodal outputs. This is seen in figure 6.10 on page 97 where the prediction from the random forest regressor jumps chaotically between the two output modes present in the training data. However, extending the trees with multimodal capable models in the leaves and making suitable changes to the split criteria selection, it is possible to construct random forests which properly handle multimodal outputs.

5.5 Hebbian Learning

Hebbian learning is used in the later application examples. Although proposed a long time ago and quickly abandoned in its pure form, the simplicity of the method can be compensated for by using a more expressive information representation. For unsupervised learning, Hebbian ideas have survived in the iterative algorithm by Oja [81] for finding the eigenvector corresponding to the largest eigenvalue of a matrix.

The name Hebbian originates from the Canadian psychologist Donald Olding Hebb. In his 1949 book he proposed a mechanism by which learning can come about in biological neural networks [44]. The often quoted lines, referred to as Hebb's rule, read:

Let us assume then that the persistence or repetition of a reverberatory activity (or “trace”) tends to induce lasting cellular changes that
add to its stability. The assumption can be precisely stated as follows:
When an axion of cell A is near enough to excite a cell B and repeatedly
or persistently takes part in firing it, some growth process or metabolic
change takes place in one or both cells such that A’s efficiency, as one
of the cells firing B, is increased.
– Donald Hebb, 1949 [44]

Simplified and applied to terms of perception and action, this would imply that
for any perception repeatedly present simultaneously with a particular action, the
particular action will more and more easily be triggered by the presence of this
particular perception. This relates to the dogs of Pavlov; whose salivation action
was possible to trigger with perceptions not related to food.

For a technical system, one of the simplest examples of Hebbian learning is
a scalar valued linear function of a vector $x$ parameterized by a weight vector $w$
with synaptic strengths,
$$ y = w^T x. \tag{5.4} $$
Introducing a discrete time parameter $t$ and a set of training data $(x_1, x_2, \ldots)$,
a simple application of Hebb’s rule generates the synaptic weight update scheme
(Equation (8.37) in [43])
$$ w_{t+1} = w_t + \eta y_t x_t, \tag{5.5} $$
where $\eta$ sets the learning rate.

Direct application of the learning rule (5.5) would lead to unlimited growth
of the elements in the weight vector. This can be mitigated by introducing a
normalization in each step
$$ w_{t+1} = \frac{w_t + \eta y_t x_t}{\sqrt{(w_t + \eta y_t x_t)^T(w_t + \eta y_t x_t)}} \tag{5.6} $$
which, assuming a small $\eta$, can be simplified to
$$ w_{t+1} = w_t + \eta y_t (x_t - y_t w_t). \tag{5.7} $$
This relation, known as Ojas rule, was shown by Oja [81] to converge to the largest
principal component of $x$, that is $w$ converges to the eigenvector corresponding to
the largest eigenvalue of the correlation matrix $E[xx^T]$ under the assumptions of
zero mean distribution of $x$ and the existence of a unique largest eigenvalue. By
removing the projections of the input vectors from the input vectors and learning
a new $w$, the second principal component can be obtained, and so forth.

The above example is unsupervised learning; the outcome does only depend
on the input. There is no prediction of any output. A similar approach can be
used for linear regression. Assume a linearly generated output $y$ depending on the
input $x$ and the fixed parameters in the vector $\beta$,
$$ y = \beta^T x. \tag{5.8} $$
Let $(y_n, x_n)$ be training data pairs fulfilling $y_n = \beta^T x_n$ for $n = 1, 2, \ldots, N$. Furthermore, let
$$ w = \frac{1}{N} \sum_{n=1}^{N} x_n y_n = \frac{1}{N} \sum_{n=1}^{N} x_n (\beta^T x_n), \tag{5.9} $$
5.6. ASSOCIATIVE LEARNING

That is, \( w \) is a weighted sum of the \( x_n \), where each term lies in the half-space \( \{ z : \beta^T z \geq 0 \} \). Given certain symmetry conditions on the distribution of the \( x_n \), it is geometrically clear that \( w \) will tend to be parallel to \( \beta \) for increasing \( N \) since the total contribution from the \( x_n \) orthogonal to \( \beta \) will remain small compared to the total contributions along \( \beta \). This is seen as each \( x_n(\beta^T x_n) \) will contribute a step in the positive \( \beta \) direction with probability one\(^1\) while the total contribution orthogonal to \( \beta \) is a random walk.

However, convergence is rather slow and the symmetry requirements on \( x_n \) limits the applicability of this direct Hebbian regression method. Convergence is illustrated in figure 5.1 for a 5D input space. For comparison, since there is no noise, the true parameter vector could be found from five linearly independent training samples by solving a linear equation system. From Hebb’s book it is also clear that the proposed learning mechanism is not supposed to be applied directly to the values of entities to be learned, but to some other representation of the entities where several neurons are used to represent each entity. The channel representation, or any similar population coding representation are such examples.

Similar approaches to Hebbian learning exist in literature. These have received more theoretical attention over the years. Two examples are Canonical Correlation Analysis (CCA) and Independent Component Analysis (ICA) [43, 9].

5.6 Associative Learning

Using associative learning, perceptions (inputs) \( x \) are associated to actions (outputs) \( y \). In this work, linear association is considered, such that actions and perceptions can be related by a linkage matrix \( C \),

\[
y = Cx
\]

(5.10)

The perception and action vectors represent the grade of activation of specific perceptions and actions and are non-negative. Typically, these are represented using the channel representation.

For finding \( C \), the traditional approach has been to minimize an expression on the general form

\[
\frac{1}{N} \sum_{n=1}^{N} \| y_n - Cx_n \|
\]

(5.11)

over a set of channel-represented training data pairs \( (x_n, y_n) \) and with a non-negativity constraint on all elements of \( C \). Different norms and minimization approaches have been evaluated, and extensions to allow incremental training have been explored [35, 49, 32].

Given a set of training examples \( (x_n, y_n), n = 1, 2, \ldots, N \), a Hebbian approach for learning \( C \) is the sum of outer products

\[
C = \sum_{n=1}^{N} y_n x_n^T
\]

(5.12)

\(^1\)Assuming no impulse in the distribution at \( x_n = 0 \).
Figure 5.1: Convergence of direct linear Hebbian regression in a 5D space on isotropic white Gaussian input data. Angle between estimated and true parameter vectors (radians) versus number of presented training samples. Top: average over 3000 trials. Bottom: one trial.
since for each outer product, each element in the matrix is large if both the corre-
sponding perception and action are activated in the current training sample. Con-
sidering partial sums $C_n = C_{n-1} + y_n x_n^T$, (5.12) is obviously an online method.
However, there are issues such as elements in $C$ growing without bound. These are addressed in chapter 6 on Hebbian associative learning.

Figure 5.2: Artificial neural network view of associative learning with $I$ inputs and $J$ outputs. The inputs $x_1, x_2, \cdots, x_I$ represent activations of visual features and the outputs $y_1, y_2, \cdots, y_J$ represent activations of actions. The synaptic weights are given by the elements of the matrix $C$: $c_{11}, \cdots, c_{J1}, c_{12}, \cdots, c_{JI}$.

The linear associative mapping (5.10) can be expressed as a single layer artificial neural network with linear activation functions where the synaptic weights are the elements in $C$, see figure 5.2. In this simplest form, only linear relations can be learned (see the regression example in figure 5.1). For learning more complex mappings, the approach of artificial neural networks and associative learning differ. Artificial neural networks employ non-linear activation functions and more involved structure with multiple neuron layers and possibly recurrent links. For associative learning, the linear mapping (5.10) is used but the representation of perceptions and actions is enhanced such that linear mappings on the new representation correspond to non-linear mappings in the original perception-action space. Using the channel encoding $C(\cdot)$ and decoding $C^\dagger(\cdot)$ operators, a mapping from $\mathcal{R}$ to $\mathcal{R}$ is expressed as

$$C^\dagger (C C (\xi)) \quad . \quad (5.13)$$
Chapter 6

Hebbian Associative Learning

This chapter presents a novel learning method based on the ideas of Hebbian and associative learning using the channel representation. In the ideal case, the full joint distribution of input features and output values is learned. However, in the general case with continuous inputs and outputs, the joint distribution cannot be represented by a finite set of parameters and therefore an approximate representation is needed. Limiting to second order approximations (Gaussian assumption or linear relations) is too restrictive in the case of outliers and multimodal distributions. Using mixtures of Gaussians such as Real-time Overlapping Gaussian Expert Regression (ROGER [40]) tends to slow down with increasing number of training samples [32].

For any distribution with limited support, histograms provide an approximate representation where computational effort is independent of the shape of the represented distribution and the number of bins directly controls the trade-off between computational demands and representation accuracy [98, 100]. However, the channel representation [37] provides higher representation accuracy given the same number of channels/bins compared to histograms [30]. The ideas developed here can also be applied for maintaining distribution-based image models for visual object tracking. This chapter is based on publications by Öfjäll, Felsberg and Robinson [75, 80, 79].

6.1 Introducing Hebbian Associative Learning

This section introduces Hebbian associative learning using a graphical approach where the basic principles appear more clearly. Further details are presented in the following sections.

The general idea is to obtain a linkage matrix $C$ mapping channel representations of the input $\mathbf{x} = C(\zeta)$ to channel representations of the output $\mathbf{y}$, from which the output is obtained by decoding $\hat{\eta} = C^\dagger(\hat{\mathbf{y}})$. The possibly non-linear mapping
\( \hat{y} = \mathcal{M}(\xi) \) is thus represented by
\[
\hat{y} = \mathcal{C}^\dagger \hat{y} = \mathcal{C}^\dagger \mathcal{C} x = \mathcal{C}^\dagger \mathcal{C} C(\xi)
\] (6.1)

Although the mapping of the channel vectors \( \hat{y} = \mathcal{C} x \) is linear, the non-linear decoding allows for non-linear mappings \( \mathcal{M}(\xi) = \mathcal{C}^\dagger (\mathcal{C} \mathcal{C}(\xi)) \).

A mapping where \( \eta \) and \( \xi \) belongs to \( \mathbb{R} \) can be illustrated in a plane where the input \( \xi \) is along the horizontal axis and the output \( \eta \) is along the vertical axis. This is illustrated in figure 6.1. The basis functions for representing input and output are drawn along the respective axis.

Figure 6.1: Illustration of basis functions for input representation (horizontal) and output representation (vertical). Both are represented using ten channels each. The first and last basis functions are centered around points outside the representable range.

Assume a set of training data with corresponding input and output pairs \((\xi_n, \eta_n)\). Individually channel encoding the input and output generates two channel vectors with at most three non-zero elements in each vector, \( x_n = \mathcal{C}(\xi_n) \) and \( y_n = \mathcal{C}(\eta_n) \). In this example, both inputs and outputs are represented using ten channels, scaled to represent values in the interval from zero to two. In general, the number of channels in the input and output representations can be different, resulting in a non-square linkage matrix \( \mathcal{C} \). Furthermore, the scaling and translation of the basis functions should be adjusted to fit the range to be represented.

The outer product \( y_n x_n^\dagger \), hereafter denoted \( \mathbf{D}_n \), will have nine non-zero values\(^1\) representing the position of the training sample in the joint space of inputs and outputs.

\(^1\)Unless any encoded value is located exactly on the boundary between the support of two basis functions.
outputs. This is illustrated in figure 6.2. Since ten basis functions are used for both input and output, \( D_n \) is a ten by ten matrix. However, the edge basis functions have their centers located outside the representable interval and thus figure 6.2(a) shows only the central eight by eight part of \( D_n \). Each square is centered on the centers of the corresponding basis functions in the input and output representations.

Similar to soft histograms, the position of the peak is represented with higher precision than the spacing of the basis functions. This can be seen by plotting a linear combination of all basis function pairs, weighted by the corresponding elements in \( D_n \). This is illustrated in figure 6.2(b), where the peak of the represented joint density is at the training sample.

![Figure 6.2](image)

**Figure 6.2:** Illustration of the outer product of a single training sample. In (a), the outer product matrix is shown using an inverse grayscale color-map where white corresponds to zero activation and black corresponds to maximum activation. (b) illustrates the represented distribution. The training sample is marked with a cross.

Combining several training data pairs

\[
C_N = \frac{1}{N} \sum_{n=1}^{N} D_n = \frac{1}{N} \sum_{n=1}^{J} y_n x_n^T = \frac{1}{N} \sum_{n=1}^{J} C(\eta_n)C(\xi_n)^T ,
\]

the linkage matrix \( C \) approaches a smooth representation of the joint distribution of inputs and outputs. This is illustrated in figure 6.3. Note that the relation between inputs and outputs cannot be described using mathematical functions as some input values are related to more than one output value.

Prediction, that is, estimating the output value \( \hat{\eta} \) given an input value \( \xi \) and previous training data, then becomes a matrix-vector multiplication of channel representations \( \hat{y} = Cx \). In terms of the probabilistic view, this corresponds
to finding the conditional distribution of the output given the input. This is illustrated in figure 6.4. Note that the output distribution represented by $\hat{y}$ has two modes, corresponding to the two different groups of examples seen during training. Using channel decoding, the strongest mode will be obtained. Furthermore, if the two modes have close to equal strength, their ordering can be controlled by applying a prior on the output distribution. The distribution representation of the output $\hat{y}$ is essential for representing multimodal (many-to-many) mappings. A video illustration of the prediction step is provided as supplementary material\(^2\).

If there is uncertainty regarding the input, this can be properly handled by the learning system. Let the input representation $x$ be a weighted combination of all current input options, or a representation of the distribution of current inputs. E.g. if, for some reason, the current input is $\xi = 0.75$ with probability 0.8 and $\xi = 1.5$ with probability 0.2, let the input representation be

$$
x = 0.8\, C(0.75) + 0.2\, C(1.5)
$$

(6.3)

From the linearity of the learned mapping in the representation space follows that the output representation will be the correspondingly weighted predictions. In the example above, the predicted output representation would thus be

$$
\hat{y} = 0.8\, \mathbf{C}(0.75) + 0.2\, \mathbf{C}(1.5)
$$

(6.4)

In general, the probabilistic interpretation corresponds to an integration of conditional distributions over a range of input values $\xi$.

\(^2\)http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125916
6.2. FROM HEBBIAN TO QHEBB LEARNING

The direct adaptation of Hebbian associative learning for online use

\[ C_n = C_{n-1} + D_n \quad \text{with} \quad D_n = y_n x_n^T \]  \hspace{1cm} (6.5)

does not limit the growth of elements in \( C \). Instead, applying a convex combination of the previous \( C \) and the update \( D \),

\[ C_n = (1 - \gamma) C_{n-1} + \gamma D_n \]  \hspace{1cm} (6.6)

growth is limited. By letting \( \gamma \) decay as \( \gamma = 1/n \), each \( C_N \) will be the partial sum \( C_N = \sum_{n=1}^{N} D_n \) of the \( N \) first training data pairs, giving equal weight to all training data samples. By instead letting \( 0 < \gamma < 1 \) stay constant, more weight is given to more recent training data, whereafter its weight decays exponentially.

In this section, the notation \( [\cdot]_{kl} \) is used to obtain a single element in a matrix, at row \( k \) and column \( l \). Each element in the linkage matrix \( C \) represents a link from a certain range of input values to a certain range of output values. Suppose that the element \( [C]_{kl} \) is large, then any input which activates the \( l \)th input channel will generate an activation of the \( k \)th channel of the output representation. Increasing the value of \( [C]_{kl} \) strengthens the connection (learning) while decreasing its value weakens the connection (forgetting).

During training, each element of the linkage matrix can be interpreted as the result of applying a first order filter with a pole at \( (1 - \gamma) \) to the sequence of values of the corresponding element in the outer product matrices \( D_n = y_n x_n^T \). The forgetting factor \( 0 < (1 - \gamma) < 1 \) thus also directly controls the learning rate.
CHAPTER 6. HEBBIAN ASSOCIATIVE LEARNING

Increasing $\gamma$ puts more weight on each new training sample. This accelerates learning when the corresponding element is activated by the current training data but also leads to faster forgetting when the current element is not activated.

Due to the symmetry of the response of linear filters, consider e.g. the positive and negative step function responses, it is clear that the learning and forgetting symmetry cannot be circumvented using any higher order linear filter. For asymmetric learning and forgetting, a non-linear filter is required.

Such a non-linear filter would offset a conventional Hebbian learning approach acting directly on data vectors. However, a non-linear but monotonic scaling function applied to all values of a density function will not offset the positions of density maxima, although the result may no longer be a valid probability density function. Thus, for Hebbian associative learning acting on channel representations of data, non-linearities can be tolerated.

A monotonic power combination function is proposed, with parameter $q \geq 1$, resulting in the qHebb learning equation

$$C_n = \left( \left( 1 - \gamma \right) C_{n-1}^q + \gamma D_n^q \right)^{\frac{1}{q}}$$ (6.7)

where matrix exponentiation is to be taken element-wise, i.e. $[C^q]_{kl} \equiv [C]_{kl}^q$ for all $k$ and $l$.

6.2.1 Properties of qHebb Learning

The elements of $C$ are bounded since element-wise

$$\min(C_{n-1}^q, D_n^q) \leq (1 - \gamma) C_{n-1}^q + \gamma D_n^q \leq \max(C_{n-1}^q, D_n^q)$$ (6.8)

and thus, since the power function is monotonic

$$\min(C_{n-1}, D_n) = \min(C_{n-1}^q, D_n^q)^{\frac{1}{q}} \leq \left( (1 - \gamma) C_{n-1}^q + \gamma D_n^q \right)^{\frac{1}{q}} \leq \max(C_{n-1}^q, D_n^q)^{\frac{1}{q}} = \max(C_{n-1}, D_n)$$ (6.9)

The elements of $C$ are initialized to zero and the elements of $D$ are limited by the squared amplitude of the channel basis function, with minimum zero and maximum $(\frac{2}{3})^2$ for mass-preserving $\cos^2$ basis functions.

Increasing $q$ emphasizes the larger element in each pair of corresponding elements in $C$ and $D$. Compare this to the $L_q$ family of vector norms, where for larger $q$, large elements in the vector attain higher importance. This continues up to the maximum norm $L_\infty$, which selects the largest element in the vector. The corresponding effect applies to qHebb learning, if $[D]_{kl} > [C]_{kl}$, i.e. the current training sample is dominating, increased $q$ leads to faster learning. On the other hand, if $[D]_{kl} < [C]_{kl}$, increased $q$ leads to slower forgetting. Using both $q$ and $\gamma$, learning rate and forgetting rate can be set independently.
Linear Hebbian associative learning (6.6) is obtained as a special case when \( q = 1 \). Furthermore, maxHebb learning

\[
C_n = \max(C_{n-1}, D_n)
\]

is another special case obtained for \( q \to \infty \). This leads to immediate learning and no forgetting. At this extreme, the parameter \( \gamma \) has no effect.

Using any \( q > 1 \), the direct interpretation of \( C \) as a representation for a joint distribution is lost. Instead \( C \) can be interpreted as a representation for an activity normalized distribution, where low densities are amplified and high densities are dampened. Increasing \( q \) thus have a mitigating effect in cases where biased training sets poses problems. For large \( q \), modes in the training data with relatively few samples obtains an increased effect on the learned mapping, see figure 6.5. If this effect is desired or not depends on the application and the reason for the bias in the data set. The effect of biased data sets can also be increased by using \( 0 < q < 1 \), reducing the effect of rare training samples.

Figure 6.5: Illustration of the represented mappings learned using qHebb with different \( q \) on a biased training data set. Increasing \( q \) normalizes the relative strength of the two modes.

### 6.3 Coherence in Hebbian Associative Learning

In this section, the coherence and evidence of section 4.3 are again considered, but in the light of Hebbian associative learning. These concepts can be expanded to the linkage matrix \( C \).

Any input feature often activated simultaneously with a certain output value will have high evidence, i.e. at least one large coefficient in \( C \). However, there may be input features which are always activated, independent of the output
value. Their evidence will be high but such an input feature will be useless in terms of predicting the output. In terms of the linkage matrix $C$, the entire column corresponding to this input feature will have large values, see figure 6.6. During prediction, this will generate contributions to all coefficients of the output. Traditional Hebbian associative learning is based on evidence and thus valuable information regarding the output may drown in non-informative contributions from high-evidence input features.

Instead, consider the coherence of each column in the linkage matrix. Any constantly active input channel will provide low coherence contributions to the output prediction. On the other hand, any input which is only active for a certain small range of output values will generate a sharp, high-coherence, peak in the linkage matrix. The coherence will be high even if this input feature is rarely activated in training data as the defined coherence is independent of evidence. Consider input values close to $\xi = 1$ in figure 6.6. Although such input maps to both $\eta = 0.5$ and $\eta = 1.9$, more information is gained regarding the output compared to predictions from $\xi = 1.6$. Both output peaks corresponding to $\xi = 1$ have high coherence. By rescaling the predictions of each input feature depending on the coherence, noise in the output representation can be reduced.

Expressed in a slightly different way, for a linkage matrix $C$, the coherence of each column is indicative of how precise statements regarding the output is made by the activation of the corresponding input channel. This can be used to reduce the influence of input channels producing vague (wide) output predictions. The coherence definition is thus extended to include the coherence of a linkage matrix.
6.4. INCORPORATING PRIORS

The distribution interpretation of the channel representation provides for a direct incorporation of priors. Priors are typically provided by some higher-level system for controlling the order of modes in a multiple-hypothesis output. In an autonomous road following application, a prior from the driver or a navigation system may select which road to follow at an intersection.

Mode selection priors belong to the absolute priors, adjusting the probability of ranges of output values. There is also the possibility of relative priors, where a higher-level system can affect the position of a mode relative to its current position. This pushes the output distribution in any desired direction.

6.4.1 Absolute Priors

Absolute priors are incorporated by considering the channel represented output of the associative learning, \( \hat{y} \), as a likelihood function. The posterior distribution is obtained by element-wise multiplication with the channel represented prior distribution \( y_\pi \), that is, the output prediction

\[
\hat{y}_\pi = C^\dagger (\text{diag}(y_\pi)CC(\xi)) \tag{6.13}
\]

is obtained.

The priors can be obtained by marginalization over subsets of the training data if the proper annotation exists. E.g. averaging the control representation of all cases where a car is making a left turn at an intersection, the control prior \( y_{\text{left}} \) is obtained. The obtained prior should be combined with its conjugate to allow for other options if the requested action is not available. The obtained prior \( y_{\text{left}} \) for example, would possibly completely remove all modes related to turning right. However, if the only available option is to turn right, the vehicle should be allowed to do that. The applied prior

\[
y_\pi = sy_{\text{left}} + (1-s)y_{\text{conj}} \tag{6.14}
\]

is a weighted combination of the prior obtained from training data and its conjugate. The parameter \( 0 < s \leq 1 \) controls the strength of the prior.
6.4.2 Relative Priors

The relative priors are applied when there is a need for a higher-level system to slightly adjust the output relative to the first prediction. In the road following application, this could include pushing the vehicle to adjust its speed slightly or making the vehicle keep a bit to either side of the lane.

Such an effect, pushing the entire represented distribution, cannot be accomplished by an element-wise product as for absolute priors. It requires a proper transformation of the channel space. Consider the decoding window of a channel representation and suppose the decoding value should be shifted from $\eta$ to $\eta + \Delta \eta$.

In the light of channel geometry in section 4.2, this corresponds to rotating the channel space around the symmetry axis of the cone.

With unit spacing of channel centers, the rotation angle should be

$$\alpha = \frac{2\pi}{3} \Delta \eta$$

(6.15)

For a different channel center distance, the rotation should be scaled accordingly. As noted in section 4.1.1, the rotation matrix

$$W = \frac{1}{\sqrt{3}} \begin{pmatrix} \sqrt{2} & -1/\sqrt{2} & 1/\sqrt{2} \\ 0 & \sqrt{3/2} & -\sqrt{3/2} \\ 1 & 1 & 1 \end{pmatrix}$$

(6.16)

from (4.8) aligns the cone symmetry axis with the third dimension. The sought transformation thus reads

$$W^T \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} W$$

(6.17)

and performing the multiplications, the circulant matrix

$$\frac{1}{3} \begin{pmatrix} 2\cos \alpha + 1 & 1 - \sqrt{3}\sin \alpha - \cos \alpha & \sqrt{3}\sin \alpha - \cos \alpha + 1 \\ \sqrt{3}\sin \alpha - \cos \alpha & 2\cos \alpha + 1 & 1 - \sqrt{3}\sin \alpha - \cos \alpha \\ 1 - \sqrt{3}\sin \alpha - \cos \alpha & \sqrt{3}\sin \alpha - \cos \alpha + 1 & 2\cos \alpha + 1 \end{pmatrix}$$

(6.18)

is obtained. This corresponds to circular convolution with

$$h = \frac{1}{3} \begin{pmatrix} 1 - \sqrt{3}\sin \alpha - \cos \alpha \\ 2\cos \alpha + 1 \\ \sqrt{3}\sin \alpha - \cos \alpha + 1 \end{pmatrix}$$

(6.19)

and the output representation $\hat{y}$ is obtained through

$$\hat{y} = h * CC(\xi)$$

(6.20)

Using more than three channels, the rotation operation should be carried out similarly on all decoding windows and the operation is no longer a convolution. However, convolving with $h$ approximates the correct operation also in higher-dimensional channel spaces. If the relative prior is the last operation before decoding, first decoding and then adding the offset $\Delta \eta$ is of course faster. The convolution operation is useful when the output distribution should be further processed, such as also incorporating an absolute prior.
6.5 Learning from Performance Feedback

The Hebbian associative learning framework can be extended to support an immediate form of reinforcement learning. The feedback is immediate in the sense that in the current implementation, the feedback is related to the last prediction of the system. Feedback thus has to be provided timely, similar to training a dog.

Two operations are required on the learned linkage matrix: strengthening of connections corresponding to correct predictions and weakening of connections corresponding to erroneous predictions. The feedback time window can be extended by storing the last $T$ inputs and outputs. The representation also allows for storing a filtered version of the input and output representations. This can be used to extend the feedback time window while limiting the memory needed for storing old inputs and outputs. Using a first order recursive filter, feedback is most strongly associated with the latest prediction, but also to previous predictions with exponentially decaying weights.

6.5.1 Positive Feedback

Positive feedback should strengthen the corresponding connection. This can be accomplished by means of the present learning framework. The input and predicted output representations are returned to the system as a learning from demonstration sample, section 6.2. The predicted representation $\hat{y}$ may contain multiple modes, however, only the mode corresponding to the final decoded prediction $\hat{\eta}$ should be preserved for avoiding reinforcing alternative modes. The prediction may be further processed before being fed back to the system to alter its behavior in a desired direction. This is related to incorporating priors, see section 6.4.

6.5.2 Negative Feedback

Negative feedback is accomplished by reducing the strengths of the connections in the linkage matrix $C$ producing the undesired mode in $\hat{y}$. Similar to the positive feedback case, the output representation is processed to remove any additional modes, forming the channel vector $\tilde{y}$. By an outer product with the input representation $x$, an indication is given which elements to reduce in the linkage matrix $C$. By multiplying each element in the linkage matrix with the corresponding element of $(1 - \lambda (3/2)^2 \tilde{y} x^T)$, the undesired connection is reduced without introducing negative components in $C$. The parameter $0 < \lambda \leq 1$ determines the influence of the feedback. The scaling factor $(3/2)^2$ is related to the mass preserving $\cos^2$ basis functions. Note that most elements of the correction factor matrix are one, leaving the corresponding elements of $C$ unchanged. This should be considered in an implementation to avoid unnecessary calculations, only elements corresponding to active channels in both input and output are affected.
6.6 Models for Visual Object Tracking

The channel representation and qHebb linkage matrix update ideas can be applied to patch-based visual object tracking. By representing each pixel in the target model patch using the channel representation, the model allows for more general changes in pixel intensity compared to first order approximations (storing the mean value of each pixel) or second order approximations (mean and variance).

A similar representation of the target model patch using histograms was proposed by Sevilla et al. [100]. The tracker performance was increased by replacing the histograms with the channel representation [29]. Applying ideas presented earlier in this chapter can further improve performance. In this section, qHebb learning is adapted for updating tracking target models. Furthermore, distance functions for comparing the target model with image patches of a new frame are presented. This section is based on a contribution to the visual object tracking challenge [79].

![Figure 6.7: Illustration of a pixel-wise channel representation (with \( K = 7 \)) of two images of canals. The top planes show the grayscale images while the lower seven planes indicate the activation of each channel (black: no activation, white: full activation). The lowest plane represents low image values (dark) while the seventh plane represents high image values (light). With permission of Springer.](image)

6.6.1 Representation of Tracking Target Models

The tracking target model representation is illustrated in figure 6.7. An image patch is represented by a three-dimensional array \( C \) with elements \( [C]_{ijk} \). The
first two dimensions correspond to the pixel indexing of a regular image. The third dimension indexes the channel coefficients, i.e., the representation is a matrix of channel vectors representing each pixel. Figure 6.9 motivates the distribution-based representation, where a few represented distributions of pixels of the target model after the final frame is shown. In the particular sequence, the target is moved over both dark and light background, resulting in the two modes of the background pixel distribution.

An encoded image patch from a new frame will be denoted $D$. The similarity in notation with linkage matrices for Hebbian associative learning is intentional. The similarity of tracking target model update and linkage matrix update will be obvious.

### 6.6.2 Target Model $q$-Update

Assume a tracking target model $C_{n-1}$ updated with information from the last frame $n-1$, and an encoded representation $D_n$ of the found image patch of the target in frame $n$. Transferring the qHebb linkage matrix update (6.6) to the model representation, the update equation

$$C_n = \left((1 - \gamma)C_{n-1}^q + \gamma D_n^q\right) \frac{1}{q}$$

is obtained, where array exponentiation is to be taken element-wise. The update function

$$g(C, D) = \left((1 - \gamma)C^q + \gamma D^q\right) \frac{1}{q}$$

will be used in the tracking application chapter.

All properties of the qHebb linkage matrix update are retained, such as the bounds on elements in $C$. Similarly, increased $q$ leads to faster adaptation to new information and slower decay of unactivated modes in the target model. Increasing $\gamma$ quickens both adaptation to new modes and decay of unactivated modes. The parameter $q$ adjusts the influence of low probability events, $q > 1$ increases the influence while $0 < q < 1$ reduces the influence. Regular linear update is obtained for $q = 1$. For $C_n$ to approach the true sampled kernel density estimates of the intensity distributions for each pixel, linear update and a time dependent learning rate $\gamma = 1/t$ is required. Using a fixed learning rate, more emphasis is given to more recent samples.

### 6.6.3 Model Comparison

The comparison function $d(C, D)$ should map two representations to a real non-negative value related to the dissimilarity of the represented information. A selection of possible functions was evaluated by Felsberg [29], including distance measures specifically intended for distributions. However, the $L_1$ norm of the element-wise differences gave the best tracking results.

Using the possibility to obtain variance (section 3.4) and coherence (section 4.3) of channel representations, the $L_1$ norm can be weighted to reduce influence of highly varying pixels. Typically, such pixels belong to the background.
The coherence weighted $L_1$ norm
\[
d_{\text{coh}}(C, D) = \sum_{i,j,k} |[C]_{ijk} - [D]_{ijk}| \left( \text{coh}([C]_{ij}) + \kappa \right) \quad (6.23)
\]
is similar to the coherence weighted prediction (6.12) where the parameter $\kappa \geq 0$ controls the influence of the weighting. The element at index $i, j$ is denoted \([\cdot]_{ij}^k\) while \([\cdot]_{ij}\) denotes the full channel vector corresponding to the pixel at $i, j$. The sum is over all elements in $C$ and $D$.

The variance weighted $L_1$ norm is
\[
d_\sigma(C, D) = \sum_{i,j,k} \frac{1}{\sigma_{ij}} |[C]_{ijk} - [D]_{ijk}| , \quad (6.24)
\]
where each $\sigma_{ij}$ is the standard deviation of the represented distribution of each channel vector $[C]_{ij}$ in the target model. These can be obtained by scalar products with a set of constant weight vectors, see section 4.3. The influence of the weighting can be reduced by introducing a modified denominator $\sigma_{ij} + \kappa$, $\kappa \geq 0$, similar to the coherence-based norm.

### 6.7 Summary and Contributions

This chapter introduced Hebbian associative learning on channel representations and the generalization to qHebb learning with linear Hebbian learning and max-Hebb learning as special cases. The basic qHebb algorithm is summarized in Fig. 6.8. Furthermore, a selection of extensions of the algorithm based on the probabilistic interpretation of the channel representation are presented. Introducing priors allows higher-level systems to affect the predictions. Using coherence-based weighting improves the signal to noise ratio in predictions. An immediate variant of reinforcement learning for associative learning is introduced. The ideas presented for associative learning are also applied to visual object tracking.

![Figure 6.8](image)

Figure 6.8: Summary of computational steps for qHebb learning and coherence weighted prediction with absolute prior.
6.7. SUMMARY AND CONTRIBUTIONS

Figure 6.9: Target and target model representation at the end of the VOT2013 cup-sequence. Left: found target patch. Middle: coherence of the target model (black: low, white: high). Right: represented pixel value distributions for a selection of points marked in left and middle images. Large coherence corresponds to static pixel values on the tracked object and narrow distributions (blue, magenta). Low coherence corresponds to background pixels (red, multimodal distribution) and varying pixels on the target (green, single wide mode). With permission of Springer.

Figure 6.10: qHebb, ROGER, SVR, RFR and linear regression on a synthetic multimodal dataset. During evaluation, the learning methods should choose the stronger of the two modes present in the training data. Unimodal methods tend to mix the two modes.
Chapter 7

Factorization and Non-Linear Channels

In higher-dimensional channel representations and learning, there are more possibilities regarding channel layout and associative mappings. In this chapter, a selection of alternative channel layouts is presented. The factorization approach is a complement to concatenation of encoded dimensions in the input space, where each input dimension imposes a constraint on the output. Logarithmic channel support expansion is explored for representing time such that time resolution is reduced for distant events. This reduces the number of channels required to cover a given time interval. Foveal image sampling patterns leads to log-polar layouts.

7.1 Subspace Factorization

For the case of multi-dimensional input, specifying a value in a single input dimension imposes a constraint on the output values. One typical example is the inverse kinematics mapping of a robotic arm. Given a coordinate system, specifying the desired x-position of the end effector adds a constraint on the joint angles of the robotic arm. Specifying desired y- and z-positions further narrows the set of possible joint angles. Using a probabilistic representation of joint angles, the solutions (set of joint angles) fulfilling the desired position in all three dimensions can be found by point wise multiplication of the representations of the solution sets for each individual constraint.

Such operations are supported by the channel representation and associative learning. In this section, these ideas are presented and illustrated on a low-dimensional robotic arm where the joint angle space can be illustrated in print. A further advantage of the associative learning approach is the possibility of using pose constraints with multiple options or ranges of allowed values. A gripping tool on the robot can be restricted to a given position, however it may be sufficient to limit the orientation of the tool only to a semi-sphere, not requiring a fixed orientation and allowing for other factors to affect the final pose.
7.1.1 Factorization of Associative Mappings

Let the full associative mapping be represented by $N$ partial mappings $C^{(n)}$, where each partial link matrix maps a constraint in one input dimension $n$ to the encoded representation of all output combinations fulfilling the constraint. The vector $\hat{y}^{(n)} = C^{(n)}x^{(n)}$ thus contains the channel representation of all points in the output space that are associated with the value $\xi^{(n)}$, encoded in $x^{(n)}$. By element-wise multiplication ($\circ$) of $\hat{y}^{(n)}$ for different $n$, the points in the output space that satisfy combined input space constraints are found. That is,

$$\hat{y} = \hat{y}^{(1)} \circ \hat{y}^{(2)} \circ \ldots \circ \hat{y}^{(N)} = C^{(1)}x^{(1)} \circ C^{(2)}x^{(2)} \circ \ldots \circ C^{(N)}x^{(N)} \quad (7.1)$$

is the channel vector of all points in the output space that correspond to constraints in all $N$ dimensions, i.e. a specific point in the input space.

For implementational convenience, the individual subspace linkage matrices $C^{(n)}$ and the input channel vectors are concatenated,

$$C = \begin{pmatrix} C^{(1)} & C^{(2)} & \ldots & C^{(N)} \end{pmatrix} \quad (7.2)$$

and

$$x = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(N)} \end{pmatrix}, \quad (7.3)$$

allowing the direct use of non-factorized training algorithms. The prediction step is replaced by block wise matrix-vector products followed by element wise multiplication of the partial results (7.1).

If the input space can be factorized in this way, the size of the concatenated linkage matrix is reduced, similar to concatenation of the representation of individual dimensions instead of using the Kronecker product. The use of concatenated channel vectors on the input side does in fact result in the same expressions for learning. The difference to the factorization approach lies in the prediction. Applying the standard association multiplication results in adding the representations of the individual constraints

$$\hat{y} = Cx = \begin{pmatrix} C^{(1)} & \ldots & C^{(N)} \end{pmatrix} \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(N)} \end{pmatrix} = C^{(1)}x^{(1)} + \ldots + C^{(N)}x^{(N)} \quad (7.4)$$

which should be compared to (7.1). The factorized approach is thus expected to generate more distinct output representations. A robotic arm with two degrees of freedom is used to illustrate the ideas and for comparing different approaches in a synthetic experiment.
7.1. SUBSPACE FACTORIZATION

7.1.2 Inverse Kinematics

The inverse kinematics of a robotic arm is a mapping from a desired pose of the manipulator to the angle of each joint of the robot. For a general robotic arm, the forward kinematics is not necessarily injective and the inverse kinematics may have several solutions. For a redundant robotic arm, the dimensionality of the joint space is larger than the dimensionality of the pose space. Given a desired pose, the solutions lie in a non-linear, possibly not connected, subset of the joint angle space.

The idea of subspace factorization is illustrated on a planar robotic arm with two degrees of freedom, two rotational joints. The joint angle space can thus be illustrated in a planar plot. This is the case in figure 7.3. The heat maps show the joint space of the two joint angles, i.e. the vectors \( \hat{y}^{(1)} \), \( \hat{y}^{(2)} \) and \( \hat{y} \) reshaped to matrices. Each of the two partial outputs show the one-dimensional manifolds in joint space where the end-effector of the robotic arm is limited to a horizontal or vertical line respectively. The Hadamard product shows the two configurations where the manifolds intersect. Both configurations will place the end-effector at the intersection of the lines.

7.1.3 Subspace Factorization Evaluation

A simulated planar robotic arm with three degrees of freedom is used for evaluating the subspace factorization approach. The simulated arm is similar to the previous example, with an additional link. The pose of the end effector is taken as the position of the end of the last link, i.e. the pose space has two dimensions and thus the robotic arm is redundant.

One million training points are generated randomly from a uniform distribution in joint angle space. Two thousand evaluation points are randomly generated uniformly distributed in the reachable pose space of the robot. Each method is evaluated by showing training samples one by one to the method. The incompletely trained systems are regularly evaluated using all evaluation points. For each evaluation point in pose space, \( p_j \), each system is used to calculate a solution to the inverse kinematics problem and the robot is moved accordingly, reaching a pose \( q_j \). The accuracy is assessed as the mean pose error over all \( J_e = 2000 \) evaluation points, \( E_{\text{pose}} = \frac{1}{J_e} \sum_{j=1}^{J_e} \| p_j - q_j \|_2 \).

For the associative learning methods, all dimensions are encoded in 16-dimensional channel vectors. The joint space is thus covered with \( 16^3 \) channels. In the outer product case, the pose space uses \( 16^2 \) channels while using concatenations, the pose space uses 32 channels. Training is carried out using encoded pose-joint pairs. For evaluation, the desired pose is encoded and mapped to the output representation \( \hat{y} \) either by (7.4) or according to (7.1) for subspace factorization (si). The strongest mode is decoded, obtaining joint angles for the three joints.

A selection of different associative learning approaches is evaluated, such as general qHebb as well as the special case of Hebbian learning (\( q = 1 \)). Stochastic gradient descent approaches are also evaluated, using different error norms such as Hellinger, Chi^2, L1 and L2. The gradient descent approaches are further described
elsewhere [32, 49]. Some traditional learning systems are also evaluated, such as Locally Weighted Projection Regression (LWPR) [26], ROGER [40], Support Vector Regression (SVR) [25] and Random Forest Regression (RFR) [12].

Figure 7.1 shows the results of several different associative learning methods when using full outer product representations of both input (pose) space and output (joint angle) space. For comparison, the results of a selection of common learning methods are also included. These do not use the channel representation of inputs and outputs. The batch learning methods are trained on different subsets of the training data set, the results of these are marked with crosses instead of lines. For the slowest methods, training and evaluation had to be aborted after eleven days, indicated by missing data for the largest training sets.

Most associative learning methods show a similar pose precision, with gradient descent Hellinger and qHebb outperforming the competing methods. The methods generating linear or independent models for each output dimension such as LWPR, SVR and RFR show inferior performance. This is suspected to be due to dependencies between joint angles.

When the input representation is concatenated, using 16 channels per dimension (32 in total), there are two options for combining partial results before decoding. Figure 7.2 shows the results of using conventional matrix-vector multiplication compared to using the factorized approach (7.1). For the concatenated pose space, precision differs more among the evaluated methods. The factorized approach, (si) in the legend, clearly is an advantage during the early training phase. qHebb with subspace factorization reach similar accuracy as for the outer product encoded pose space case. Using subspace factorization, the space complexity is reduced to the level of concatenated channels without affecting accuracy compared to outer product representations. Here, qHebb clearly outperforms regular Hebbian learning.

Further, the effect of the parameter $q$ in the robustness of the method is evaluated. Outliers are introduced in the training data. These are training samples with independent and random erroneous pose-joint-pairs. The results are shown in figure 7.4. The diagram shows the pose error as a function of $q$ and number of learning samples for four different outlier rates. As expected, higher outlier rates require more training samples to achieve the same accuracy. Higher $q$ lead to more robust results, including the case $q \to \infty$, which is maxHebb learning. What is surprising is that for low and intermediate values of $q$, the accuracy decays for large numbers of training samples, indicated by the red areas on the right. The learning rate parameter is set to $\gamma = \frac{1}{j}$, i.e. all training samples obtain the same weight in the trained system.
7.1. SUBSPACE FACTORIZATION

Figure 7.1: Evaluation results for learning associative mappings from pose space covered by $16^2$ channels to joint space covered by $16^3$ channels. For comparison, evaluation results learning the inverse kinematics mapping using ROGER, LWPR, SVR and RFR.

Figure 7.2: Evaluation results for concatenated pose space encoding, 32 channels. Subspace factorization is indicated by (si).
Figure 7.3: 2D example. Top left: encoded joint angles resulting in end-effector at $y = 3$. Bottom left: encoded joint angles resulting in end-effector at $x = -1$. Top right: Hadamard product of the two joint angle encodings showing the two possible joint configurations placing the end-effector at the intersection. Bottom right: robot illustrations of both solutions with lines $y = 3$ and $x = -1$.

Figure 7.4: Robustness to outliers over $q$ (vertical axis) and number of learning samples (horizontal axis). The input data for the four blocks contains 0%, 10%, 20% and 30% outliers (from top to bottom). The pose error for each combination is indicated by the color.
7.2 Time Logarithmic Channels

When the values to be channel encoded represent time, a non-uniform channel distribution may be beneficial. This allows representing recent events with higher precision than events more distant in time. The logarithm provides a constant expansion of the channels, the support of every channel is a constant factor wider than the support of the previous channel. One application example is pixel-based background models for static cameras. For representing spatial positions, the related log-polar channel layout is presented in the following section.

The basis functions for the logarithmic channels are

\[ b_k(\xi) = \cos^2 \left( (\log_d(\xi) - k) \frac{\pi}{3} \right) \xi^k, \quad d^{k-1.5} \leq \xi \leq d^{k+1.5} \quad (7.5) \]

and zero everywhere else. Using the base \( d \) logarithm, the parameter \( d \) determines the rate of expansion of the channels. The factor \( \frac{1}{\xi} \) keeps the total weight of each basis function equal, for higher \( k \), basis functions get wider and lower. The factor is obtained from a change of variables in the weight integral. Encoding a sequence of sufficiently frequent equidistant impulses using basis functions (7.5) results in constant channel coefficients. Normal and logarithmic basis functions are shown in figure 7.5.

Letting \( d = 2 \), each channel will be twice as wide as the previous channel. Instead letting \( d = 2^{1/3} \), the basis function support will be doubled every third channel, i.e. when a channel support ends, the new channel will be twice as wide, see figure 7.6.

7.2.1 Linear Approximation of Time Shifting

As new data arrive, previous information in the channel vector should be shifted in time. That is, given a channel vector \( \mathbf{x} \) with a few encoded events, a time shifted version is desired, corresponding to the encoding of each previous event with an additional time shift. this should be obtained without re-encoding the past events. For equally spaced channels, this corresponds to rotations in channel space or a shift in the coefficient vector in case the desired shift is a multiple of the channel spacing.

In case of logarithmic channels, the operation is more complex. However, a linear approximation has been sufficient and has in experiments introduced less errors than the time precision decay due to wider basis functions. In the experiments, the time shift matrix \( \mathbf{A} \) is obtained numerically by minimizing the difference between channel vectors \( \mathbf{Ax} \), and the corresponding true shifted versions of the \( \mathbf{x} \).

Using \( d = 2^{1/3} \) the adjacent non-overlapping channels double their size when using 3 simultaneously active channels. In this setting, a local and smaller time shifting operation may be applied. The local shift operation is then repeated over the next set of channels but at half the rate, and so on. This idea has not been further pursued.
Figure 7.5: From top to bottom: five regular channels, five logarithmic channels and five scaled logarithmic channels with constant area.

Figure 7.6: Layout of basis function supports using logarithmic channels and expansion parameter $d = 2^3$. Crosses indicate the channel support bounds and overlapping channel supports are distributed on the three lines.
7.2.2 Time Logarithmic Channel Background Models

All conventional pixel-based background models have some parameter controlling the update rate. Fast changes are considered foreground while slower changes are assumed to be due to changes in the background. Problems arise when foreground objects stop and remain stationary for shorter or longer periods.

Using logarithmic temporal channels, information regarding the whole history of an area in an image can be stored with very little effort but with reduced temporal resolution further back in time. Using time shifting, the background model can be updated without re-encoding previous frames. In figure 7.7, background modelling using logarithmic temporal channels is illustrated on a sequence from the PETS data set.

Gray scale values are encoded using conventional channel coding. The decoding of a few points at the end of the sequence are shown. The white van previously stopping in the lower right corner of the image is clearly visible, also the change in lighting conditions caused by moving clouds starting at the lower left corner at the end of the sequence can be seen. In this way, some fixed or varying update rate is not necessary. It is thus possible to answer questions such as for how long a certain image region has remained stationary, and how a certain region appeared a given time ago.

7.3 Log-Polar Channels

The log-polar arrangement of channels in a plane is based on the ideas of foveal vision, where higher resolution is available in the central parts of the visual field. Figure 7.8 illustrates an arrangement of channels in a log-polar grid.

A Cartesian image position \((\xi, \psi)\) is mapped to the log-polar grid \((r, \theta)\) by the principal branch of the complex logarithm \(r + i\theta = \Log(\xi + i\psi)\). The logarithmic radial position \(r\) and the angular position \(\theta\) are encoded in an outer product channel representation. The angular channels are modular, mending the branch cut of the logarithm function. Channel coefficients are scaled with a factor \(1/(\xi^2 + \psi^2)\) to maintain a constant weight of all basis functions, compensating for the polar coordinate system and the logarithm of the radial position. Note that the supported radial range is limited at both ends, avoiding an infinite channel density at the origin.

A grayscale image is encoded by summing outer products between the channel encoded pixel value and the channel encoded pixel position for all pixels in the image. Since pixel positions are constant, position dependent coefficients can be pre-calculated. Image information can be decoded by first using weighted spatial channel basis functions sampled at the pixel positions to obtain one full resolution image for each intensity channel. A decoded image is then obtained from regular channel decoding of each pixel. This is illustrated in figure 7.10.
Figure 7.7: Illustration of the information available in five pixels in a time logarithmic background model. The pixels are marked with crosses in the most recent frame of the sequence. In the plots for each of the five pixels, the y-axis corresponds to pixel intensity, where 0 is white and 255 is black. The x-axis is time (in frames) since the most recent frame. In the model, time resolution is higher closer to the present, while intensity resolution is constant.
7.3. LOG-POLAR CHANNELS

Figure 7.8: Example of basis functions using three radial and five angular channels. For clarity of presentation, the normalization factor is removed and thus the amplitude of all basis functions are the same. The sum of all normalized basis functions is shown in figure 7.9.

Figure 7.9: Sum of all basis functions, generating a flat surface on the disc-shaped representable range.
CHAPTER 7. FACTORIZATION AND NON-LINEAR CHANNELS

7.3.1 Visual Tracking Illustration

Using the log-polar image representation is illustrated on a simple visual tracking example. The operation of moving the central position of the log-polar grid followed by re-encoding the image is approximated by a linear operation directly on the previous channel coefficients, similar to time shifting in the logarithmic time representation. Since the high resolution area will be at a different part of the image after translation and only lower resolution information is available in the previous channel representation, obtaining the correct representation is obviously not possible.

For rotations of the image in increments corresponding to the channel spacing in the angular direction, the corresponding new channel coefficients are obtained through a circular shift of the old coefficients. Given a translation operator in one direction, translation operations in other directions can thus be obtained by a rotation followed by translation and the opposite rotation.

Given a set of translation operators for translation distances equally spaced in the logarithmic domain, translated approximations of the represented image can be obtained for translation points in a log-polar grid. By comparing these translated representations with the log-polar channel representation of the next frame, a representation of differences in appearance given different translations is obtained. Such representations of translation-dependent appearance differences are illustrated in figure 7.11. In this manner, more information regarding the local error surface is obtained.
Figure 7.11: Estimated difference between translated representations of one frame compared to the representation of the next frame, sampled on a log-polar grid and interpolated using log-polar channel basis functions. In the upper case, the precise translation is uncertain, however there is a strong indication that the tracked object has moved downwards in the image. In the lower case, the precise translation is more certain. The white markers indicate global minima of the error function with respect to translations.
7.4 Summary and Contributions

This chapter presented a selection of extensions to the channel representation and learning framework. All have not been fully evaluated in applications yet, but possible applications have been pointed out in the text.

The proposed factorization approach is especially useful in scenarios similar to inverse kinematics where the input space can be partitioned in an axis aligned manner. The learning step is similar to regular Hebbian learning on concatenated channel representations, only the evaluation step differs. The evaluation step returns a representation of solutions fulfilling all specified input constraints, some of which may be left unspecified.

The time logarithmic channel layout and time shift has been briefly evaluated on generating background models for static cameras. No larger scale application using the proposed layout has been created so far. There is no application of the foveal ideas of the log-polar channel layout, however it is used for bird tracking, section 9.2.
Part III

Applications
Chapter 8

Online Learning of Autonomous Road Following

This chapter presents an application of learning systems to learn vision-based autonomous road following. This has been the primary application while developing the Hebbian learning systems. The first autonomous systems used off the shelf learning methods such as random forest regression [12] and locally weighted projection regression [26]. The later autonomous systems are based on Hebbian associative learning, presented in chapter 6.

The use case scenario for the online learning systems is presented as a video among the supplementary material.\(^1\) Imagine driving on a backcountry road. The regular lane keeping system does not work due to lack of lane markers to follow. However, the online learning system processes visual information and your actions as a driver. After a minute, the system has learned the visual appearance of the current road and a driving style suitable for the current conditions. You can let go of the controls and let the learning system drive for you.

The demonstrator system is developed on a smaller platform. However, experiments are performed outdoors on real roads. Experiments are also performed in a more controlled lab environment which allows for more precise evaluation. The material in this chapter is based on a series of publications [27, 77, 75, 80]. After the introductory sections, the evolution of the demonstrator system follows in chronological order.

8.1 Introduction and Related Work

The road infrastructure is currently designed primarily for visual sensing, such as signs, traffic lights, road markings, other traffic and so on. By redesigning the road system specifically for autonomous vehicles, the demands on visual sensing may be reduced, e.g. as for autonomous forklift systems in warehouses. This would also allow systems like autonomous intersection management for increased traffic.

\(^{1}\)http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125916
throughput [15], however, this would require a majority of autonomous vehicles in the system and special solutions for manually driven vehicles. Furthermore, adapting the infrastructure for autonomous vehicles will most likely be an option for highways and primary roads. Thus, vision will stay the main sensor modality for autonomous driving on smaller roads for a long time. Smaller roads are also more challenging as their appearance varies to a larger extent compared to highways.

Figure 8.1: A selection of different roads encountered by the author within a small region of the earth. IEEE 2016.

Autonomous driving has previously been approached in different ways in the literature, which can be roughly divided into classical control-based approaches [24, 106, 104] and learning-based approaches [86, 87, 89]. Most approaches to autonomous vehicles employ a wide range of different sensors [65, 54, 34] and purely vision-based systems are rare, e.g. ALVINN. The latter required special attention to obtain a balanced training set, e.g. in terms of number of left turns, right turns and straight ahead examples [4].

DeepDriving [17] and visual off-road driving [64] are more recent examples of learning visual driving, although training is offline. The advantage of vision-based systems is that those systems will react to the same events as a human driver.
This reduces the risk of surprising actions due to events perceivable only by either the human or the computer driver.

The application presented in this chapter is a purely visual autonomous road following system using a slightly different approach. Online learning allows the system to adapt to previously unseen roads and driving conditions, initially using the human driver as a teacher. The system has demonstrated capability to follow different types of roads, with and without lane markers while most other systems rely on lane markers or offline training. The presented system can thus be used as a complement to traditional lane-marker based road following systems to extend the range of roads where autonomous operation is possible.

The primary challenge of a purely vision-based system for following general roads is their diverse appearance. A small selection of roads is shown in figure 8.1, and although the selection is small and geographically limited, visual appearance varies greatly. The system must learn to associate the presence of certain input features with the appropriate action. Since many features depend on the surroundings and not on the road itself, this makes the task more difficult. Features may also be activated in cases where different actions are required. At the extreme, the set of active visual features in two different situations are exactly the same but the actions are different, such as just before turning at an intersection. This is pointed out also by Chen et al. [17], human drivers may take different decisions in similar situations. However, using multi-hypothesis capable learning systems, such situations can be handled.

No prior information regarding traffic rules, road features or layout is provided to the system, not even the notion of a road is present in the system. This is demonstrated by teaching the same system to follow a person instead, without any change of parameters. The system is trained by manually driving the vehicle. Around one minute of training time is usually sufficient. The online learning system also allows further training on the fly. Switching between autonomous driving and demonstration is done without stopping the robot.

8.2 Evaluation System and Experimental Setup

The learning systems are evaluated on a platform based on a radio controlled car. A laptop and a single monochrome camera is fitted on the platform. The camera is mounted on a rotating platform and follows the steering. This was required to keep the road in view during tight turns. The laptop interfaces the servo controller on the car and the original radio control receiver. This enables car control from the laptop as well as providing external control signals to the laptop via the radio control transmitter. For outdoor experiments at night, lights were added to the system, see figure 8.2.

8.2.1 Experimental Scenarios

The experimental scenarios are of two different types, outdoor and indoor. Outdoor scenarios evaluate the system performance in real conditions on varying road
types and during different times of the day. Such experiments did not allow qualitative evaluation due to lack of ground truth and accurate positioning systems. The outdoor experiments are primarily presented by video recordings of the experiments themselves.

The indoor experiments allow accurate positioning in a controlled environment. These provide more precise performance measures. The controlled evaluation environment is based on a reconfigurable circuit made of tiles of carpet with sections of road markings. Using differently colored carpets, different road-styles can be created. Different circuits can be made by placing tiles in the desired configuration. Figure 8.5 provides some examples of circuits with similar layout but varying road-style. For the multimodal, i.e. multiple-hypothesis capable learning systems, the track layout is extended to include intersections.

The vehicle position is provided by a visual tracking system. Accurate trajectories are obtained by tracking a red marker attached to the vehicle. The dashed lines in figure 8.5e show the results of the tracker. By placing the vehicle with the red marker above the corners of each track section, a homography mapping the image position of the marker to the ground position of the vehicle is found. The ground position is marked with solid lines in figure 8.5e. Furthermore, the ground plane is projected onto a metric planar view, figure 8.5d, where axes are in meters. Linear interpolation of the trajectories is used when the red marker is occluded by the red beam.

The primary goal of each scenario is for the autonomous system to learn to follow the track, independent of the appearance of the current track. Specifically, there are no parameter changes or other changes allowed to the learning system when performing experiments in different scenarios. In some cases, where a fixed speed is used, a higher vehicle speed is used for outdoor scenarios. However, the learning system and all parameters are the same for both indoor and outdoor trials.

Figure 8.2: The autonomous road following system at night. IEEE 2016.
8.2. VISUAL INPUT

Visual input from a single monochrome camera is the only sensory data available to the system. The visual perception consists of a generic, holistic representation of the whole visual field, using Visual Gist [82]. The visual gist encodes the whole visual field into one feature vector, using multi-scale filtering to encode the visual scene.

There exist different versions of the Gist features. In all experiments, the feature vector is computed by filtering a downscaled (128 $\times$ 128 pixels) version of the image with Gabor filters tuned at different scales and orientations. The responses of each filter are averaged over Gaussian channels regularly spaced over the image. Four scales, eight orientations and eight-by-eight spatial channels are used, resulting in a feature vector of dimension 2048.

8.2.3 EXPERIMENTAL SETUP

All experiments are initiated by a learning-from-demonstration phase where correct driving behavior is demonstrated by manually controlling the vehicle. The early offline learning systems are then trained offline and their performance is evaluated in a new experiment. The later, online learning systems, are trained during the training data collection and are ready for evaluation as soon as the teacher lets go of the controls. Depending on the system, further training sessions, possibly using different training methods, may be initiated.

More specifically, the learning system should learn a mapping from the 2048-dimensional gist feature vector of each input image to the car control. In the first experiments, the car control to be predicted is the steering angle while the throttle is kept at a constant value. In the later experiments, both steering and throttle should be predicted.

Vehicle control is estimated directly from the visual input, frame by frame. This stops prediction errors to propagate over time but at the same time limits the learning system to static features of the images. Allowing the Gabor filters to also extend over time may capture motion information and is expected to increase performance. Motion features have not been necessary to achieve autonomous road following nonetheless, and the limitation provides insight into what can be achieved by simple frame by frame predictions. This is along the same lines as the Hebbian learning approach: obtaining impressive results using simple algorithms (instead of the opposite).

The system does not have or build a model of the track. This allows for navigation along potentially infinite paths. The visual input of the system represents the full visual information. The Gabor filters are general purpose filters not targeting the specific application of road following. It is the learning that specifies which aspects of the visual information that is relevant for control prediction. For the online learning systems, learning and predictions are performed in real-time while the system is driving. This also provides the possibility to correct any misbehavior or lack of training data by additional manual control, seamlessly without stopping the system.
CHAPTER 8. AUTONOMOUS ROAD FOLLOWING

8.3 Offline Learning of Steering

The first incarnation of the demonstrator system used offline learning, more specifically random forest regression [27]. The random forest learning system was able to successfully learn to follow different layouts of the indoor circuit. In addition to training data where the vehicle was driven manually around the track, additional training data demonstrating corrective action was required. Such corrective actions were captured by placing the vehicle slightly off the road and then collecting training data while it was driven back onto the road.

A second version for offline learning using the convolutional neural networks of LeCun [62] was evaluated by Schniterlöw [97]. The system was fast enough to produce predictions at video rate and the predictions made on offline test sets seemed to match the controls of the human driver, yet the system was unable to achieve stable road following.

The third offline learning system evaluated used the Caffe deep learning framework [47] and a pre-trained reimplementation of the AlexNet [58]. This work was carried out to obtain a comparison system to the latest online learning demonstrator [80] and was published along with it. The following part is based on that publication. The main work on the deep learning method was done by Robinson.

8.3.1 Deep Learning Experiments

The used deep learning network was originally designed for image classification but was adapted to control the vehicle. The top three layers, i.e. those initially responsible for object classification, were replaced by three layers of 128, 128 and 2 neurons. The two-neuron layer is the output, with one neuron responsible for steering and the other for throttle control.

The training parameters are left at their default values, with one exception, the batch size is reduced from 256 to 20 to fit in the GPU memory (2GB). Training is performed on 750 consecutive frames (50 seconds) of a sequence where the vehicle is manually driven counter clockwise around the test track. Images from the camera are recorded together with the steering and throttle commands used by the human driver. During training, the sum of squared difference between predicted and ground-truth steering and throttle was minimized. The initial error was 0.2 and when the training was terminated after 1500 iterations, the final error was 0.005. This error was not significantly lowered by further training. The throttle range is from 0 to 1 and the steering signal range is from $-1$ to 1.

The trained network is evaluated on a separate test set with 750 frames from the same recording scenario as used for training. A qualitative result can be observed in figure 8.7. It shows the steering and throttle control of 200 frames excerpted from the test sequence. The manual control signals are compared to the predictions of the network. Although the vehicle is driven manually, the predictions of the network look plausible. However, this is no guarantee that the system would function in an online driving scenario.

Unfortunately, the trained network was unable to run online as it was too computationally demanding for real-time use, even on the Linux-based desktop.
computer used for training. The desktop GPU (Nvidia GeForce GTX 560 Ti) processes approximately 12 frames per second, which is already below the 15 frames per second output of the camera. The notebook-class GPU onboard the vehicle (Nvidia GeForce GTX 480M) is slower. When Caffe is configured to evaluate the network on the desktop CPU (Intel Core i7 920 at 2.67 GHz), the throughput drops to approximately 0.6 frames per second. These experiments are listed among the others in the summary table 8.1.

8.4 Learning of Unimodal Steering Control

The first online learning approach to learning autonomous road following uses locally weighted projection regression, which had successfully been used for learning to play the labyrinth game [72]. The projective part of the learning system is expected to reduce the dimensionality of the input vector and in this way reduce the computational burden. The experiments and results presented in this section have been published as a chapter in the book New Development in Robot Vision [77].

The learning system was unable to process the 2048-dimensional feature vectors at video rate. The primary reason for this was found to be the 2048 by 2048 matrices determining the size of each local model. Before feeding the visual feature vector to the learning system, the dimensionality is thus reduced from 2048 to 256 by means of principal component analysis (PCA). The principal components are calculated from the feature vectors of a sequence where the vehicle was manually controlled one lap clock-wise and one lap counter clock-wise on the track in figure 8.3. The same PCA dimensionality reduction was used for driving outdoors, indicating that the reduced feature vector still allows some generality although obtained from indoor training data. The supplementary video is available online.²

8.4.1 Results

Figure 8.3 shows the trajectory for the first five laps of a previously untrained system. The robot is manually controlled through six corners (dashed blue line) whereafter the controls are released and the robot continues autonomously around the track (solid magenta line). As the trajectory in the lower part of the track was not considered accurate enough, manual control was used to override steering predictions through two corners (dashed black line). The new training data correct the undesired behavior and during the following laps, the robot stays on the road (solid cyan line).

After the first five clock-wise laps, the robot is manually turned around for counter clock-wise laps. These are shown in figure 8.4. The robot is manually con-

²http://users.isy.liu.se/cvl/ofjall/onlinePerceptionActionLearningSmaller.mp4
Figure 8.3: Trace of the first five clock-wise laps for a previously untrained system overlaid on an image of the track reprojected onto the ground coordinate system, with coordinates in meters. With permission of Springer.

Figure 8.4: Trace of the four laps following Fig. 8.3 where the robot runs counter clock-wise. With permission of Springer.
8.4. LEARNING OF UNIMODAL STEERING CONTROL

Figure 8.5: Autonomous trajectories by the robot while reconfiguring the track (cyan), with last laps on unmodified track for comparison (magenta). With permission of Springer.
Figure 8.6: Steering control during initial sequence, $-1$ is max right, $1$ is max left. With permission of Springer.

Figure 8.7: Results from batch training using a CNN approach [47] in comparison to manual control signals. Top: results for throttle; bottom: results for steering. IEEE 2016.
8.5 Learning of Multimodal Steering Control

Using the proposed qHebb associative learning on the channel representation allows for multiple-hypothesis predictions for steering. The system is primarily evaluated on a track with intersections. The results in this section have previously been presented at the British machine vision conference [75].
For the channel representation, 7 channels are used for each element of the 2048-dimensional feature vector. The channel representations are concatenated into a channel vector with 14336 elements. The output steering signal is represented by 15 channels and qHebb is used to learn a linkage matrix from input features to the steering representation.

The proposed coherence weighting, section 6.3, is essential as many visual features are always activated or show activation patterns independent of the appropriate steering signal. After training, columns in the linkage matrix corresponding to those features are constant and hence activates all output channels when such an input feature is activated. Only a few features show activation patterns dependent on the appropriate steering signal. Without coherence weighting, the activation from steering-dependent features drowns in the noise of activations from features independent of steering. The non-linear $q$-update, section 6.2, is required for reducing the effects of unbalanced training data e.g. from differences in the numbers of left, right and straight ahead examples.

8.5.1 Results

The experimental results are summarized in the supplementary video.\footnote{http://users.isy.liu.se/cvl/ofjall/bmvc2014.mp4} A lower resolution video is available at the BMVC web site of the publication.\footnote{http://dx.doi.org/10.5244/C.28.94} Table 8.1 presents an overview of approaches for autonomous road following.

The learning system using qHebb learning successfully learns to follow the roads. The layout of the track is changed while the vehicle is operating autonomously, this indicates that the system has learned to follow the road, not other visual features available in the surroundings. Using a layout with intersections, the multiple-hypothesis ability of the system is tested. After driving both right and straight at the intersection, the predictions contain both modes, however, the vehicle either goes straight or turns right. The two modes are never mixed. In the following section, the method for incorporating priors is evaluated, enabling a higher-level system to select mode where there are multiple modes.

8.6 Multimodal Steering and Throttle Control

Here, the autonomous road following system is extended to also include throttle control. Furthermore, the possibility of using external and internal feedback learning is introduced together with higher-level control by control priors. This section is based on the publication [80].

There are two major reasons for introducing learning from performance-feedback, i.e. immediate reinforcement learning. The first is the ability to go beyond the level of the teacher. In an analogy to the athlete world, top athletes perform better than their coaches. While learning from demonstration is effective during the early years, some other method of training will be required for the athlete to progress beyond the level of the coach. The second reason is that providing performance-
feedback, i.e. determining if the system performs well or not, is much easier for the teacher than providing correct demonstration examples.

Introducing soft priors enables a higher-level system, such as a passenger or navigation system, to provide requests to the system. A request such as: turn right on the next occasion where it is possible, can be represented as a prior favoring right turn modes.

Learning from demonstration, learning from performance-feedback and imposing priors can be used interchangeably and at any time, thus providing a more intuitive Human-Machine Interface. That is, the teacher can in every instance and at her own discretion, choose the most appropriate modality for training. Earlier artificial learning systems have used a single type of learning to which teacher had to adapt.

The goal of the learning system is slightly modified in this case. The learning system should learn a mapping $M : \mathbb{R}^m \rightarrow \mathbb{R}^n$, $\xi \mapsto M(\xi)$, mapping visual input $\xi$ to vehicle control $M(\xi)$. The training samples are on the form $(\xi_j, (\eta_j | r_j | \emptyset))$, where the notation $(\eta_j | r_j | \emptyset)$ indicates that either the desired output $\eta_j$ (learning from demonstration), or performance-feedback $r_j$ (good, neutral or bad), or no external input $\emptyset$, is provided with the visual input $\xi_j$ for video frame $j$. Here, $m = 2048$ and $n = 2$, steering and throttle.

As for the previous system, 7 channels are used for each element of the 2048-dimensional visual feature vector. The channel representations are concatenated into a channel vector with 14336 elements representing the input to the learning system. The control signals are represented using a 7 by 8 outer product channel representation, i.e. output is represented by 56 channels. The steering signal magnitude is mapped by $s \rightarrow s^2$ prior to encoding and the inverse function is applied after decoding, resulting in higher channel density close to zero steering angle. Steering magnitude is represented by a real number between 0 (straight) and 1 (full turn), the sign of the steering signal determines the direction.

As the throttle signal is used in training, the release of the throttle on the remote control transmitter cannot be used as an indication for the system to switch from learning mode to autonomous mode. A separate switch on the transmitter is used for this purpose. There are three different modes, manual control without learning, manual control with learning and autonomous mode. In autonomous mode, performance-feedback can be provided using the throttle control. The throttle range is discretized into three feedback levels, negative, no feedback and positive. Steering priors are provided using the steering control while the vehicle is in autonomous mode. Steering priors are also discretized into three levels: left prior, no prior and right prior.

### 8.6.1 Self-Feedback by Road Prediction

In the present system, means for self-feedback is explored using road predictive coding [90]. The feedback system is based on the idea of predicting the visual feature vector of the next frame from the feature vector of the current frame using Hebbian associative learning. The part of the feature vector corresponding to filter responses close to the vehicle is used. Assuming that the road close to the
vehicle should stay similar as long as the vehicle performs well, the similarity of the predicted feature vector and the feature vector obtained from the input image is used as performance-feedback.

A visual representation of the feature vector prediction linkage matrix after training is shown in figure 8.9. Activation of each feature is normalized for the internal structure to appear clearly. The matrix has a block Toeplitz like structure from the Kronecker products of the inputs and the predictions. This corresponds to smoothing along each of the three dimensions of the input features (orientation, scale and horizontal position). This can be more efficiently implemented as a smoothing operation or approximately as an exponential moving average of old feature vectors, i.e. a first order IIR filter. The linkage matrix corresponding to using a moving average is also shown in figure 8.9. The latter is implemented in the system.

The scalar product of the normalized predicted feature vector and the normalized newly observed feature vector is calculated for each frame, providing a similarity measure between 0 (since entries are non-negative) and 1. The offset from the mean similarity determines the feedback: if the current input is more similar to the predicted input than average, positive feedback is provided, else negative feedback is provided. The feedback strength is modulated depending on the absolute offset from the average similarity.

8.6.2 Experiments

In the following experiments, the main goal is to drive as close to the middle marker on the road as possible, in contrast to the previous experiments where the main goal was to stay on the road. This provides finer grained performance measures. The setup is shown in figure 8.10 and in the supplementary video.\footnote{http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125916}

A set of experiments are performed to evaluate different aspects of the system. To verify the mode selection by higher-level priors, experiments are performed on a track with intersections. The results of these experiments are more visual in nature and are presented primarily in the supplementary video. The prior distributions are set to $y_{\text{left}} \mid \text{right}(s) = \frac{1}{2}(1 \pm s)$ for left and right prior respectively (the signal range for steering, s, is $-1$ to 1). The conjugate prior is set to the uniform distribution.

The self-feedback behavior is evaluated in experiments where the vehicle is first trained to go around the test track with feedback disabled. After driving autonomously around the track a few laps, the self-feedback is enabled. The self-learning is combined with a bias towards higher speeds. The external feedback is evaluated in a similar way. After autonomous operation is achieved, positive and negative feedback is provided to the system. Positive feedback is applied together with a relative prior to go faster.

A series of experiments is performed outdoors under different conditions and on different road types. These experiments are presented in the supplementary video.
Figure 8.9: (a) Linkage matrix for predicting visual features close to the vehicle, the structure from the Kronecker product of orientation (8 coefficients), scale (4 coefficients) and horizontal position (8 coefficients) is visible. Viewed as a linear operator, it has a smoothing effect on the represented 3D features. (b) Corresponding figure using the exponential moving average of previous feature vectors. IEEE 2016.
Table 8.1: Overview of approaches for visual autonomous driving. Results from [97, 27, 75, 77, 80] are included.

<table>
<thead>
<tr>
<th>Method</th>
<th>Successful Driving</th>
<th>Learning Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>original CNN [97]</td>
<td>No, offline predictions</td>
<td>Days (batch)</td>
</tr>
<tr>
<td>Random Forest [27]</td>
<td>Yes, static track</td>
<td>Hours (batch)</td>
</tr>
<tr>
<td>LWPR [77]</td>
<td>With input projection</td>
<td>Video rate initially</td>
</tr>
<tr>
<td>Associative Hebbian [75]</td>
<td>No</td>
<td>Video rate (online)</td>
</tr>
<tr>
<td>qHebb [75]</td>
<td>Yes, dynamic track</td>
<td>Video rate (online)</td>
</tr>
<tr>
<td>CNN [80]</td>
<td>No (too slow)</td>
<td>Hours (batch)</td>
</tr>
<tr>
<td>SOLAR [80]</td>
<td>Yes, dynamic track</td>
<td>Video rate (online)</td>
</tr>
</tbody>
</table>

8.6.3 Results

The system is primarily compared to the previous multiple-hypothesis learning system from section 8.5. Following the naming of the original publication, the previous system is denoted qHebb in this section, and the new system is known as symbiotic online learning of associations and regression (SOLAR). Nonetheless, both are based on qHebb associative learning. Table 8.1 presents an overview of current and earlier approaches.

The results of the evaluation experiments related to the multimodal capabilities of the system are primarily presented in the supplementary video. These experiments show that a weak prior is sufficient to select driving direction at an intersection while the same prior applied at a different place will not make the car leave the road, e.g. although providing left prior in a right turn, the car follows the road to the right.

A typical run of learning from demonstration followed by autonomous driving and then increasing speed by throttle bias and positive feedback is shown in figure 8.11. The last step is only available for SOLAR. The achieved steering accuracy is comparable for both algorithms. This was to be expected, since the learning-from-demonstration part of SOLAR is very similar to qHebb. The difference lies in the output representation. While qHebb is basically remaining static when in autonomous mode, SOLAR is continuously changing the mapping towards higher speed due to the imposed prior. The effect is clearly visible in figure 8.11 where the average speed goes up from about 0.5 m/s to 0.7 m/s during the time from 120 s to 300 s.

At the same time, steering performance decreases, which is expected since the previously learned steering control patterns no longer fit the changed vehicle dynamics, see the time interval from 140 s to 280 s. By punishing poor steering performance and awarding good steering actions, the overall steering accuracy improves eventually and new patterns that are valid for higher speeds are generated. At 280 s - 300 s, which corresponds to more than one lap around the track, the...
Figure 8.10: Snapshot from the supplementary video. A robotic car is trained from demonstration to drive autonomously on a reconfigurable track. No information regarding visual appearance, track geometry, or driving behavior is given initially. IEEE 2016.

Table 8.2: Median speed and deviation from the desired trajectory, evaluated over selected intervals corresponding to different phases of the experiments. IEEE 2016.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Time</th>
<th>Speed</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLAR [80]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial training (man.)</td>
<td>0-15s</td>
<td>0.5278</td>
<td>0.0812</td>
</tr>
<tr>
<td>Auto., after manual (a)</td>
<td>40-110s</td>
<td>0.5249</td>
<td>0.0605</td>
</tr>
<tr>
<td>Auto., after reinf.</td>
<td>280-300s</td>
<td>0.7415</td>
<td>0.0862</td>
</tr>
<tr>
<td>Auto., after manual (c)</td>
<td>40-110s</td>
<td>0.3150</td>
<td>0.2347</td>
</tr>
<tr>
<td>Auto., after self-reinf.</td>
<td>500-600s</td>
<td>0.4331</td>
<td>0.1517</td>
</tr>
<tr>
<td>Auto., after self-reinf.</td>
<td>1100-1200s</td>
<td>0.5440</td>
<td>0.1427</td>
</tr>
<tr>
<td>qHebb [75]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial training (man.)</td>
<td>0-18s</td>
<td>0.3555</td>
<td>0.1494</td>
</tr>
<tr>
<td>Auto., after manual (b)</td>
<td>50-200s</td>
<td>0.3630</td>
<td>0.1492</td>
</tr>
<tr>
<td>Auto., after reinf.</td>
<td>Feedback learning not possible.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 8.11: Measured deviation from the ideal driving line and speed of the RC-car during (a) SOLAR online learning, (b) qHebb online learning [75] and (c) SOLAR online learning with self-feedback. The line-color at the bottom indicates the driving mode: magenta – learning from demonstration, blue – autonomous driving. The line-color above indicates the reinforcement feedback: green – positive, red – negative, magenta – self-feedback, white – none. See the supplementary video. IEEE 2016.
steering accuracy is at the same level as for the low-speed case. Median values over these intervals are presented in table 8.2. The increased speed introduces slightly larger deviations from the desired line while the median speed is significantly higher.

This behavior of SOLAR continues, since the prior pushes the speed even higher and the system changes between periods with increasing velocity level, more failures, and several instances of negative feedback, and periods with constant velocity, no failures, and mostly positive feedback. In this way, the system acquires good steering capabilities at successively higher speeds. Finally, the recurring negative feedback due to steering error, forces the learning to no further increase the speed and the mapping has converged to a maximum possible speed given the track layout and the latencies of the system and the teacher. As the effect of providing positive or negative feedback to the system is immediate, feedback has to be provided timely.

The self-feedback setup is evaluated in a separate experiment. Figure 8.11c shows the corresponding speed and position offset plots. The strength of the feedback is reduced when using self-feedback compared to external feedback from the teacher. Too strong feedback resulted in unstable behavior. This indicates that the human teacher provides better feedback than the self-contained visual feature prediction system. Due to the weaker feedback, the time axis is longer to capture the behavior of the system.

After initial training the system is allowed to run four laps autonomously before self-feedback is activated. This increases the speed of the vehicle. After 300 seconds, the system goes beyond its own capabilities and manual control is required. After this, speed continues to increase, mostly without supervision. In the end, speed has increased more than 0.2 m/s, similar to manual feedback, however much less manual supervision was required compared to feedback learning in figure 8.11a. Steering deviation decreases slightly, table 8.2, mostly due to additional manual control.

8.6.4 Conclusion

The learning-from-demonstration capabilities of the system are similar to the previous learning system, section 8.5, while the new system also predicts throttle and not only steering. The novel learning modalities allows for increasing the performance of the system without the need of providing better demonstration. With performance-feedback, the system obtains performance beyond the demonstrated level. The increase in performance can also be achieved by self-feedback, although longer time is required. Using absolute priors, the system can be controlled at intersections. At the same time, the priors are weak enough not to make the vehicle leave the road even if a turn prior is applied at a straight section of road. As demonstrated in the supplementary video, the system is able to learn to follow regular roads outside the lab environment.
8.7 Learning to Follow a Human

Given the general structure of the system, it should be able to learn to follow other objects. This possibility was explored in a quick experiment using the system of section 8.6, without any modifications. In this single experiment, only the learning-from-demonstration ability was used.

The experiment shows that the system is able to learn to follow a person. A video sequence illustrating the on-board data of the system during the experiment is available among the supplementary material. A frame from the sequence is shown in figure 8.12. These results have not been previously published.

Figure 8.12: One frame from the person following demonstration video. The image is from the on-board camera. The heat-map illustrates the predicted control distribution, throttle is vertical and steering is horizontal. The strongest mode, marked with a dot, suggests turning right at about a quarter throttle.

*http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125916*
8.8 Summary and Conclusion

Through this chapter, the demonstrator system for autonomous road following has evolved from an offline learning system to an online learning system allowing several different learning modalities. This generates a teacher friendly human-machine interface where the teacher may select any type of learning at any time the teacher finds appropriate. This has allowed the system to reach performance levels beyond the level of demonstration by the teacher.

The experiments have illustrated how a simple and lightweight online learning system, comparable in complexity to a single layer artificial neural network, can achieve performance comparable to the heaviest offline state-of-the-art learning systems such as deep learning and especially convolutional neural networks. The central representation of what the system has learned, the linkage matrix, has 56 rows and 14336 columns in the latest system. The matrices in the previous systems are smaller. This corresponds to 802816 elements, significantly smaller than a single image from a modern digital camera. With this in mind, it would be possible to extend the system using several different linkage matrices and using the most appropriate matrix at any given time, e.g. linkage matrix selection by scene recognition.

Obviously there are many more tasks to take care of in an autonomously driving vehicle than autonomous road following. By reducing the effort required for road following, more computational capacity is available to the remaining system. The primary advantage of the presented approach is the ability to follow roads also without lane markers.

A common argument against using learning systems in traffic situations is their unknown behavior in rare situations related to traffic accidents since such events are more unlikely to be part of the training. However, there are systems already on the market taking care of traffic safety issues that may monitor the learning system and take over in case there is a risk of an accident happening.
Chapter 9

Bird and Head Orientation Regression

This chapter presents two applications for detecting orientation and position of birds and heads in images. The head orientation estimation uses Hebbian associative learning to learn a mapping from image information directly to head orientation. The second application uses more traditional image processing followed by a channel decoding operation to obtain a more robust position estimate.

The primary motivation of the head orientation estimation is to obtain comparable results for the qHebb associative learning. The head pose material was published in the BMVC publication on learning of road following [75]. Bird tracking on the other hand is motivated by biological research in an attempt to automate evaluation of experiments in so called Emlen funnels and to obtain new measurements which has not been possible using previous evaluation methods [6].

9.1 Head Pose Regression

The head pose regression experiment is a direct reproduction of an experiment from ECCV 2012 [45] for comparing Hebbian associative learning to other learning methods. The methods should learn a mapping from images of faces to head yaw angle for a subset of the Multi-PIE dataset [41]. The subset contains images of 11 different head poses with yaw angles from $-90^\circ$ to $+90^\circ$ in increments of $15^\circ$ under varying lighting conditions, this is the same subset used for the original
experiment [45]. The images are cropped and information from a subset of the images are randomly removed (square blocks of size 0.1 times image width at 5 random locations). For different scenarios, either none, 20%, 40% or 80% of the images are corrupted. The local orientation in the images is calculated and the orientation images are down-sampled to 25 by 30 pixels. The set of images is divided into two equal sets for training and evaluation with 1565 images in each set. Ten of the training images are shown in figure 9.1, the evaluation images are similar.

For qHebb associative learning with infinite $q$ (i.e. maxHebb), the orientation images are encoded using $15 \times 10 \times 10$ (1500 elements) channels in a Channel Coded Feature Map (CCFM) configuration [49] where $15 \times 10$ channels are used for spatial encoding and 10 channels are used for orientation encoding. The local orientation vector magnitude is used for weighting the orientation channels. The output space, head yaw, uses 16 channels. Coherence-weighting, with $\kappa = 2$, is used for predicting head yaw on the evaluation set. It is noted that direct coherence-weighting, with $\kappa = 0$ down-weights too much of the information in the linkage matrix which degrades performance.

For robust regression (RR) [45] and random forest regression (RFR) [12], the $25 \times 30 \times 2$ (1500 elements) orientation images are vectorized. Training on each full training set with different ratios of corrupted versus non-corrupted images required 12 hours for robust regression while qHebb required 15 seconds, both are MATLAB implementations.

The results are presented in figure 9.2, showing mean prediction error over the evaluation set as a function of number of training data used from the training set. Two points from the graph is also shown in table 9.1, results after training on partial and full training set. For comparison, since RR and RFR only run in batch mode, the training and evaluation are repeated for subsets of the training data. Support vector regression (SVR) [25] and ROGER [40] could not be used. SVR generated constant predictions independent of training set size and parameter changes. ROGER did not produce any useful results within a computation time of three days, and is marked as did not finish (DNF) in table 9.1.
9.1. HEAD POSE REGRESSION

Figure 9.2: Robust Regression, Random Forest Regression and associative learning using qHebb with infinite $q$ and coherence-weighting. 0%, 20%, 40% and 80% corrupted images respectively.

![Graph showing mean error on evaluation set versus training points](image)

Table 9.1: Mean angular error (degrees) for training on partial and full training set. Best results in boldface. Associative learning is evaluated using infinite $q$, both with and without coherence weighting.

<table>
<thead>
<tr>
<th>Corruption</th>
<th>391 training samples</th>
<th>1565 training samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0%</td>
<td>20%</td>
</tr>
<tr>
<td>RR</td>
<td>15.08</td>
<td>15.99</td>
</tr>
<tr>
<td>qHebb</td>
<td>12.24</td>
<td>12.22</td>
</tr>
<tr>
<td>qHebb, coh</td>
<td><strong>9.61</strong></td>
<td><strong>9.75</strong></td>
</tr>
<tr>
<td>RFR</td>
<td>10.10</td>
<td>10.75</td>
</tr>
<tr>
<td>SVR</td>
<td>DNF</td>
<td>DNF</td>
</tr>
<tr>
<td>ROGER</td>
<td>DNF</td>
<td>DNF</td>
</tr>
</tbody>
</table>

The results for robust regression using the full training set are similar to those reported by the original authors [45]. After 130 training samples, qHebb associative learning on 80% corrupted images outperforms robust regression on corruption free images. From table 9.1 it is clear that qHebb without coherence-weighting is less affected by image corruption than robust regression. Using coherence-weighting, qHebb results are not degraded by image corruption at all. Performance even seem to increase slightly with higher ratios of corrupted images. The higher ratio of corrupted images seems to reduce overfitting, visible as a slight performance decrease for large numbers of training data. The corruption process of removing image information is similar to the dropout idea for reducing overfitting in deep learning methods, where connections in the networks are randomly removed.

At 391 training samples, qHebb with coherence-weighting outperforms all competing methods while qHebb without coherence-weighting is outperformed by ran-
dom forest regression. For more than 1000 training samples, RFR produces results slightly superior to qHebb. However, RFR is trained in batch mode and requires storing all samples which is not feasible in an online learning scenario.

The setup is rather brutal in the sense of learning a mapping directly from orientation pixel values directly to head yaw angle. This is reflected in the relatively large errors. Most specialized head pose estimation systems should perform significantly better by utilizing prior information, such as that the images depict heads and the typical appearance of a head. Nonetheless, the motivation for the experiment is regression method evaluation for high-dimensional inputs, not head pose estimation in itself.

9.2 Bird Position and Pose Estimation

In a research field first appearing quite distant from computer vision, bird migration is a spectacular phenomenon. This phenomenon has been studied longer than the computer vision field has even existed. One particular discovery was that migratory birds in cages during migration season tend to try to fly in the direction they are supposed to go [102]. Emlen and Emlen proposed a special cage for studying this. Such a cage, later known as an Emlen funnel, is illustrated in figure 9.3. When the bird tries to fly, it will make marks on the funnel walls which can be used to generate a directional histogram.

![Figure 9.3: Emlen funnel. Image: L. Shyamal, Wikimedia.](image)

The marks on the funnel walls were processed manually, and still today much of this work is manual although some simplifications in the manual processing has been introduced such as only counting scratches crossing one specific line around the funnel [52]. Automatic counters using micro switches has been used, although electrical equipment close to the birds may possibly disturb the magnetic field and thus the results. To reduce any possible interference from sensors, a proposed solution is to place a camera to observe the birds from a distance [6].

Automating this observation and histogram generation requires cooperation across several fields, combining questions such as which bird behaviors are inter-
9.2. BIRD POSITION AND POSE ESTIMATION

Estimating the activity of birds and what is possible to achieve by automatically processing the video stream. For comparability with results in earlier publications, the system should be able to produce summaries similar to the manual Emlen funnel method. Furthermore, the constant monitoring allows new measurements to be collected, such as how the activity of the bird varies over time.

The full system contains a processing chain from position and pose detection via event detection to data summarization. This section will cover the image processing part of the system. From a pure computer vision perspective, the system uses mostly well-known methods and is of limited interest in that respect. The most significant contribution is the use of channel decoding to avoid influence from spurious noise in the segmented image.

9.2.1 Image Processing

A few frames from a recorded sequence is shown in figure 9.4. The camera view covers four cages. Concerning bird segmentation, it is noteworthy that the background may change quicker than the foreground (birds). The transition from daylight to night with near-infrared illumination takes less than eight minutes while the bird in the top left cage remains stationary for over 25 minutes.

An individual moving quantile background model set at 90% intensity is used for each pixel in the image sequence as lighting conditions changes over the sequences. Update rate is set slow enough for the birds not to become a part of the background even during long periods of inactivity. The moving quantile background model keeps the background model value close to the set quantile when considering the distribution of pixel values over time. The quantile is set high as the background tends to have a higher intensity than the bird.

For each frame, the pixel-wise difference image is calculated. The difference image is summed over channel basis functions in a log-polar grid, figures 7.8 and 7.9 on page 109, similar to a soft histogram. From this representation, the peak of the strongest mode is obtained by channel decoding. The locality of the decoding avoids influence from differences not related to the bird. The strongest peak is the estimated bird position, which is obtained with a higher precision than the bin spacing by use of the soft basis functions.

The body orientation is found by first calculating the 95% quantile of the difference image followed by thresholding the difference image and removing all but the largest connected component. An ellipse is fitted to the bird pixels; its main axis orientation provides an estimate of body orientation. Additionally, the variance of the distribution of bird pixel values in the difference image provides an indication of wing fluttering. When the bird is active, motion blur of primarily the wings generates a wide distribution of deviation pixel values. When the bird is stationary, the variance is smaller.

Using a combination of bird position changes and the wing fluttering signal, takeoff events are detected and the direction of takeoff as well as the bird orientation before takeoff is recorded.
Figure 9.4: Four images of a sequence captured of four bird cages. The images, in order top left, top right, bottom left, bottom right, were captured at wall clock times 15:59, 16:15, 16:23 and 16:48. The fitted ellipses are drawn in blue. The red circles are the centers of the ellipses and the blue circles mark the radially most extreme point of the bird detection. Note that in eight minutes, the image style changes from daylight to night illumination by near-infrared light. For comparison, the top left bird does not move between the third and fourth image, a time period of 25 minutes.
9.3 Conclusion

The experiments on head pose regression provides a possibility to compare the qHebb associative learning with other popular learning systems, to which qHebb compares favorably. The online learning from demonstration used in the road following experiments trains the system on a new training set every experiment, reducing comparability.

The novelty of the bird tracking system lies more on the biological methodical side. The application has nonetheless demonstrated that channel decoding can be used instead of a long processing chain of thresholding, morphological noise reduction operations, labeling and removal of all but the largest connected component followed by centroid extraction. The current system is a scripted prototype running at framerates below video rate. However, the system still saves the biology researchers the several hours of manual labeling work currently associated with each experiment.

The bird extraction should be a suitable application for the time-logarithmic channels. A background model based on such a representation would contain information regarding both quick and slow changes, delaying the selection of the corresponding update rate to the time of deciding between foreground and background for each pixel and frame.
Chapter 10

Visual Object Tracking

This chapter presents a visual object tracking framework based on the channel representation and the qHebb ideas. The framework is made for general short term visual object tracking like the annual VOT challenges.\footnote{http://votchallenge.net/} A bounding box containing the object to be tracked is provided with the first frame and the task is to follow the object in the following frames, using only the information available in the images.

Distribution-based methods have successfully competed in the challenge, both intensity-based \cite{100, 29} and also using color \cite{20}. The distribution view of the channel representation allows for other types of comparisons and update schemes compared to the direct pixel value representation. Such ideas were presented in section 6.6. The present tracking framework uses these ideas, exploiting the potential of the channel representation. This work has previously been published with the VOT challenge \cite{79}. A video illustrating the approach is available.\footnote{http://users.isy.liu.se/cvl/ofjall/vot2014.mp4}

10.1 General Tracking Framework

The tracker framework is similar to previous distribution-based trackers such as distribution field tracking (DFT) \cite{100} and the enhanced distribution field tracker (EDFT) \cite{29}. All pixels of the given target patch of the first image are channel encoded using $K = 15$ channels, resulting in an $I$ by $J$ by $K$ array denoted $C$, where $I$ and $J$ are the height and width of the supplied bounding box.

In the following frame, the target representation $C$ is compared to channel encoded patches (denoted $D_{mn}$) from the new frame, where $m$ and $n$ represent a shift of the selected patch over the image. Gradient descent is used to find a minimum of a given comparison function, $d(C, D_{mn})$, with respect to the shift $(m, n)$. Finally, the target representation is updated, $C \leftarrow g(C, D_{mn})$ and tracking continues in the next frame.

Prior to comparison, i.e. calculation of $d(C, D_{mn})$, the channel planes of $C$ and
are smoothed. This was shown to increase the size of the basin of attraction for the correct solution \[100\]. Also, as in DFT and EDFT, a simple motion model (constant velocity model in the image plane) is used for initializing the gradient descent. The distance function \(d(C, D)\) and the update function \(g(C, D)\) are presented in section 6.6, equations (6.23), (6.24) and (6.22), where the relation to qHebb associative learning is pointed out.

Since objects to be tracked are rarely rectangular, background pixels will be present in the bounding box. These pixels will generally vary more than the pixels on the object and such background pixels may disturb the tracker. This leads to a hypothesis that a weighted norm where the influence of inconsistent pixels is reduced, will improve the tracking results. Furthermore, there may be areas of the tracked object which frequently change appearance, a weighted norm should also put more emphasis on parts of the tracked object showing more static appearance. Two different norm weighting schemes are evaluated. The first uses the reciprocals of the standard deviations of the represented distributions, the second approach uses the coherence.

10.2 Experiments

The enhanced trackers using the new update function and weighted comparison functions are compared on the VOT2014 challenge benchmark, using the evaluation framework provided by the challenge \[57\]. Comparisons are made with DFT, EDFT and NCC, an example normalized cross correlation tracker distributed with the evaluation framework.

Trackers using the proposed q-update scheme are prefixed with a \(q\), and followed by an indication of the value of \(q\). Infinite \(q\) is denoted by \(\max\), a special case as the q-update tend to the max-operation for increasing \(q\). Trackers using the proposed coherence-weighted comparison are prefixed with a \(w\) and trackers using the proposed standard-deviation-weighted comparison are prefixed with \(w\sigma\). Unmarked trackers use the \(L_1\) norm comparison. For coherence-weighting, the parameter \(\kappa\) was set to 2. For all trackers, learning rate \(\gamma\) is set to 0.05 and 15 channels are used.

Three performance measures are available in the evaluation framework \[57\]. Accuracy is the ratio of the joint area of tracker output and ground truth and the union of the two, averaged over each sequence (larger is better). Robustness is the reset count, the evaluated tracker is reset as soon as there is no overlap between tracker output and ground truth (smaller is better). Speed is the average framerate of the tracker (larger is better).

The evaluation framework uses two different experiments. In the first, denoted baseline, each tracker is initialized using the ground truth bounding box of the first frame. In the second experiment, region noise, each tracker is initialized with the ground truth bounding box with a random offset. In the second experiment each sequence is evaluated 15 times with different offsets and the mean is reported by the VOT evaluation framework. The results for the baseline experiment are presented in table 10.1, and the results for the region noise experiment are presented in
10.2. EXPERIMENTS

Table 10.1: Summarized results for the baseline experiment, comparison to competing methods (best scores in boldface). With permission from Springer. Column labels are in order Accuracy, Robustness and Speed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>Rbst</td>
</tr>
<tr>
<td>NCC</td>
<td>0.467</td>
<td>2.960</td>
</tr>
<tr>
<td>DFT</td>
<td>0.531</td>
<td>2.200</td>
</tr>
<tr>
<td>qEDFT (q=2)</td>
<td>0.525</td>
<td>2.000</td>
</tr>
<tr>
<td>qEDFT (q=3)</td>
<td>0.536</td>
<td>1.720</td>
</tr>
<tr>
<td>qEDFT (q=4)</td>
<td>0.547</td>
<td>1.720</td>
</tr>
<tr>
<td>qEDFT (q=5)</td>
<td>0.552</td>
<td>1.720</td>
</tr>
<tr>
<td>qEDFT (q=6)</td>
<td>0.540</td>
<td>1.920</td>
</tr>
<tr>
<td>wEDFT</td>
<td>0.523</td>
<td>2.040</td>
</tr>
<tr>
<td>qwEDFT (q=2)</td>
<td>0.544</td>
<td>1.560</td>
</tr>
<tr>
<td>qwEDFT (q=3)</td>
<td>0.547</td>
<td>1.600</td>
</tr>
<tr>
<td>qwEDFT (q=4)</td>
<td>0.550</td>
<td>1.560</td>
</tr>
<tr>
<td>qwEDFT (q=5)</td>
<td>0.554</td>
<td>1.640</td>
</tr>
<tr>
<td>qwEDFT (q=6)</td>
<td>0.558</td>
<td>1.920</td>
</tr>
<tr>
<td>qwEDFT (q=7)</td>
<td><strong>0.558</strong></td>
<td>1.640</td>
</tr>
<tr>
<td>maxwEDFT</td>
<td>0.545</td>
<td>1.960</td>
</tr>
<tr>
<td>qwσEDFT (q=2)</td>
<td>0.522</td>
<td>1.400</td>
</tr>
<tr>
<td>qwσEDFT (q=3)</td>
<td>0.522</td>
<td>1.440</td>
</tr>
<tr>
<td>qwσEDFT (q=4)</td>
<td>0.540</td>
<td><strong>1.360</strong></td>
</tr>
<tr>
<td>qwσEDFT (q=5)</td>
<td>0.545</td>
<td>1.520</td>
</tr>
<tr>
<td>qwσEDFT (q=6)</td>
<td>0.541</td>
<td>1.480</td>
</tr>
<tr>
<td>qwσEDFT (q=7)</td>
<td>0.545</td>
<td>1.600</td>
</tr>
<tr>
<td>maxwσEDFT</td>
<td>0.547</td>
<td>1.960</td>
</tr>
</tbody>
</table>

data 10.2. The average and median score over all sequences are presented for each tracker.

For the baseline experiment (table 10.1), all channel-based trackers outperform the tracker based on normalized cross correlation (NCC) in accuracy and robustness. For evaluation of the proposed extensions, the trackers are compared to the baseline channel-based tracker (EDFT). Introducing the non-linear update (qEDFT) increases accuracy and slightly increases robustness (decreasing failure rate) for increasing $q$ up to $q = 5$. For $q = 6$, performance decreases slightly. Only using the proposed weighted comparison (wEDFT), robustness decreases slightly while accuracy stays similar to EDFT.

The best performance is achieved by combining the non-linear update with the weighted comparison. Using non-linear update and coherence-weighted comparison (qwEDFT with $q = 4$), mean accuracy increases more than 5% and mean robustness is 15% better than EDFT. For larger $q$, accuracy increases further while the robustness degrades. The corresponding standard-deviation-weighted methods


Table 10.2: Summarized results for the region noise experiment, comparison to competing methods (best scores in boldface). With permission from Springer. Column labels are in order Accuracy, Robustness and Speed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th></th>
<th></th>
<th>Median</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>Rbst</td>
<td>Spd</td>
<td>Acc</td>
<td>Rbst</td>
<td>Spd</td>
</tr>
<tr>
<td>NCC</td>
<td>0.456</td>
<td>2.973</td>
<td>14.0</td>
<td>0.414</td>
<td>1.8</td>
<td>12.0</td>
</tr>
<tr>
<td>DFT</td>
<td>0.493</td>
<td>2.389</td>
<td>6.0</td>
<td>0.512</td>
<td>2.4</td>
<td>5.9</td>
</tr>
<tr>
<td>qEDFT (q=2)</td>
<td>0.492</td>
<td>2.059</td>
<td>10.3</td>
<td>0.488</td>
<td>1.9</td>
<td>10.8</td>
</tr>
<tr>
<td>qEDFT (q=3)</td>
<td>0.497</td>
<td>2.000</td>
<td>6.9</td>
<td>0.518</td>
<td>1.8</td>
<td>6.8</td>
</tr>
<tr>
<td>qEDFT (q=4)</td>
<td>0.498</td>
<td>2.032</td>
<td>6.7</td>
<td>0.492</td>
<td>1.7</td>
<td>6.6</td>
</tr>
<tr>
<td>qEDFT (q=5)</td>
<td>0.502</td>
<td>2.008</td>
<td>6.7</td>
<td>0.512</td>
<td>1.3</td>
<td>6.7</td>
</tr>
<tr>
<td>qEDFT (q=6)</td>
<td>0.499</td>
<td>2.093</td>
<td>6.4</td>
<td>0.520</td>
<td>1.6</td>
<td>6.5</td>
</tr>
<tr>
<td>wEDFT</td>
<td>0.489</td>
<td>2.088</td>
<td>6.2</td>
<td>0.492</td>
<td>1.9</td>
<td>6.3</td>
</tr>
<tr>
<td>qwEDFT (q=2)</td>
<td>0.501</td>
<td>1.835</td>
<td>6.7</td>
<td>0.500</td>
<td>1.9</td>
<td>6.8</td>
</tr>
<tr>
<td>qwEDFT (q=3)</td>
<td>0.508</td>
<td>1.819</td>
<td>5.4</td>
<td>0.520</td>
<td>1.6</td>
<td>5.4</td>
</tr>
<tr>
<td>qwEDFT (q=4)</td>
<td>0.509</td>
<td>1.747</td>
<td>5.1</td>
<td>0.502</td>
<td>1.5</td>
<td>4.9</td>
</tr>
<tr>
<td>qwEDFT (q=5)</td>
<td>0.515</td>
<td>1.819</td>
<td>5.2</td>
<td>0.499</td>
<td>1.5</td>
<td>5.1</td>
</tr>
<tr>
<td>qwEDFT (q=6)</td>
<td>0.516</td>
<td>1.837</td>
<td>5.2</td>
<td>0.530</td>
<td>1.5</td>
<td>5.2</td>
</tr>
<tr>
<td>qwEDFT (q=7)</td>
<td>0.514</td>
<td>1.923</td>
<td>5.1</td>
<td>0.515</td>
<td>1.5</td>
<td>5.1</td>
</tr>
<tr>
<td>maxwEDFT</td>
<td>0.514</td>
<td>2.163</td>
<td>6.3</td>
<td>0.500</td>
<td>2.0</td>
<td>6.2</td>
</tr>
<tr>
<td>qwσEDFT (q=2)</td>
<td>0.500</td>
<td>2.029</td>
<td>8.6</td>
<td>0.520</td>
<td>1.5</td>
<td>8.8</td>
</tr>
<tr>
<td>qwσEDFT (q=3)</td>
<td>0.502</td>
<td>1.832</td>
<td>6.6</td>
<td>0.510</td>
<td>1.5</td>
<td>6.6</td>
</tr>
<tr>
<td>qwσEDFT (q=4)</td>
<td>0.521</td>
<td>1.893</td>
<td>7.1</td>
<td>0.534</td>
<td>1.6</td>
<td>6.7</td>
</tr>
<tr>
<td>qwσEDFT (q=5)</td>
<td>0.506</td>
<td>1.803</td>
<td>6.5</td>
<td>0.515</td>
<td>1.7</td>
<td>6.4</td>
</tr>
<tr>
<td>qwσEDFT (q=6)</td>
<td>0.510</td>
<td>1.787</td>
<td>6.8</td>
<td>0.517</td>
<td>1.7</td>
<td>6.7</td>
</tr>
<tr>
<td>qwσEDFT (q=7)</td>
<td>0.511</td>
<td>1.795</td>
<td>6.6</td>
<td>0.517</td>
<td>1.8</td>
<td>6.4</td>
</tr>
<tr>
<td>maxwσEDFT</td>
<td>0.516</td>
<td>2.109</td>
<td>7.9</td>
<td>0.529</td>
<td>2.0</td>
<td>7.8</td>
</tr>
</tbody>
</table>

Perform slightly inferior to the best methods (the coherence-weighted) in terms of mean accuracy. However, the best robustness is achieved by a standard-deviation-weighted method (qwσEDFT with \( q = 4 \)). In general, accuracy seem to improve with larger \( q \) while the best robustness is achieved for \( q \) close to 4. For median accuracy, \( q = 4 \) gives the best performance for both coherence-weighted trackers and standard-deviation-weighted trackers, with better results for coherence-weighting.

For the region noise experiments (table 10.2), accuracy generally increases with increasing \( q \) while the best robustness is achieved for \( q = 4 \) for the standard-deviation-weighted tracker and for \( q = 6 \) for the coherence-weighted tracker. Contrary to the baseline experiments, in the region noise experiments the coherence-weighted methods perform best with respect to robustness while the standard-deviation-weighted methods perform best with respect to accuracy.

In table 10.3, the results for each sequence for three trackers are presented. A comprehensive description of the sequences themselves is available at the VOT challenge site.\(^3\) Both proposed trackers outperform the EDFT tracker with respect to accuracy on 15 out of 25 sequences. On four sequences the EDFT tracker

\(^3\)http://votchallenge.net/vot2014/dataset.html
Table 10.3: Detailed baseline experiment results for three trackers. Best scores in bold. With permission from Springer. Column labels are in order Accuracy and Robustness.

<table>
<thead>
<tr>
<th></th>
<th>EDFT acc</th>
<th>EDFT rbst</th>
<th>qwσEDFT (q=4) acc</th>
<th>qwσEDFT (q=4) rbst</th>
<th>qwEDFT (q=4) acc</th>
<th>qwEDFT (q=4) rbst</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball</td>
<td>0.51</td>
<td>0.57</td>
<td>0.59</td>
<td>0.59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>basketball</td>
<td>0.56</td>
<td>3</td>
<td>0.57</td>
<td>1</td>
<td>0.59</td>
<td>3</td>
</tr>
<tr>
<td>bicycle</td>
<td>0.44</td>
<td>0</td>
<td>0.43</td>
<td>0</td>
<td>0.43</td>
<td>0</td>
</tr>
<tr>
<td>bolt</td>
<td>0.51</td>
<td>0.56</td>
<td>3</td>
<td>0.56</td>
<td>0.56</td>
<td>3</td>
</tr>
<tr>
<td>car</td>
<td>0.53</td>
<td>0.53</td>
<td>1</td>
<td>0.53</td>
<td>0.53</td>
<td>1</td>
</tr>
<tr>
<td>david</td>
<td>0.68</td>
<td>0.71</td>
<td>0</td>
<td>0.72</td>
<td>0.72</td>
<td>0</td>
</tr>
<tr>
<td>diving</td>
<td>0.16</td>
<td>3</td>
<td>0.16</td>
<td>3</td>
<td>0.16</td>
<td>3</td>
</tr>
<tr>
<td>drunk</td>
<td>0.51</td>
<td>0.49</td>
<td>0</td>
<td>0.50</td>
<td>0.50</td>
<td>0</td>
</tr>
<tr>
<td>fernando</td>
<td>0.40</td>
<td>2</td>
<td>0.40</td>
<td>2</td>
<td>0.43</td>
<td>2</td>
</tr>
<tr>
<td>fish1</td>
<td>0.38</td>
<td>4</td>
<td>0.40</td>
<td>4</td>
<td>0.42</td>
<td>4</td>
</tr>
<tr>
<td>fish2</td>
<td>0.28</td>
<td>6</td>
<td>0.30</td>
<td>5</td>
<td>0.32</td>
<td>6</td>
</tr>
<tr>
<td>gymnastics</td>
<td>0.55</td>
<td>2</td>
<td>0.54</td>
<td>2</td>
<td>0.53</td>
<td>2</td>
</tr>
<tr>
<td>hand1</td>
<td>0.59</td>
<td>2</td>
<td>0.56</td>
<td>0</td>
<td>0.60</td>
<td>0</td>
</tr>
<tr>
<td>hand2</td>
<td>0.42</td>
<td>7</td>
<td>0.44</td>
<td>3</td>
<td>0.44</td>
<td>6</td>
</tr>
<tr>
<td>jogging</td>
<td>0.79</td>
<td>2</td>
<td>0.80</td>
<td>2</td>
<td>0.80</td>
<td>2</td>
</tr>
<tr>
<td>motocross</td>
<td>0.18</td>
<td>3</td>
<td>0.23</td>
<td>4</td>
<td>0.20</td>
<td>3</td>
</tr>
<tr>
<td>polarbear</td>
<td>0.53</td>
<td>0</td>
<td>0.58</td>
<td>0</td>
<td>0.59</td>
<td>0</td>
</tr>
<tr>
<td>skating</td>
<td>0.61</td>
<td>1</td>
<td>0.61</td>
<td>1</td>
<td>0.62</td>
<td>1</td>
</tr>
<tr>
<td>sphere</td>
<td>0.62</td>
<td>1</td>
<td>0.69</td>
<td>0</td>
<td>0.71</td>
<td>0</td>
</tr>
<tr>
<td>sunshade</td>
<td>0.65</td>
<td>3</td>
<td>0.70</td>
<td>1</td>
<td>0.71</td>
<td>1</td>
</tr>
<tr>
<td>surfing</td>
<td>0.85</td>
<td>0</td>
<td>0.89</td>
<td>0</td>
<td>0.90</td>
<td>0</td>
</tr>
<tr>
<td>torus</td>
<td>0.82</td>
<td>0</td>
<td>0.77</td>
<td>0</td>
<td>0.80</td>
<td>0</td>
</tr>
<tr>
<td>trellis</td>
<td>0.51</td>
<td>2</td>
<td>0.52</td>
<td>1</td>
<td>0.56</td>
<td>1</td>
</tr>
<tr>
<td>tunnel</td>
<td>0.31</td>
<td>0</td>
<td>0.31</td>
<td>0</td>
<td>0.31</td>
<td>0</td>
</tr>
<tr>
<td>woman</td>
<td>0.61</td>
<td>1</td>
<td>0.72</td>
<td>1</td>
<td>0.69</td>
<td>1</td>
</tr>
</tbody>
</table>

outperforms the proposed trackers and in three cases, performance is equal among the three trackers. With respect to robustness, all three trackers perform equal on 18 out of 25 sequences. The improvement compared to EDFT with respect to robustness is largest on the sequences where EDFT performs worst. On the hand2 sequence, EDFT loses track of the object seven times while the proposed qwσEDFT tracker loses track of the object three times.

No parameters have been changed from those used in the baseline trackers, with the exception of the newly introduced parameter $q$. Since $q$ and the learning rate $\gamma$ together determine the effective learning and forgetting rates of the final tracker, a further increase in performance should be possible by jointly optimizing these parameters. The current implementation of the extended trackers recomputes weights every search iteration. By avoiding this, an increase in framerate should be possible to achieve.
A selection of different comparison functions was evaluated such as $L_2$, variance weighed $L_2$ and Hellinger distance. However, these performed inferior to the weighted $L_1$ norms.

10.3 Conclusion

The ideas from qHebb associative learning regarding update and weighting applied to channel-based tracking increased the tracker performance on the VOT challenge benchmark. The best performance was obtained for a combination of both weighting and non-linear update and a $q$ value around four to five, i.e. neither linear update nor max update.
Chapter 11

Projected Histogram Rail Curvature Estimation

This chapter presents a system for estimating rail curvature in front of trains from a train mounted thermal infrared camera. The rail curvature estimation system is one part of a collision warning system for trains. This work uses non-overlapping channels, i.e. regular histograms. This allows pre-calculating a bin map image for each rail which maps each pixel of an image to the corresponding curvature bin, but comes with the cost of reducing the estimation accuracy to the bin width. This chapter is based on the geometrical contributions of the SCIA publication [5].

11.1 Introduction

Due to the long stopping distances of a train driving at normal speeds, exceeding one kilometer, collisions with objects located on the railway is common. In conditions with good visibility, the speed of the train can be significantly reduced.

![Figure 11.1: The camera installation for data collection. (a) The camera house with a 45-degree mirror. (b) The camera house mounted on the front of a train. (c) The display at the driver’s panel. With permission of Springer.](image)
before collision, limiting damage to the train. However, during limited visibility conditions, such as due to fog, precipitation or limited light, high speed collisions are common. Such collisions cause delays in the train traffic, severe damage to the train and threaten the safety of the train passengers.

The train safety system uses a thermal infrared camera, a FLIR SC655 acquiring images in the 8–12 \( \mu \text{m} \) band at 50 Hz framerate, making the system independent of external illumination. However, the spatial resolution is more limited, 640 by 480 pixels. The camera installation for the prototype system is shown in figure 11.1. The arrangement with the mirror is due to the risk of collisions with smaller objects in combination with the expensiveness of the camera and protective windows which are transparent in the thermal infrared band. The final system has a different camera arrangement.

The prototype system was used for data collection for initial system development. Optics with horizontal fields of view of 7 and 25 degrees were evaluated. The field of view presents a tradeoff between detection ability at curved sections and detection distance. A moose at a distance of one kilometer occupies less than a pixel when using a field of view of 25 degrees. The images shown in this chapter has been acquired using the 25-degree optics.

The first stage is to detect the rails. After rail detection, an anomaly filter is used to detect areas not looking like rails where the rails are supposed to be. The collision warning system is illustrated in 11.2. The geometry and rail detection parts will be treated here.

11.1.1 Railway and Road Detection

At a first glance, railroad detection problem may seem similar to that of road marker detection. However, at a closer look, the two problems are quite different. Roads are typically structure-less, but with defined borders such as lines, curbs, or railings. In urban areas and on highways, visual road detection is carried out by detection of lane markers. A common approach is to reproject the image onto the ground plane followed by line detection in the resampled image [1, 55, 83]. Using other sensors such as LIDAR, similar lane detection approaches are used [50].

For railways, the situation is different. The railway has a defined structure; two rails at a specified distance from each other and perpendicular sleepers. On
the other hand, there are no defined borders or lane markers. A different strategy is needed. Nonetheless, some ideas from lane detection are transferable to this new domain; for example, resampling the image in a ground plane grid is advantageous compared to line detection directly in the camera image [10]. Unfortunately, this resampling destroys information close to the vehicle where the road is densely sampled in the original image, while much effort is spent representing far areas where little information is available. However, under all normal operating conditions, the train is always centered and aligned with the rails. Using this allows obtaining the advantages of rail detection in the reprojected ground plane image without the need of explicitly reprojecting each frame, thus avoiding the drawbacks of resampling.

11.2 Rail Geometry

Assuming a flat ground, which is appropriate in a railway setting with limited gradients, there is a one to one mapping from pixels to points in a ground coordinate system fixed to the train, commonly referred to as the inverse perspective mapping (IPM), a homography determined from the known camera parameters. Further, the train has a fixed position and orientation relative to the railway. Assuming a locally constant curvature of the railway, the curvature is the only free parameter determining the position of the rails in the image. This is exploited to obtain fast and reliable rail detections.

Given the design of a railway engine, it is apparent that the rails will be parallel to the engine at a point midway between the fore and aft bogies, see 11.3a, later referred to as the parallel point. The orthogonal offset at this point, \( \lambda \) in 11.3a, the signed distance between the center of the railway and the center of the engine, is determined by the local curvature, \( Q = 1/R \), and the wheel base, \( c \). It is given by the width of a circle segment

\[
\lambda = \frac{1}{Q} - \sqrt{\frac{1}{Q^2} - \frac{c^2}{4}} \approx \frac{Qc^2}{8}, \tag{11.1}
\]

where the approximation for small curvatures is linear in \( Q \). Together with the length-wise camera mount offset \( t \) and camera parameters, this determines the position of the rails in the image for each possible curvature.

The deviation, \( d \) in 11.3b, of a rail from the parallel point can be derived from geometric relations and is given by

\[
d = h \cot \left( \frac{\pi - \sin^{-1}(hQ)}{2} \right) \tag{11.2}
\]

with the inverse

\[
Q = \frac{1}{R} = \frac{1}{h} \sin \left( \pi - 2 \cot^{-1} \frac{d}{R} \right). \tag{11.3}
\]

The geometry and parameters are illustrated in 11.3b.
11.2.1 Histogram Bin Mappings

Given the camera parameters, a pixel position in the image can be translated to the corresponding position in a world coordinate system fixed to the train, assuming a flat ground. Using the relations (11.1) and (11.3), the required curvature of each rail for the corresponding rail to pass through the given point can be found. This can be determined for all points below the horizon. By binning the one-curvature space, each pixel below the horizon will fall into one specific bin for each rail. I.e. each pixel will map to two curvature bins, one for each rail and if e.g. the left rail is found in this pixel, the curvature is known up to the bin width. One bin map for each rail is generated once, shown in figure 11.4. A suitable range of curvature is discretized into 100 bins. If soft overlapping bins had been used, a channel representation, the curvature estimate could have been extracted with sub-bin precision. However, this would require three bin map images per rail for channel representations with overlap three. Further, the weights for each bin and pixel would need to be stored.

In automotive applications, the lateral position and to some extent the orientation of the car on the road is not fixed, thus requiring at least a two-dimensional histogram. Further, each pixel should possibly be mapped to several bins and the number of bin map images would increase with the bin count.

Given that one rail pass through a given pixel, the orientation of the rail at the corresponding point is given by the geometry. This can be transferred to the projected rail orientation in the image plane. The orientation images, one for each rail, is shown in figure 11.5, where 0 is vertical and $\pm \pi/2$ is horizontal.
11.2. RAIL GEOMETRY

Figure 11.4: Images illustrating the mapping from pixel positions to the corresponding curvature histogram bin indexes, for left and right rail respectively. With permission of Springer.

Figure 11.5: Expected rail direction in image, left and right rail respectively. With permission of Springer.
11.2.2 Curvature Estimation

The curvature histogram mappings and the expected orientation are calculated in advance for rails close to the camera where the flatness and constant curvature assumptions hold. For each new frame, edge strength and orientation is estimated for all pixels below the horizon using Gaussian derivative filters. The horizon is known beforehand from the camera geometry. For each pixel \((x, y)\) and rail, the edge strength \(A_m(x, y)\) is modulated depending on the difference between the estimated orientation \(P_m(x, y)\) and the expected orientation \(P_e(x, y)\), figure 11.5, according to

\[
A_m(x, y) \exp \left( - \frac{(P_m(x, y) - P_e(x, y))^2}{\sigma_P^2} \right),
\]

where the parameter \(\sigma_P\) determines the orientation error tolerance. This expression does not regard the modular nature of orientations. However, by properly placing the branch cut and assuming rails not to turn back towards the train, the issue is avoided.

For each pixel and rail, the modulated value is added to the curvature bin determined by the bin map, figure 11.4. After summing all contributions from all pixels and both rails, the peak of the curvature histogram is extracted. The result is illustrated in figure 11.6, where an image is shown with the areas corresponding to the histogram peak overlaid. Two versions of the histogram are shown, one using both edge strength and orientation and one using only edge strength.

For detecting rail further ahead, more traditional image processing methods are used. Lines are traced starting from the histogram based detection. A spline based rail model is adapted to the line detections by optimizing a cost function containing the edge strengths along the model and regularization terms.
11.3 Conclusion

The histogram based rail curvature detection utilizes the additional positional constraints available in the rail scenario to extract the advantages of both an image based detection and a ground plane based detection. Since the rail and train geometry is constrained, bins in the curvature histogram for each rail correspond to fixed, non-overlapping regions. The orientation weighting significantly reduces false curvature responses from strong non-rail edges in the image.
Part IV

Concluding Remarks
Chapter 12

Discussion on Past, Present and Future Work

Exploring the path towards answers to one set of questions, there will always be alternative paths to explore and new questions will arise. This chapter discusses some of the present results and points out a few unexplored areas that have appeared along the way.

The underlying form of the associative learning framework can be interpreted as a single layer artificial neural network. From this, there are two different paths towards better performance. The previously more common way is expanding the complexity of the network, leading to multi-layer networks or networks with even more layers, the deep learning ideas. In the deep learning case, the extreme growth of parameters prohibits the use of fully connected networks, and parameter reducing approaches are used, such as spatially linking parameters over an input, obtaining convolutional layers and convolutional neural networks.

The other path instead uses a more expressive representation of inputs and outputs. Using such an approach, the autonomous road following application have reached abilities comparing favorably to convolutional neural networks while using only a fraction of the possible number of parameters. This allows the associative learning frameworks to be trained online. The limited number of parameters will eventually limit the learning capability. Analyzing the changing structure of eigenvalues of the linkage matrix as the learning progresses may provide some insights, observing the similarity to Hopfield networks.

There are several ways of increasing the capacity of the learning system. Having a collection of different linkage matrices and switching between them have been touched upon earlier. When each linkage matrix is smaller than a photo from an average digital camera, space is not an issue. A different path goes towards more complex learning structures, using multiple layers. This however, may threaten the online learning capability and it reduces the possibilities of analyzing the trained system. These are similar disadvantages as for deep learning approaches. In the end, it will always be interesting to do as much as possible using a system as simple as possible.
The linkage matrix of the associative learning on channels resembles the switchboard idea of Donald Hebb, mapping input ranges to output ranges. However, all variations in behavior could not be explained by this model, or as Hebb wrote:

In mammals even as low as the rat it has turned out to be impossible to describe behavior as an interaction directly between sensory and motor processes. Something like *thinking*, that is, intervenes. “Thought” undoubtedly has the connotation of a human degree of complexity in cerebral function and may mean too much to be applied to lower animals. But even in the rat there is evidence that behavior is not completely controlled by immediate sensory events: there are central processes operating also.

– Donald Hebb, 1949 [44]

Thinking, or rather the result of thinking, can in the current system affect the output in a Bayesian sense by means of output prior distributions fed down from higher-level systems. In a system with multiple linkage matrices, such higher-level systems may also affect the selection of the active linkage matrix.

A linear combination of linkage matrices also makes sense in the associative learning framework. This would correspond to a weighted combination of distributions resulting from the different linkage matrices. With a Bayesian interpretation, the output is a sum of terms where each term is the probability of the particular linkage matrix multiplied with output of the linkage matrix, along the lines of

\[
y = \sum_k P(C_k \mid x, \text{prior expectations}) \, C_k x.
\]

The concept of *sleeping* could generate a hybrid between online and batch learning systems for systems which are not constantly in operation. When the system is not active, i.e. sleeping, there is opportunity to perform rearrangements and compression of linkage matrices and information stored therein. For systems with multiple linkage matrices, this includes splitting and joining such matrices for different cases.

Using deeper Hebbian associative learning systems, one layer may allow online learning as the systems presented earlier, while the other layers are only updated while the system is inactive. The representation may also be updated during sleep, allowing non-regularly spaced channels, placing narrower channels more densely at more important ranges of values, obtaining adaptive foveal vision. In a sense, this corresponds to using channel representations of density functions for remapping values leading towards uniform marginal distributions. Sleeping is closely related to dreaming, evaluating and training the learning system on past or synthesized inputs during inactive periods.

On the theoretical side, the relation joining the channel representation and the kernel density estimation have been explored. A sum of channel encoded values corresponds to a sampled kernel density estimate using the basis function as kernel. Considering this from a sampling point of view, if the true density function convolved with the kernel is limited to half the sampling rate, i.e. the rate of placing basis functions in the value space, the full kernel density estimate
can be obtained by reconstruction using sinc functions. The kernel has a low pass characteristic, but the truncated \( \cos^2 \) kernel with overlap three is not sufficient to guarantee fulfilling the sampling theorem. However, using overlap four, the first zero of the kernel function in frequency space is at half the sampling rate. If there is a further interest in using channel representations for kernel density estimation, these paths should be further explored.

The bird tracking application uses the log-polar channel layout, lending ideas from foveal vision where channels are denser in the center and sparser further out. The supplementary material\(^1\) contains a sequence where each frame has been encoded and reconstructed using a log-polar channel layout. Although interesting, the layout has not yet found its use in many applications. For the bird tracking, where the change rates of foreground and background overlap, there may be an application for time logarithmic channels, where the present is represented with densely placed channels and information further back in time is more sparsely represented. Using such a representation for each pixel, the model will contain much more information. However, it is yet unclear how to utilize that information.

The autonomous road following may improve by using time information. The current approach predicts control signals on a frame by frame basis. The gradient-type Gabor filters can be extended to include a temporal component by keeping a buffer of the last acquired frames. Previous predictions can also be fed back to the system as a prior. However, as long as this is not necessary, this eliminates any possibility of propagating faulty predictions.

All experiments regarding autonomous road following have been performed on a reasonably sized platform. The obvious next step is moving the learning system to a full size platform or onto an unmanned aerial vehicle. Experiments in real traffic will raise questions far beyond computer vision and machine learning.

Approaching the field of autonomous vehicles always raises safety concerns. One common concern is the use of learning systems in autonomous vehicles, systems whose actions may be unpredictable in rare situations. The first thing to note is that this is already the case, however, the systems to learn driving are biological learning systems – humans. Artificial learning systems can be copied and thus one single artificial system can be evaluated in several different scenarios before being distributed to a large number of vehicles. As each human has to be trained and tested individually, that only allows for a far less thorough evaluation of each driver. Furthermore, there are more and more safety systems appearing on the market for collision avoidance and similar tasks. Such systems are currently intended to reduce the effects of unpredictable human drivers. However, these systems can also be used to monitor a learning system for autonomous driving.

\(^1\)http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-125916
Chapter 13

Conclusion

This work has explored a simple learning system capable of learning advanced tasks. Contributions are made both with respect to the learning system and with respect to the representation on which the learning system is based. A selection of applications has been presented, illustrating some possibilities.

The system learns online. Different learning modalities are available, providing the teacher with more freedom, from initial training and short corrections by demonstration to fine tuning through performance feedback. Multiple-hypothesis learning is possible and high-dimensional visual data is processed in real time. Very few learning systems combine all these properties. Furthermore, the system allows uncertainty and multiple hypotheses in the input.

The probabilistic interpretation allows and provides a deeper analysis of the representation. The geometric analysis explores the channel decoding and provides a clear view on cases of biased decoding. Different decoding approaches are compared and found to differ only with respect to the selection of the decoding window. The truncated $\cos^2$ basis functions have been provided with a mathematical foundation demonstrating their uniqueness. The proposed non-linear learning allows for adjusting the balance between common and uncommon training data, breaking the link between learning rate and forgetting rate. The coherence weighting can reduce the effect of non-informative inputs. These ideas are also transferred to visual tracking.

The most prominent application demonstrates online learning of autonomous road following, capable of following all types of roads the system has been tested on. The system is also general enough for learning to follow an object instead. Associative learning using qHebb was the only learning system capable of running online multimodal learning at video rate.

Fully autonomous driving is approaching step by step, the increasingly advanced driver assistance systems continuously drive the development of required components. Interestingly enough, now may be the most challenging time for autonomous vehicles. As such vehicles become more common, road infrastructure may be adapted more towards autonomous driving as an increasing ratio of autonomous vehicles reduces the number of human drivers around.
Bibliography


[85] Ivan Pavlov. The experimental psychology and psychopathology of animals. The 14th International Medical Congress, Madrid, Spain, 1903.


