A dissertation submitted in accordance with the requirements for the degree of Doctor of Philosophy.

Robust Online Subspace Learning

Stephan Liwicki

Imperial College of Science, Technology and Medicine
Department of Computing

August 2014
© 2014 The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text. The research detailed in this work was conducted under the guidance of Professor Maja Pantic and Doctor Stefanos Zafeiriou.

The research presented resulted in several publications with joint authorship:


This thesis contains only my contributions with guidance of my supervisors.

Stephan Liwicki, August 2014
Abstract

In this thesis, I aim to advance the theories of online non-linear subspace learning through the development of strategies which are both efficient and robust. The use of subspace learning methods is very popular in computer vision and they have been employed to numerous tasks. With the increasing need for real-time applications, the formulation of online (i.e. incremental and real-time) learning methods is a vibrant research field and has received much attention from the research community. A major advantage of incremental systems is that they update the hypothesis during execution, thus allowing for the incorporation of the real data seen in the testing phase. Tracking acts as an attractive and popular evaluation tool for incremental systems, and thus, the connection between online learning and adaptive tracking is seen commonly in the literature. The proposed system in this thesis facilitates learning from noisy input data, e.g. caused by occlusions, casted shadows and pose variations, that are challenging problems in general tracking frameworks.

First, a fast and robust alternative to standard \( \ell_2 \)-norm principal component analysis (PCA) is introduced, which I coin Euler PCA (e-PCA). The formulation of e-PCA is based on robust, non-linear kernel PCA (KPCA) with a cosine-based kernel function that is expressed via an explicit feature space. When applied to tracking, face reconstruction and background modeling, promising results are achieved.

In the second part, the problem of matching vectors of 3D rotations is explicitly targeted. A novel distance which is robust for 3D rotations is introduced, and formulated as a kernel function. The kernel leads to a new representation of 3D rotations, the full-angle quaternion (FAQ) representation. Finally, I propose 3D object recognition from point clouds, and object tracking with color values using FAQs.

A domain-specific kernel function designed for visual data is then presented. KPCA with Krein space kernels is introduced, as this kernel is indefinite, and an exact incremental learning framework for the new kernel is developed. In a tracker framework, the presented online learning outperforms the competitors in nine popular and challenging video sequences.

In the final part, the generalized eigenvalue problem is studied. Specifically, incremental slow feature analysis (SFA) with indefinite kernels is proposed, and applied to temporal video segmentation and tracking with change detection. As online SFA allows for drift detection, further improvements are achieved in the evaluation of the tracking task.
Acknowledgements

I thank my supervisor, Prof. Maja Pantic, for her trust, support and encouragement, and for enabling me to pursue my research interests. I am equally grateful to my second supervisor, Dr Stefanos Zafeiriou, for the many discussions, guidance and constructive criticism which helped shape my development as an independent scientist.

I thank the Engineering and Physical Science Research Council for providing me with financial security through their Doctoral Training Account studentship, without which this work would not have been possible. Similarly, I thank Intel for their generous gift through their Doctoral Student Honor Award, which provided the financial support in the later stages of my Ph. D. studies.

I thank the employees of Toshiba Research Europe Limited, and especially Dr Minh-Tri Pham and Dr Björn Stenger, for the fruitful and stimulating experience during my research internship. Our collaboration allowed me to approach my research from a different angle, which also inspired some of the findings reported in this thesis.

I thank the many members, past, visiting and present, of my research group, the intelligent behaviour understanding group, for their companionship. Also, I thank the administrative staff of Imperial College London for their help in the organizational aspect of the Ph. D. degree.

Finally, I thank my family – my parents, Michael Liwicki and Petra Liwicki, who supported me throughout my life and encouraged me to follow my dreams – and my loving wife, Dr Gemma Liwicki, who has always been there for me, in good times and in bad.
# Contents

Abstract

Acknowledgements

List of Tables

List of Figures

List of Algorithms

1 Introduction

1.1 Challenges

1.1.1 Robustness

1.1.2 Efficiency

1.2 Contributions

1.3 Outline

1.3.1 Chapter 2. Background

1.3.2 Chapter 3. Euler Principal Component Analysis

1.3.3 Chapter 4. Full-Angle Quaternion

1.3.4 Chapter 5. Tracking with a Non-positive Kernel

1.3.5 Chapter 6. Incremental Slow Feature Analysis

1.3.6 Chapter 7. Conclusions and Outlook

2 Background

2.1 Subspace Learning

2.1.1 Principal Component Analysis

2.1.2 Slow Feature Analysis

2.1.3 Generalized Eigenvalue Problem

2.2 Online Learning

2.2.1 Incremental Eigenvalue Analysis

2.2.2 Online Learning with Kernels

2.3 Incremental Learning for Object Tracking

2.3.1 Image Representation

2.3.2 Appearance Model

2.3.3 Motion Model

2.3.4 Incremental PCA for Tracking

Contents
3 Euler Principal Component Analysis

3.1 Cosine-based Dissimilarity .............................................. 50
3.2 Euler PCA (e-PCA) ........................................................... 52
  3.2.1 Defining Euler PCA .................................................. 52
  3.2.2 Preimage Computation .............................................. 55
  3.2.3 Incremental Learning ................................................ 58
3.3 Applications and Experiments ......................................... 60
  3.3.1 Image Reconstruction under Noise ................................ 60
  3.3.2 Object Tracking ...................................................... 68
  3.3.3 Background Modeling ............................................... 82
3.4 Conclusions .................................................................. 85

4 Full-Angle Quaternion ......................................................... 87

4.1 Representing 3D Rotations ................................................ 88
  4.1.1 Euler Angles ............................................................ 88
  4.1.2 Rotation Matrix ...................................................... 89
  4.1.3 Rotation Vector ...................................................... 90
  4.1.4 Half-Angle Quaternion ............................................. 90
4.2 Existing Rotation Distances .............................................. 91
  4.2.1 Robust Rotation Distance in 2D ................................. 91
  4.2.2 Rotation Distances in 3D ........................................... 93
4.3 Robust 3D Rotation Distance .......................................... 94
  4.3.1 Proposed Distance ................................................... 94
  4.3.2 Full-Angle Quaternion ............................................. 97
4.4 Applications of Full-Angle Quaternions ............................... 98
  4.4.1 Matching Ball Features for 3D Recognition .................... 99
  4.4.2 Object Tracking in Color Video .................................. 103
4.5 Experiments .................................................................. 107
  4.5.1 3D Object Recognition and Registration ....................... 107
  4.5.2 Object Tracking in Color Video .................................. 110
4.6 Conclusions .................................................................. 114

5 Tracking with a Non-positive Kernel .................................... 115

5.1 Kernel Functions in Krein Space ....................................... 117
5.2 Proposed Indefinite Kernel for Computer Vision .................... 119
  5.2.1 Robust Gradient-based Kernel Function ......................... 119
  5.2.2 Special Kernel Properties .......................................... 121
5.3 Direct Incremental KPCA in Krein Space ............................ 122
  5.3.1 General KPCA with Arbitrary Krein Space Kernels .......... 123
  5.3.2 Direct KPCA in Krein Spaces ..................................... 125
  5.3.3 Direct Incremental KPCA ........................................... 126
5.4 Visual Object Tracking ..................................................... 129
5.5 Results and Discussion .................................................... 130
  5.5.1 Object Tracking ....................................................... 130
  5.5.2 Face Recognition ..................................................... 148
5.6 Conclusions .................................................................. 150
2.1 A summary of generalized eigenvalue problems. 40
3.1 Comparison of normalized dissimilarity measures. 52
3.2 Mean RMS error for general tracking, and tracking rate. 75
3.3 Maximum similarity on background modeling. 83
3.4 Execution time required to compute (or update) the appearance model. 85
4.1 Weighting strategies for different methods. 107
4.2 Object level registration rates of all methods. 108
4.3 Summary of the results for object recognition. 109
4.4 The different tracking setups. 111
4.5 Mean RMS error for all videos and trackers. 111
5.1 Mean RMS error for tracking with video-specific parameters. 135
5.2 Mean RMS error for tracking with fixed parameters. 145
5.3 Average recognition rate with Yale B Database. 148
5.4 Average recognition rate with CMU PIE Database. 149
6.1 Mean RMS errors for DIKT and CD-DIKT with change detection. "lost" to indicate the sequences in which the tracker does not follow the target throughout, after visual inspection. 184
# List of Figures

1.1 Overview of the proposed tracking system. ........................................ 19
1.2 Subspaces using linear, quadratic and arbitrary feature representations. ... 20
1.3 Offline learning versus online learning. ........................................... 21
1.4 Examples of common challenges in visual tracking. ............................. 23

2.1 Simple illustration showing the concepts of subspace learning. ............... 30
2.2 Illustration of the non-linear mapping, using $\phi$. ............................. 34
2.3 Reconstruction for kernel PCA using preimage approximation. ............... 35
2.4 Illustration of hierarchical SFA. ..................................................... 37
2.5 SFA for temporal video segmentation. ............................................. 38
2.6 Tracking framework with incremental subspace learning. ........................ 47

3.1 Cosine-based dissimilarity measure .............................................. 51
3.2 Motivating example for the use of the cosine-based dissimilarity measure ... 52
3.3 Visualization of the explicit mapping with different $\alpha$. .................... 53
3.4 Illustration of the preimage computation. ....................................... 58
3.5 Cropped example images from the AR Database ................................ 61
3.6 Reconstructed images after learning with random corruptions. ............... 63
3.7 Reconstruction error and angular error for different values of $\alpha$. ......... 64
3.8 Approximation error for $e$-PCA and $e$-PCA-GA. ............................ 65
3.9 Reconstruction error with different rates of occlusions....................... 66
3.10 Angular error with different rates of occlusions. ............................... 67
3.11 Examples of the corruptions in the training data. ............................. 68
3.12 Reconstruction error with different rates of hand occluded images .......... 69
3.13 Angular error with different rates of hand occluded images. ................. 70
3.14 Reconstructed images after learning with skin-like corruptions. .......... 71
3.15 The Euler tracker framework with incremental $e$-PCA. ...................... 72
3.16 Mean RMS error of all tested trackers at each frame. ....................... 76
3.17 Tracking precision for each video sequence. ................................... 77
3.18 Qualitative tracking results for Vid1. .......................................... 78
3.19 Qualitative tracking results for Vid2. .......................................... 78
3.20 Qualitative tracking results for Vid3. .......................................... 79
3.21 Qualitative tracking results for Vid4. .......................................... 79
3.22 Qualitative tracking results for Vid5. .......................................... 80
3.23 Qualitative tracking results for Vid6. .......................................... 80
3.24 Qualitative tracking results for Vid7. .......................................... 81
3.25 Qualitative tracking results for Vid8. ........................................... 81
3.26 Similarity with changing number of components. ......................... 82
3.27 Examples of background modeling for each video and each method. 84
3.28 Per frame running time of e-PCA and its batch version e-PCA-batch. 85

4.1 Visual comparison between rotation representations. ....................... 89
4.2 Cosine-based distance function for different angle differences. ........ 92
4.3 The proposed cosine-based distance for 3D rotations. ....................... 94
4.4 Histograms of the normalized squared distances. .......................... 96
4.5 Object recognition based on oriented ball features. ........................ 100
4.6 Tracking results of the proposed FAQ tracker. .............................. 110
4.7 RMS error of each frame for Vid2, Vid3, Vid5 and Vid7. .................. 112
4.8 Precision graphs of the object tracking results. ............................. 113

5.1 Visualization of the proposed kernel when corruption is present. ...... 120
5.2 The direct incremental KPCA tracker (DIKT) framework. ................. 130
5.3 Mean RMS error for video-specific trackers at each frame. .............. 133
5.4 Tracking precision of video-specific trackers for each video sequence. 134
5.5 Example frames of the tracking with video-specific parameters for Vid1. 136
5.6 Example frames of the tracking with video-specific parameters for Vid2. 137
5.7 Example frames of the tracking with video-specific parameters for Vid3. 138
5.8 Example frames of the tracking with video-specific parameters for Vid4. 139
5.9 Example frames of the tracking with video-specific parameters for Vid5. 140
5.10 Example frames of the tracking with video-specific parameters for Vid6. 141
5.11 Example frames of the tracking with video-specific parameters for Vid7. 142
5.12 Example frames of the tracking with video-specific parameters for Vid8. 143
5.13 Example frames of the tracking with video-specific parameters for Vid9. 144
5.14 Mean RMS error for all trackers with fixed parameters at each frame. 146
5.15 Tracking precision of each video sequence with fixed tracking parameters. 147
5.16 The data, as projected by the subspaces of PCA, Gaussian and DKPCA. 148

6.1 Illustration of incremental SFA at initialization (time 0) and time step t. 161
6.2 Illustration of a preimage optimization setup. ............................... 166
6.3 Tracking with multiple appearance models, using change detection. 172
6.4 The distance between a data point and the mean of a subspace. ........ 175
6.5 A comparison of different SFA implementations. ............................. 176
6.6 Angular error between batch and online learned subspaces. .......... 178
6.7 Angular error with and without reduced set. .................................. 178
6.8 False positive over false negative rate of different kernel setups. .... 179
6.9 Change detection for D-Krein with direct mapping. ....................... 180
6.10 The three slowest features after 900 frames using the linear kernel. 181
6.11 Segmentation for yoga and ballet scenes. .................................... 182
6.12 False positive over false negative rate comparison. ...................... 183
6.13 Tracking results and models of appearances. .............................. 184
List of Algorithms

3.1 Estimating the Principal Subspace ........................................... 54
3.2 Embedding of new Samples ..................................................... 54
3.3 Incremental Subspace Estimation .............................................. 59
3.4 Euler Tracker at Time $t$ ......................................................... 73

4.1 Offline Phase: Creating Hash Tables ......................................... 101
4.2 Online Phase: Vote Evaluation .................................................. 102
4.3 Quaternion Subspace Estimation ................................................ 106
4.4 Incremental Quaternion Subspace Estimation ............................. 106

5.1 Direct KPCA in Krein Space ..................................................... 127
5.2 Update Procedure of Direct Incremental KPCA .......................... 129
5.3 DIKT at Time $t$ ................................................................. 131

6.1 Batch SFA in Krein Space ....................................................... 159
6.2 Testing SFA in Krein Space ..................................................... 159
Chapter 1

Introduction

The aim of my thesis, is to advance the theory of online non-linear subspace learning, and kernel methods in particular, through the development of fast and robust incremental methods, which are applicable to adaptive real-time object tracking in realistic environments. Figure 1.1 shows the general layout of the proposed system.

**Figure 1.1:** Overview of the proposed tracking system. The main aim of my research is the creation of an efficient and robust adaptive appearance model through non-linear online subspace learning.
Subspace learning has a long-standing history in machine learning and pattern recognition, and is employed in many common applications (Bishop 2006; Jolliffe 2002). Its primary aim is to map high-dimensional data into a (possibly) lower-dimensional space, in which the important properties of the original samples are retained or even exaggerated. For example, principal component analysis (PCA) extracts the directions with largest variance from a population of data points (figure 1.2(a)). PCA is often considered the most prominent subspace algorithm, as its solution can be computed efficiently via eigenvalue decomposition, and its applications are numerous (Jolliffe 2002). Other subspace methods which are solvable by an eigenvalue problem include slow feature analysis (SFA) and linear discriminant analysis (LDA).

While the subspace of linear component analysis can be thought of as a transformation from original space onto a hyper-plane, non-linear subspace learning can be more complex (figure 1.2(b) and figure 1.2(c)). The advantage is that complex and robust feature spaces can be employed by the expressive non-linear setup. Disadvantageously, in PCA learning for example, finding the solution commonly becomes more difficult and inefficient.

Another aspect of subspace learning is the model acquisition. Traditional methods extract informative properties from a set of training data offline. On the other hand, online (i.e. incremental and real-time) systems improve the hypothesis of the learning algorithm by considering the available test data as it arrives (figure 1.3). With an ever increasing importance of realizing adaptive applications with minimal \textit{a priori} knowledge, online learning methods have become a popular research topic in recent years (Babenko et al. 2011; Ross et al. 2008).

The remainder of this chapter is organized as follows. In section 1.1, I will introduce the challenges of robust online subspace learning in the context of visual object tracking.
Figure 1.3: Offline learning versus online learning. Online systems require minimal training as the hypothesis is reinforced by incoming test data.

Section 1.2 lists the contributions of my work, and section 1.3 outlines the contents of the chapters in this thesis.

1.1 Challenges

Closely related to online learning is the challenge of visual object tracking. In fact, tracking is a very popular evaluation technique for many incremental systems (Babenko et al. 2011; Chin and Suter 2007; Ross et al. 2008). Tracking applications are ubiquitous and reach from automation and validation of common tasks (Collins et al. 2001; Haritaoglu et al. 2000; Hsieh et al. 2006; Kamijo et al. 2000) to human-computer interaction and behavior analysis (Cohn et al. 1999; Gunes and Pantic 2010; Wren et al. 1997). In the following, the demands of a real-time tracking system for realistic environments are compared to the task of incremental learning. Specifically, the tracking system should

1. run in real-time,
2. support occlusions and corruptions from noisy data,
3. handle varying illumination and casted shadows,
4. allow for pose variation, and
5. adapt to appearance changes.

In the subsequent sections, I will detail these challenges in the context of tracking with an online learning system, designed for robustness and efficiency.
1.1.1 Robustness

Plentiful tracking systems have been introduced to the computer vision community. However, tracking for real environments remains unsolved. In their survey, Yilmaz et al. (2006) reveal that complex motion, changes in illumination, occlusions and non-rigid appearance variations are some examples which cause most trackers to fail. Figure 1.4 shows instances in which these challenges occur.

Occlusions

An occlusion refers to the tracking instances in which other objects are positioned between the recording device and the target. This means that parts of the object’s appearance will be hidden from view, causing corruptions or noisy data samples in the learning system.

Illumination Changes

Illumination changes are caused by alterations in the lighting conditions. Global as well as local changes may occur. Depending on the choice of feature space, illumination changes can cause drastic variation in the image as often a large proportion of pixels are affected.

Casted Shadows

Casted shadows occur especially in sunny outdoor scenes and in circumstances where the target traverses several different lighting conditions. From a computer vision perspective, casted shadows are akin to a combination of occlusions and illumination changes, as the shadowed pixels change their intensity values, while shadow boundaries may cause noise and corruption in feature space.

Complex Motion

While the trajectory of vehicles on roads is typically very expectable, object trackers for arbitrary target classes must be able to cope with less predictable motion. The target may move out of the frame or cause motion blur through fast movements. Furthermore, their path may suddenly be altered by other objects in the scene.

Pose Variation

Dependent on the motion model, pose variation can be challenging when out-of-plane rotations or other unsupported movements occur. Typically pixels of the background may corrupt the features used in the tracking framework, and thus noisy data is received.
Figure 1.4: Examples of common challenges in visual tracking.
Intrinsic Appearance Changes

Intrinsic appearance changes predominantly arise in non-rigid or compound object classes. In particular, objects may be added or removed from the original appearance, or the surface of the object is stretched, e.g. during emotion changes in face tracking.

1.1.2 Efficiency

In real-time applications, efficiency is vital. Thus it is essential to develop systems whose complexity remain unchanged over time. Clearly, if test data is to be utilized for learning, standard offline batch approaches, learned at each time-step, will not be feasible. Instead, an incremental acquisition of the model is required whose footprint does not grow during execution.

Ross et al. (2008) introduce a linear online subspace algorithm based on PCA. Their approach allows for fast computation and constant running speed. Outliers and noisy data are however not supported in their system. Other approaches, e.g. the non-linear kernel method by Chin and Suter (2007), apply more robust learning frameworks. These methods support outliers but at a cost, as they inadvertently introduce expensive optimization problems or approximations, which reduce performance. Solutions that combine efficiency and robustness are still seldom found in the literature of online subspace learning.

1.2 Contributions

In this thesis, I present the following contributions to the computer vision community:

1. I introduce several robust kernel functions for measuring visual similarities. Instead of using off-the-shelf kernels which are often not robust, I develop dissimilarity measures based on robust concepts and features in computer vision.

2. As one of the proposed kernel function is indefinite (i.e. non-positive definite), I extend the existing literature on non-linear kernel PCA (KPCA) from positive definite kernels in Hilbert space to general kernels in Krein space. Furthermore, the generalized eigenvalue problem, e.g. in SFA, is extended to handle indefinite Krein space kernels.

3. Online learning is instantiated for KPCA and kernel SFA (KSFA) via efficient and accurate incremental algorithms, which employ a special family of kernels. These frameworks are applicable to all kernels proposed in this thesis.
4. The introduced online subspace learning framework for KPCA, in combination with my robust kernels, is applied to visual tracking, for which state-of-the-art performance is achieved.

5. Additionally, I propose an online learning framework for the generalized eigenvalue problem in KSFA with arbitrary kernels in Krein space. Here, an optimization problem is employed to ensure constant memory and time complexity.

6. Finally, incremental KSFA is used to implement SFA’s first online temporal video segmentation, and tracking with change detection.

1.3 Outline

This thesis consists of 7 chapters in total. In this section I outline and summarize the content of the remaining parts.

1.3.1 Chapter 2. Background

Chapter 2 presents the related research in literature. The focus lies on subspace learning and incremental eigenvalue analysis. Moreover, kernel methods and online kernel learning is discussed. Finally, visual object tracking is introduced in the context of adaptive appearance models via online learning.

1.3.2 Chapter 3. Euler Principal Component Analysis

A robust and efficient kernel subspace method, called Euler PCA (e-PCA), is proposed in chapter 3. In particular, e-PCA employs a robust dissimilarity measure based on the Euler representation of complex numbers. In my work, I show that e-PCA suppresses outliers whilst retaining standard PCA’s desirable properties. An incremental version of e-PCA is introduced for adaptive object tracking. Finally, two further computer vision applications are employed for evaluation, and the proposed method performs comparably to other state-of-the-art approaches – but faster.

Related Publications


1.3.3 Chapter 4. Full-Angle Quaternion

3D data is commonly present in computer vision. In chapter 4, I introduce a new distance for robustly matching vectors of 3D rotations. A special representation of 3D rotations, which I coin full-angle quaternion (FAQ), then allows me to express this distance as Euclidean. The proposed distance is applied to 3D shape recognition and 2D object tracking from color video. For the former, I use the concurrent and efficient hashing for scale and translation proposed in collaboration with Toshiba Research Europe Limited (Toshiba) (Liwicki et al. 2014) to form vectors of rotations. For the latter, incremental PCA is performed with the proposed FAQ representation to highlight its benefit for online subspace learning.

Related Publication


1.3.4 Chapter 5. Tracking with a Non-positive Kernel

In chapter 5, I propose an exact framework for online learning with a family of indefinite (not positive definite) kernels. As the case of indefinite kernels is studied, KPCA is first extended from a reproducing kernel Hilbert space (RKHS) to Krein space. An incremental KPCA in Krein space is then presented. The approach of this chapter is motivated by the challenge of visual tracking. In particular, I develop a gradient-based kernel function with robust properties. The incremental framework is tested on several popular and difficult tracking scenarios. An evaluation on face recognition is additionally presented.

Related Publication


1.3.5 Chapter 6. Incremental Slow Feature Analysis

Slow feature analysis (SFA) is a subspace learning technique which has been linked to how visual brain cells work. In recent years, SFA has been adopted for computer vision tasks. Chapter 6 proposes an exact KSFA framework for positive definite and indefinite
kernels in Krein space. Online KSFA which employs a reduced set expansion is then formulated. Finally, by exploiting a special kind of kernel family, I propose an efficient online KSFA algorithm for which no reduced set is required. The proposed incremental system is applied to develop SFA-based change detection for stream data, and a temporal video segmentation algorithm is presented. I test my setup on synthetic and real data streams. When combined with an online learning tracking system from previous chapters, the proposed change detection approach improves upon the original performance, which does not incorporate change detection.

Related Publications


1.3.6 Chapter 7. Conclusions and Outlook

Chapter 7 concludes this thesis. I summarize the most important findings of previous chapters, and possible future research paths are introduced.
Chapter 2

Background

LIKE MANY MACHINE learning techniques, subspace learning aims to describe complex sets of data points through simpler representations. Consider a set of samples \( \{x_n\}_{n=1}^N \) where each sample \( x_n \) resides in a high-dimensional space. In subspace learning, the aim is to map the samples onto a low-dimensional sample set \( \{y_n\}_{n=1}^N \), such that the underlying meaning of the original space is retained. For example, in some subspace setups it is advantageous to retain the distinguishing parts to find the class labels of the points. Thus, a subspace is a mapping that reduces the original dimensionality, by extracting and combining only informative dimensions.

I will review the most relevant techniques for subspace learning in section 2.1. Particular focus lies on systems which may be expressed as a generalized eigenvalue problem. Incremental eigenvalue analysis, and general online learning is discussed in section 2.2. Finally in section 2.3, arguably the most prominent application of online systems, visual object tracking, is presented in the context of adaptive appearance models that learn during execution.

2.1 Subspace Learning

The need for subspace learning can be found in many applications of machine learning and pattern recognition (Bishop 2006). It is widely employed for de-noising (Broida and Chellappa 1986; Mika et al. 1999), reducing the cost of computing with high-dimensional data (Bishop 2006; Yeung and Ruzzo 2001), or compact feature extraction (Black and Jepson 1998; Nguyen and de la Torre 2009; Tzimiropoulos et al. 2012b). Figure 2.1 illustrates the workings of a subspace approach. Original samples \( \{x_n\}_{n=1}^N \) in a high-dimensional space \( x_n \in \mathcal{X} \) are transformed into a low-dimensional subspace, such that the samples are mapped to \( \{y_n\}_{n=1}^N \), \( y \in \mathcal{Y} \). The mapping function \( \psi : \mathcal{X} \rightarrow \mathcal{Y} \) which transforms any point in \( \mathcal{X} \) onto \( \mathcal{Y} \) is sought. The main challenge is to find a mapping
ψ which preserves the information of the original space. For this, many strategies exist. The most relevant methods for this thesis are discussed in the following.

2.1.1 Principal Component Analysis

In pattern recognition, PCA is perhaps the most classical tool for dimensionality reduction and feature extraction. It is widely employed in a great variety of disciplines, including agriculture, biology and economics (Jolliffe 2002). Researchers in computer vision employ PCA for face recognition (Turk and Pentland 1991), object tracking (Ross et al. 2008), background modeling (Li 2004) and many other applications (Jolliffe 2002). It is primarily used for efficient dimensionality reduction, which preserves the variance of the original high-dimensional data. Different formulations of PCA exist in the literature. I will now summarize these in the following.

Standard $\ell_2$-norm PCA

The most prominent version of PCA is based on the $\ell_2$-norm. Consider the set $\{x_n\}_{n=1}^N$ where $x_n \in \mathbb{R}^P$ is a $P$-dimensional vector, in the high-dimensional original space. Let us write the population of $N$ samples in matrix format, by generating the sample matrix $X = \left[ x_1 \, \cdots \, x_N \right] \in \mathbb{R}^{P \times N}$ (without loss of generality, a zero mean is assumed). Then standard PCA is employed to find a set of $R$ orthogonal basis functions $\{u_r\}_{r=1}^R$, which project the original space onto subspace. The dimensionality of this subspace is guaranteed to be less than or equivalent to the dimensionality of the original space, i.e. $R \leq P$ – usually $R \ll P$. The projection matrix $U = \left[ u_1 \, \cdots \, u_R \right] \in \mathbb{R}^{P \times R}$ is designed to minimize the $\ell_2$-norm between samples and their reconstruction, i.e.

$$U = \min \arg \|X - \tilde{U}^T X\|_F,$$  \hspace{1cm} (2.1)
where $\| \cdot \|_F$ denotes the Frobenius norm and $(\cdot)^T$ computes the transposition of a matrix.\textsuperscript{1} The above optimization problem is equivalent to the maximization problem

$$U = \max \arg \text{tr}\ (\hat{U}^TXX^T\hat{U})$$
subject to $U^TU = I,$ \hfill (2.2)

where $\text{tr}(\cdot)$ computes the trace of a matrix. Its solution is provided by the eigenvalue decomposition of the so-called scatter matrix $S = XX^T = \hat{U}\Sigma\hat{U}^T$, where $\Sigma$ is a diagonal matrix containing the eigenvalues, and $\hat{U}$ contains the eigenvectors. In particular, given the decomposition of the scatter matrix, eq. (2.2) can be expanded to

$$U = \max \arg \text{tr}\ (\hat{U}^T\hat{U}\Sigma\hat{U}^T\hat{U})$$
$$= \max \arg \text{tr}(\Sigma)$$
$$= \max \arg \sum_r \lambda_r$$
subject to $U^TU = I,$ \hfill (2.3)

where $\lambda_r$ is the $r$th largest eigenvalue in $\Sigma$. Thus, the solution of standard PCA is given by the eigenvectors corresponding to the $R$ largest eigenvalues obtained from the eigenvalue decomposition of scatter matrix $S$. Alternatively, the singular value decomposition (SVD) of the sample matrix $X = \hat{U}\Sigma^{1/2}\Omega^T$ can be employed. Notice, as $\Omega$ and $\hat{U}$ are orthonormal matrices, we may write

$$XX^T = \hat{U}\Sigma^{1/2}\Omega^T\Omega\Sigma^{1/2}\hat{U}^T = \hat{U}\Sigma\hat{U}^T.$$ \hfill (2.4)

Finally the mapping of any sample $x \in \mathbb{R}^P$ is given by $\psi(x) = U^Tx$.

Sometimes, the inverse mapping $\psi^{-1}$ is utilized. For example, image de-noising removes small varying pixel noise by means of projection and reconstruction of a sample $x$, via $\psi^{-1}(\psi(x)) = \hat{U}U^Tx$.

**Robust Principal Component Analysis**

Often encountered in learning are corrupted data samples. Noisy data is a common problems in applications such as image registration (Tzimiropoulos et al. 2012b), image matching (Fitch et al. 2005), shape matching (Grauman and Darrell 2005), face recognition (Marras et al. 2012), and feature learning (Nguyen and de la Torre 2009). Formally, Hawkins (1980) defines an outlier as an observation which deviates significantly from the

\textsuperscript{1}This formulation preserves the original data.
other observations, arousing suspicion that it was generated by a different mechanism. These phenomena are also referred to as abnormalities, discordan ces or deviants in liter-ature (Aggarwal 2013), and ought to be suppressed when modeling the data. The error function in eq. (2.1), however, is based on the $\ell_2$-norm which is optimal only for independent and identically distributed Gaussian noise. It is therefore extremely sensitive
to gross errors caused by non-Gaussian outliers (de la Torre and Black 2003). Recent
methods attempt to mitigate this sensitivity by adopting different error functions.

The statistically more robust $\ell_1$-norm as distance measure is introduced to PCA learning by Ke and Kanade (2003, 2005). In their setups, the projection $\mathbf{U}$ is optimized via

$$
\mathbf{U} = \min \arg \mathbf{U} \| \mathbf{X} - \hat{\mathbf{U}}\hat{\mathbf{U}}^\top \mathbf{X} \|_1 = \min \arg \mathbf{U} \sum_{n=1}^{N} \| x_n - \hat{\mathbf{U}}\hat{\mathbf{U}}^\top x_n \|_1,
$$

where $\| \cdot \|_1$ computes the sum, composed out of the absolute values in a matrix or
vector. The proposed $\ell_1$-norm minimization is based on either the weighted median
algorithm (Ke and Kanade 2003) or convex quadratic programming (Ke and Kanade
2005). Consequently, while these approaches reduce the effect of outliers, the optimization
of eq. (2.5) is computationally expensive. Moreover, both methods are not invariant to
rotations, which is an important property of learning algorithms (Ding et al. 2006).

Approximations of $\ell_1$-norm PCA include the PCA-L1 by Kwak (2008) and the R1-PCA by Ding et al. (2006). In particular, PCA-L1 estimates the optimum in eq. (2.5) with a componentwise greedy search for

$$
\mathbf{U} = \arg \max \mathbf{U} \| \hat{\mathbf{U}}^\top \mathbf{X} \|_1.
$$

R1-PCA approximates eq. (2.5) by finding the optimum of

$$
\mathbf{U} = \min \arg \mathbf{U} \sum_{n=1}^{N} \left( \sum_{p=1}^{P} x_n(p) - \sum_{r=1}^{R} \hat{u}_r(p)\hat{u}_r(p)x_n(p) \right)^2,
$$

where $\{\hat{u}_r\}_{r=1}^{R}$ are the components in $\hat{U}$, and $a(b)$ extracts the $b^{th}$ value from vector $a$. Both methods allow for faster convergence towards the solution. Furthermore, both are rotational invariant.

de la TorreDE LA TORRE and Black (2003) use theories of M-Estimation to reformulate PCA robustly. Note, M-Estimators are commonly linked to robust statistics, as they decrease the impact of “non-normal” data. He et al. (2011b) introduce HQ-PCA, which is rotational invariant and robust. Their approach is based on the maximum correntropy criterion (MMC) (Liu et al. 2007), which is closely related to M-Estimators.
Here, the objective function is efficiently optimized by the half-quadratic optimization technique. Candès et al. (2009) represent PCA as an optimization problem which finds a low dimensional linear subspace and a sparse matrix which relates to outliers.

With the introduction of robustness, commonly comes a decrease in efficiency. In particular, a disadvantage of the robust methods above is the lack of an elegant, closed-form solution for the optimization problems.

**Non-linear Kernel Principal Component Analysis**

Kernel PCA (KPCA) allows for linear formulations of non-linear subspaces, which can be expressed via a simple extension to standard \( \ell_2 \)-norm PCA. Notice, the flexibility of non-linear subspaces might provide solutions to some of the problems often associated with robustness in linear PCA, while conserving efficiency. Before presenting further details, let me first review the general idea of kernel learning (Schölkopf et al. 1997).

Recall, in \( \ell_2 \)-norm PCA, data points reside in an Euclidean space, where the squared Euclidean distance between two points \( x_i \) and \( x_j \) is given by

\[
||x_i - x_j||^2 = x_i^T x_i - x_j^T x_j - x_i^T x_j - x_j^T x_i + x_j^T x_j = \langle x_i, x_i \rangle - \langle x_j, x_i \rangle - \langle x_i, x_j \rangle + \langle x_j, x_j \rangle. \tag{2.8}
\]

Here \( \langle ., . \rangle \) denotes the inner product (or dot-product) between two vectors.

In KPCA, the samples lie in a non-linear feature space, which is hyper-dimensional. That is, considering the original sample \( z \in \mathbb{R}^P \) in Euclidean space, kernel methods assume an implicit mapping into feature space – known as Hilbert space \( \mathcal{H} \). Essentially, the sample \( z \) is implicitly mapped as sample \( x = \phi(z) \) using an unknown mapping function \( \phi : \mathbb{R}^P \to \mathcal{H} \) (figure 2.3). Function \( \phi \) is defined by means of a kernel \( k : \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R} \) which expresses the comparison between two data points, i.e. the inner product in feature space. Formally, a Hilbert space \( \mathcal{H} \) is a complete vector space which is provided by an inner product \( \langle ., . \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R} \), which is realized by employing the kernel \( k \), that compares two samples \( z_i \) and \( z_j \) via an implicit feature space, i.e.

\[
\langle \phi(z_j), \phi(z_i) \rangle = k(z_i, z_j). \tag{2.9}
\]

In many applications, the kernel \( k \) is employed and the implicit mapping \( \phi \) is never explicitly required (Böhmer et al. 2011; Paulsen 2009; Schölkopf et al. 1997; Shawe-Taylor
Figure 2.2: Illustration of the non-linear mapping $\phi$. The original space is implicitly transformed into Hilbert space, where linear PCA can be performed (red line) via the kernel trick.

and Cristianini 2004). For example, the squared distance between two samples in the non-linear Hilbert space is given via the kernel as

$$
\|\phi(z_i) - \phi(z_j)\|^2 = \langle \phi(z_i), \phi(z_i) \rangle - 2 \langle \phi(z_i), \phi(z_j) \rangle + \langle \phi(z_j), \phi(z_j) \rangle
$$

(2.10)

Thus, kernel methods allow us to transform non-linear feature spaces into (possibly “fictitious”) linear spaces, in which standard linear methodologies apply.

Notice however, for the inner product to be valid, the kernel is traditionally required to be semi-positive definite (Schölkopf et al. 1997). Hence, given any samples $x_i, x_j, x_k \in H$ in Hilbert space, and any two scalar values $a, b \in \mathbb{R}$, the following properties are important for the inner product in Hilbert space:

$$
\langle x_j, x_i \rangle = \langle x_i, x_j \rangle \quad (2.11)
$$

$$
\langle ax_j + bx_k, x_i \rangle = a\langle x_j, x_i \rangle + b\langle x_k, x_i \rangle \quad (2.12)
$$

$$
\|x_i - x_k\| \leq \|x_i - x_j\| + \|x_j - x_k\|. \quad (2.13)
$$

In the context of robust PCA learning, it is easy to see that KPCA is especially useful for extracting features from a non-linear space (Schölkopf et al. 1997; Shawe-Taylor and Cristianini 2004). For example, consider the set of samples in input space $\{z_n\}_{n=1}^N$ and their implicit sample matrix $X = \begin{bmatrix} \phi(z_1) & \cdots & \phi(z_N) \end{bmatrix}$ in feature space (zero mean is assumed without loss of generalization). The SVD of matrix $X$, needed for solving PCA, is found by expressing the searched-for subspace as linear combination, i.e. $U = X \tilde{U}$. In particular, for KPCA, eq. (2.2) is rewritten as

$$
\tilde{U} = \max_{\tilde{U}} \text{tr} \left( \tilde{U}^T XX^T \tilde{U} \right) \\
\text{subject to } U^T U = \tilde{U}^T X^T X \tilde{U} = I. \quad (2.14)
$$
Figure 2.3: Reconstruction for kernel PCA using preimage approximation. The original sample \( z \in \mathcal{Z} \) is mapped implicitly into Hilbert space \( x \in \mathcal{H} \) and then onto the low-dimensional space \( y \in \mathcal{Y} \). The reconstruction finds \( \hat{z} \) by reverse-engineering this setup. Although space \( \mathcal{H} \) is not explicitly known, comparisons between samples can be performed via the kernel.

Following Turk and Pentland (1991), given the eigenvalue decomposition of the squared kernel matrix \( K^2 = (X^TX)^2 = \Omega \Lambda^2 \Omega^T \), the SVD of the sample matrix is provided by

\[
X = \left[ X\Omega \Lambda^{-\frac{1}{2}} \right] \left[ \Lambda^\frac{1}{2} \right] \Omega^T,
\tag{2.15}
\]

where the partial subspace is given by \( \tilde{U} = \Omega \Lambda^{-\frac{1}{2}} \). Finally, the mapping \( \psi \) of applying the subspace to a new sample in input space \( z \in \mathbb{R}^P \), is computed via a sum of kernel functions which requires all training samples in \( \{z_n\}_{n=1}^N \), as

\[
\psi(z) = \tilde{U}^T X^T \phi(z) = \tilde{U}^T \begin{bmatrix}
\phi(z_1)^T \phi(z) \\
\vdots \\
\phi(z_N)^T \phi(z)
\end{bmatrix} = \tilde{U}^T \begin{bmatrix}
k(z_1, z) \\
\vdots \\
k(z_N, z)
\end{bmatrix}.
\tag{2.16}
\]

Similarly to PCA, KPCA is also suited to the problem of sample de-noising via reconstruction. For this, preimage computation is employed (Honeine and Richard 2011; Kwok and Tsang 2004; Mika et al. 1999). Specifically, given the positive definite kernel function \( k \) and its related implicit mapping \( \phi \) into Hilbert space, as discussed above, KPCA learns a linear subspace matrix \( U = \begin{bmatrix} u_1 & \ldots & u_R \end{bmatrix} \), whose projections \( \{u_r\}_{r=1}^R \) lie in the hyper-dimensional, implicit Hilbert space. Mika et al. (1999) de-noise an unmapped sample \( z \in \mathbb{R}^P \) in its feature space \( \phi(z) \) by solving the following optimization problem:

\[
\hat{z} = \arg \min_{\tilde{z}} \| \phi(\tilde{z}) - UU^T \phi(\tilde{z}) \|^2.
\tag{2.17}
\]

Figure 2.3 visualizes the reconstruction. Notice, the above optimization problem aims to find a sample \( \hat{z} \) (i.e. the preimage) in input space such that its mapping in the feature space \( \phi(\hat{z}) \) approximates the reconstruction in the feature space \( UU^T \phi(\hat{z}) \) most accurately. The solution can be found through gradient descent.
While kernel methods, and KPCA in particular, provide a flexible framework for deriving functions that compare data samples, limited work in the literature actually employs robust kernels. Notice, popular kernels, such as Gaussian radial basis functions (GRBFs), do not possess robust properties by definition. One of the few methods that address robust subspace estimation in non-linear feature spaces is introduced by Nguyen and de la Torre (2009), who kernelize the linear robust PCA by de la Torre and Black (2003), discussed above.

2.1.2 Slow Feature Analysis

In contrast to PCA which is commonly employed in computer vision, SFA has only recently found its way into the computer vision community, where it is used as an unsupervised subspace learning technique with sequential data.

SFA originates from theories related to neural networks (Wiskott and Sejnowski 2002). Over the years, extensive studies in neural science have found similarities between SFA and the properties of brain cells in the visual cortex (Berkes and Wiskott 2005; Wiskott and Sejnowski 2002). In particular, Berkes and Wiskott (2005) analyze the learning behavior of SFA when trained on image sequences. In their experiments a significant correlation between the learned SFA components and the brain cells in the primary visual cortex is revealed.

SFA forms an important aspect of this thesis, due to its ability to learn from unlabeled input, and its applicability to finding “surprising” changes in temporally arranged data. After formally providing SFA’s optimization problem, I review some recent work in computer vision that utilize slow features as subspace in the following.

SFA’s Optimization Problem

Let me formalize the optimization problem related to SFA. Given a set of temporally arranged input data \( \{x_n\}_{n=1}^N \), the sequence can be expressed as sample matrix \( X = \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix} \in \mathbb{R}^{P' \times N} \). SFA is defined as the subspace that produces mappings \( \{y_n\}_{n=1}^N \), which change slowest over time (Wiskott and Sejnowski 2002). In particular, given the output space’s derivatives \( \dot{y}_n \), their sum of squared lengths, i.e. \( \sum_{n=1}^N \|\dot{y}_n\|^2 \), is to be minimized. Furthermore, to avoid a trivial solution, Wiskott and Sejnowski (2002) constrain the output samples \( \{y_n\}_{n=1}^N \) to be of zero mean with uncorrelated dimensions of unit variance.

36 2.1. Subspace Learning
Figure 2.4: Illustration of hierarchical SFA. Hierarchical SFA learns from a subset of dimensions and combines these layer by layer. This reduces computational cost, as only few dimensions are considered at each layer.

Berkes and Wiskott (2005) find the transformation into subspace by means of a generalized eigenvalue problem, equivalent to the trace-problem

$$U = \arg \min_U \text{tr} \left( \left( \hat{U}^T X X^T \hat{U} \right)^{-1} \hat{U}^T \hat{S} \hat{U} \right),$$

(2.18)

where $S = X X^T$ is the scatter matrix of the samples, and $\hat{S} = \hat{X} \hat{X}^T$ represents the scatter of their derivatives, which is often approximated by simple subtraction, i.e. $x_n \approx x_n - x_{n-1}$ (Böhmer et al. 2011; Nater et al. 2011). Note, in my notation both scatter matrices assume a zero mean without loss of generalization. As before, the mapping onto subspace is provided by $\psi(x) = U^T x$.

**SFA in Computer Vision**

In computer vision, SFA is adopted to learn from temporally arranged data signals such as video (Franzius et al. 2008; Kompella et al. 2012; Nater et al. 2011; Zafeiriou et al. 2013). Like PCA, it extracts an orthogonal subspace from the input data. Differently to PCA however, SFA considers the temporal information to find the most descriptive components that vary slowest over time (Wiskott and Sejnowski 2002). That is, it minimizes the change in features while maximizing the variance of the samples. The intuition behind SFA is linked to the assumption that the information contained in a signal does not change suddenly, but slowly, in sequential data. Note, an input signal generally contains high variation, often caused by noise. Nonetheless, it is the seldom varying features of the sequence that mark the separation between informative changes. SFA extracts these features, as it selects the important attributes of the video which change least when time moves forward.

Inspired by the slowness principle of neural networks presented by Földiák (1991), Franzius et al. (2008) employ SFA for invariant object localization and recognition. This work extracts slow features hierarchically. In particular, multiple areas of the input images are analyzed for slow features individually. These features are then combined and reused.
as the input signal for the next higher level of the hierarchy. In total, four SFA layers are introduced (figure 2.4). The final layer’s output features are then fed through linear regression or classification to find the system’s prediction of the target.

The recognition of human actions by SFA is proposed by Zhang and Tao (2012). Here temporal image patches are extracted from areas of motion in the videos. These patches describe the change of pixel intensities over a time window, thus resulting in a number of short video clips for SFA learning. The patches of each action in the training data are combined, and the common slow features extracted. In the recognition phase, the model most similar to the tested video provides the system’s classification.

Nater et al. (2011) exploit SFA’s properties to segment video data temporally (figure 2.5). The aim of their study is to extract unknown different actions in an image sequence. The individual segments are thought to be the activities in the examined video. After performing SFA on the complete video, the authors determine whether a split of the sequence is required. The decision is based on the median of change in the slow features. For the separation, the frame with the largest change is utilized as split position, as SFA tries to extract components that change least over time. SFA is once more performed on the resulting videos, and the process is repeated until no further split is necessary.
Another example of SFA applied to temporal segmentation is presented by Zafeiriou et al. (2013). After formulating SFA as probabilistic model and solving for expectation maximization, the authors are able to extract the temporal phases of facial expressions through SFA. Their offline algorithm successfully extracts onset (beginning), apex (duration) and offset (ending) of facial action units.

Non-Linear Kernel Slow Feature Analysis

KSFA is originally proposed by Böhmer et al. (2011) in the speech processing domain. Like in KPCA, the samples are mapped into a Hilbert space which may be implicit, as the kernel trick can be applied. However, Böhmer et al. (2011) apply standard off-the-shelf kernels and thus their system remains non-robust.

2.1.3 Generalized Eigenvalue Problem

There are many dimensionality reduction techniques, and most of them are beyond the scope of this thesis. A large proportion, however, can formulated as generalized eigenvalue problems (de la Torre 2012; Kimura et al. 2013; Kokiopoulou et al. 2011; Sun et al. 2009; Zhang et al. 2009).

Optimization Problem

The generalized eigenvalue decomposition is often expressed as

\[ A u_r = \lambda_r B u_r, \quad (2.19) \]

where \( \{u_r\}_{r=1}^{R} \) represents the set of eigenvectors and \( \{\lambda_r\}_{r=1}^{R} \) denotes the set of eigenvalues. If matrices \( A \) and \( B \) are square and regular, eq. (2.19) can be formulated as a trace problem, given by

\[
U = \arg \max_{\tilde{U}} \left( \text{tr} \left( (\tilde{U}^T B \tilde{U})^{-1} \tilde{U}^T A \tilde{U} \right) \right) \\
= \arg \min_{\tilde{U}} \left( \text{tr} \left( (\tilde{U}^T A \tilde{U})^{-1} \tilde{U}^T B \tilde{U} \right) \right),
\quad (2.20)
\]

where \( U \) is the projection onto subspace. In real data, however, matrices \( A \) and \( B \) are seldom regular. Then the pseudo-inverse applied to eq. (2.20) provides an alternative approximation to eq. (2.19).

Table 2.1 lists some common setting of matrices \( A \) and \( B \) in literature. PCA maximizes the variance of samples via the scatter matrix \( XX^T \). SFA optimizes the projection in
<table>
<thead>
<tr>
<th>PCA</th>
<th>SFA</th>
<th>LDA</th>
<th>PLS</th>
<th>CCA</th>
<th>MLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix $A$</td>
<td>$XX^T$</td>
<td>$XX^T$</td>
<td>$S^{(b)}$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$yy^T$</td>
<td>$YY^T$</td>
<td>$YY^T$</td>
</tr>
<tr>
<td>Matrix $B$</td>
<td>$I$</td>
<td>$S$</td>
<td>$S^{(w)}$</td>
<td>$I$</td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$0$</td>
<td>$XX^T$</td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$0$</td>
<td>$0$</td>
<td>$I$</td>
</tr>
</tbody>
</table>

Table 2.1: A summary of generalized eigenvalue problems. Matrix $X$ contains the sample data, and matrix $Y$ denote the relevant labels. The scatter matrix of the data derivative is given by $S$. $S^{(b)}$ and $S^{(w)}$ represent the between and within-class scatter matrix respectively.

relation to $XX^T$ and the scatter matrix of the data derivative $S$. LDA inflates the between-class variance while reducing the within-class differences – here, labeled training data is required. A complete set of labels $\{y_n\}_{n=1}^N$ and the matrix $Y = \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix}$ are also required for the remaining three algorithms, which are regression based. Partial least squares (PLS) finds a subspace that best explains the largest variance in the predictions in $Y$. Canonical correlation analysis (CCA) correlates the linear relationships between two sample sets. Multiple linear regression (MLR) minimizes the distance between $Y$ and the projected samples in $X$. Detailed summaries of eigenvalue problems are presented by de la Torre (2012); Kimura et al. (2013); Kokiopoulou et al. (2011); Sun et al. (2009); Zhang et al. (2009).

Non-Linear Kernel Methods

Like for PCA and SFA, kernelized versions are presented for the algorithms above in the literature. For example, kernel LDA is employed by Mika et al. (1999) and Yang et al. (2005b), while Lai and Fyfe (2000) introduce kernel CCA.

### 2.2 Online Learning

With an ever increasing importance of realizing online and real-time applications, incremental learning methods have become a popular research topic. In such framework, the system requires only minimal training a priori (often only an initialization), as the hypothesis is updated when new data points arrive. Hence the system is adaptive to the samples in the test data, as it incorporates such points into its hypothesis. It is important to note here, that the term ‘online’ has different meanings in the literature. Throughout my thesis, I refer to incremental, real-time setups as online systems.

Many online learning methods have been recently proposed. For example, He et al. (2011a) introduce a general online algorithm to learn from stream data. A framework for learning with non-stationary environments, for which the classes and data distribution
change over time, is presented by Elwell and Polikar (2011). Motivated by the cerebral cortex, Weng and Hwang (2007) propose an online methodology for classification and regression. Online blind source separation is introduced by Zhou et al. (2011), and several variations of incremental PCA have been recently introduced as well (Levy and Lindenbaum 2000; Ozawa et al. 2008; Ross et al. 2008; Weng et al. 2003).

In this thesis, I particularly focus on online learning with subspaces, and incremental kernel learning. Existing research related to these topics, is discussed in the following sections.

### 2.2.1 Incremental Eigenvalue Analysis

Any eigenvalue problem commonly requires the complete training data before learning can be done. This is often unfeasible, especially in adaptive real-time applications, where stream data is used as input. In the last two decades, incremental PCA has been a developing research area (Chandrasekaran et al. 1997; Hall et al. 1998; Levy and Lindenbaum 2000; Ozawa et al. 2008; Ross et al. 2008; Weng et al. 2003; Zhao et al. 2006).

The early methods by Chandrasekaran et al. (1997) and Hall et al. (1998) update the eigenvalue decomposition via an update of the covariance matrix. This, however, is unfeasible for high-dimensional image data. More recently, Weng et al. (2003) introduce candid covariance-free incremental PCA (CCIPCA) which is based on statistical efficiency, and incrementally estimates the data distribution by means of scale and mean. CCIPCA does not require the covariance matrix (or scatter matrix) of the samples and hence is suitable for image data. Nonetheless, the incremental convergence procedure requires a larger sample-to-dimension ratio for accurate results (Zhao et al. 2006).

A mathematically exact incremental update of PCA is introduced by Levy and Lindenbaum (2000), and improved by Ross et al. (2008) to allow for mean-substraction. Here the sample set \( \{x_n\}_{n=1}^N \) is approximated by a sufficient spanning set via eigenvalue decomposition. Hence, if all eigenvalues are retained, the subspace is exactly equivalent to its offline version. Nonetheless, very small approximation errors are achieved by only considering the largest eigenvalues. In particular, given the SVD of the sample matrix at time \( t - 1 \), i.e.

\[
X_{t-1} = \begin{bmatrix} x_1 & \cdots & x_{t-1} \end{bmatrix} \approx U_{t-1} \Sigma_{t-1} V_{t-1}^T,
\]

(2.21)

the new sample matrix at time \( t \) can be approximated by

\[
X_t = \begin{bmatrix} X_{t} & X_\delta \end{bmatrix} \approx \begin{bmatrix} U_{t-1} \Sigma_{t-1} V_{t-1}^T \ X_\delta \end{bmatrix},
\]

(2.22)
where $X_\delta = x_t$ represents the novel data point. Notice, eq. (2.22) can be reformulated as

$$
X_t \approx \left[ \begin{array}{ccc} U_{t-1} & U_\delta \\ \Sigma_{t-1} & 0 \\ 0 & U_\delta^T \Sigma \end{array} \right] \left[ \begin{array}{ccc} V_{t-1} & 0 \\ 0 & \mathbf{I} \end{array} \right]^T,
$$

(2.23)

where $U_\delta = \text{orth}(X_\delta - UU^TX_\delta)$ contains the components not included in subspace $U_{t-1}$ (here, orth(,) finds the orthogonal components, and 0 implies zero padding). The SVD of the central term then yields the solution, i.e.

$$
L = \left[ \begin{array}{ccc} \Sigma_{t-1} & U_{t-1}^TX_\delta \\ 0 & U_\delta^TX_\delta \end{array} \right] = \hat{U}\hat{\Sigma}\hat{V}^T,
$$

(2.24)

as $U_t = \left[ \begin{array}{ccc} U_{t-1} & U_\delta \end{array} \right] \hat{U}$ and $\Sigma_t = \hat{\Sigma}$. The orthonormal matrix $V_t$ is not required for the computation.

Other researchers target the generalized eigenvalue problem. For example, Kim et al. (2007) introduce incremental LDA by approximating the within-class scatter matrix and the total scatter matrix through sufficient spanning sets. Incremental SFA is proposed by Kompella et al. (2012). Here, CCIPCA is combined with an incremental minor component analysis (MCA) algorithm to solve SFA online.

However, I emphasize, the presented algorithms above all rely on the $\ell_2$-norm and thus fail to be robust. Furthermore, the robust reformulations of PCA, reviewed above, have no known incremental implementation and are therefore computationally expensive for large training sets, and unsuitable for online learning.

### 2.2.2 Online Learning with Kernels

In contrast to linear online subspace learning, incremental learning with non-linear setups is far from being solved. Nonetheless, many systems successfully apply online kernel learning to acquire non-linearity (Crammer et al. 2006; Engel et al. 2004; Freund and Schapire 1999; Liu et al. 2009; Orabona et al. 2009; Smale and Yao 2006; Yao 2010). Prominent methods range from classification (Bordes et al. 2005; Crammer and Singer 2003; Crammer et al. 2004; Slavakis et al. 2008; Ying and Zhou 2006) and regression (Kivinen et al. 2004; Richard et al. 2009) to novelty and change detection (Desobry et al. 2005; Kivinen et al. 2004), subspace learning (Günter et al. 2007; Kim et al. 2005), and feature extraction (Chin and Suter 2007; Laskov et al. 2006).

As seen in eq. (2.16), the online classification or regression function is typically written as a weighted sum of kernel combinations of samples from a set of stored instances, referred to as support or reduced set. One of the major challenges is that at each time step a new
instance is fed to the algorithm and added to this support set, which may grow to become arbitrarily large over time. To somewhat overcome the impact of growth, different update criterions are presented in literature.

Kivinen et al. (2004) propose online kernel algorithms for classification, regression and novelty detection using a stochastic gradient descent algorithm in the Hilbert space defined by the kernel. In order to avoid the arbitrary growth of the support set, the authors adopt simple truncation and shrinking strategies. Engel et al. (2004) present an online kernel regression algorithm that is based on constructing and solving minimum mean-squared-error optimization problems with Mercer kernels. In order to regularize solutions and keep the complexity of the algorithm bounded, a sequential sparsification process is adopted. This sparsification process has close connection to incremental PCA. Orabona et al. (2009) introduce the so-called Projectron algorithm which neither truncates nor discards instances. In order to keep the support set bounded, the algorithm projects the samples onto the space spanned by the support set. If this is impossible without greater loss, the samples are added to the support set. It is proven that by following the Projectron algorithm, the support set and, therefore, the online hypothesis, is guaranteed to converge. The drawback is that the size of support cannot be predicted in advance. To cope with this, a parameter which provides a trade-off between accuracy and sparsity, is introduced. Richard et al. (2009) propose an online regression algorithm which uses an alternative model reduction criterion. Instead of using a sparsification procedure, the increase in the number of variables is controlled by a coherence parameter, a fundamental quantity that characterizes the behavior of dictionaries in sparse approximation problems.

Although many methods have been proposed for online kernel learning for classification and regression, limited research has been conducted for online subspace learning with kernels. This research has mainly revolved around the development of incremental KPCA (Chin and Suter 2007; Günter et al. 2007; Kim et al. 2005) and kernel SVD (Chin et al. 2006) algorithms. One of the first incremental KPCA algorithms is proposed by Kim et al. (2005). This algorithm is essentially the kernelization of the generalized Hebbian algorithm, which has similar operational characteristics to a single-layer feed-forward neural network. Gain adaptation methods which improve convergence of the kernel Hebbian algorithm are introduced by Günter et al. (2007). An incremental KPCA algorithm which kernelizes the mathematically exact algorithm for incremental PCA by Levy and Lindemanbaum (2000); Ross et al. (2008) is presented by Chin and Suter (2007). In this method, in order to maintain constant update speed the authors construct reduced set expansions of the kernel principal components and the mean, by means of preimages. While such setup nicely bounds the support set, the main drawbacks of this approach are that (i) the reduced set representation provides only an approximation to the exact solution, and
(ii) the proposed optimization problem for finding the expansion inevitably increases the complexity of the algorithm.

In this section, I conclude that it still remains unclear how to combine robustness with efficiency, as most incremental kernel methods require expensive and inaccurate update strategies to prevent arbitrarily large growth in the support set.

2.3 Incremental Learning for Object Tracking

A very popular application is the evaluation of incremental learning using visual tracking (Babenko et al. 2011; Chin and Suter 2007; Ross et al. 2008). The aim of a visual tracking system is to locate a predefined target object on every frame of a video sequence. Automatic tracking systems span a wide range of applications, such as traffic monitoring (Hsieh et al. 2006; Kamijo et al. 2000), surveillance (Collins et al. 2001; Haritaoglu et al. 2000), video retrieval and summarization (Luo et al. 2003), vehicle navigation (Fraundorfer et al. 2007; Hashima et al. 1997), driver assistance (Avidan 2004; Handmann et al. 1998), human-computer interaction (Wren et al. 1997) and face analysis (Cohn et al. 1999; Gunes and Pantic 2010).

Although trackers have received plentiful attention, common problems persist. As outlined in chapter 1, complex motion, changes in illumination, occlusions and non-rigid appearance variations are some examples which make most trackers fail (Yilmaz et al. 2006). Consequently, assumptions about the data, such as static background or constant illumination, are often made to facilitate tracking. However, it is difficult to satisfy such assumptions in real data, as the aforementioned events often occur in real life settings. Few approaches address real-time tracking in realistic environments and unconstrained scenes. However, recent literature shows that adapting to the scene via online learning is advantageous in many tracking frameworks (Babenko et al. 2011; Mei and Ling 2009; Ross et al. 2008).

In tracking, some systems track the silhouette or the contour of the target (Isard and Blake 1998; Li et al. 2001; Paragios and Deriche 2005; Yilmaz et al. 2004). Other methods incorporate articulated shape information and structural features of the target (Cootes et al. 2001; Gall et al. 2009). I am particularly interested in systems which track an object as an image area (Babenko et al. 2011; Chin and Suter 2007; Mei and Ling 2009; Ross et al. 2008; Zhou et al. 2004). Usually, the search strategy of such tracking system is based on three main components

1. **Image representation**: The image representation defines the low-level features which are extracted from the frames of the video sequence.
2. Appearance model: The appearance model is usually a statistical model of the target.

3. Motion model: The motion model describes the set of parameters that define the motion of the target and its dynamics.

In the following I provide a short overview for each of these components. Finally, I present a recent example by Ross et al. (2008), which applies online subspace learning to tracking.

### 2.3.1 Image Representation

Landmark-based (also known as feature-based) algorithms represent the image using features, such as corners and edge joints (Cox and Hingorani 1996; Shapiro et al. 1992). Points of interest are detected throughout the video sequence and utilized as image features. Region-based (also referred to as appearance-based) methods use the information from the appearance directly. Popular features for appearance-based algorithms include (i) raw pixel intensities (Chin and Suter 2007; Ross et al. 2008; Zhou et al. 2004), (ii) color (Mei and Ling 2009) and (iii) gradients (Birchfield 1998; Buenaposada and Baumela 2002; Comaniciu et al. 2003; Hager and Belhumeur 1998). Other methods employ histograms of oriented gradients (Dalal and Triggs 2005), color histograms (Comaniciu et al. 2003), haar-like features (Viola and Jones 2001) or local binary patterns (Ojala et al. 2002).

### 2.3.2 Appearance Model

The appearance model is often learned a priori for the tracking task (Black and Jepson 1998; Cootes et al. 2001; Hager and Belhumeur 1998). Given sufficient training data, such methods follow the tracked object successfully. Nonetheless, unseen appearances of the target are not supported, and therefore appearance changes pose difficulties. Recent methods learn the appearance model online, either generatively (Chin and Suter 2007; Matthews et al. 2004; Mei and Ling 2009; Ross et al. 2008), discriminatively (Babenko et al. 2011; Collins et al. 2005; Ellis et al. 2011; Grabner et al. 2006, 2008; Hare et al. 2011; Kalal et al. 2012; Liu and Yu 2007) or as a combination of these (Woodley et al. 2007). This allows for online adaptation of the model, and therefore improves the support for appearance changes.

In my work, I focus on the online learning of non-linear subspace-based generative approaches to appearance-based tracking. Linear subspace analysis for visual tracking dates back to the EigenTracking by Black and Jepson (1998), the efficient region tracker by Hager and Belhumeur (1998), and active appearance models (AAMs) by Cootes et al. (2001). Over the years, several extensions to these methods have been proposed, addressing issues related to efficiency and robustness. For example, Ross et al. (2008) adopt the
incremental PCA by Levy and Lindenbaum (2000) for visual tracking, and a non-linear extension of this is described by Chin and Suter (2007). Matthews et al. (2004) propose a person-specific online version of AAMs. However, as typically encountered in online learning, the problem of efficiency versus robustness is also prevalent in adaptive visual tracking. For example, the fast incremental PCA by Ross et al. (2008) is not robust, while robust methods lack efficiency (de la Torre and Black 2003).

### 2.3.3 Motion Model

The third component is the employed motion model and the underlying dynamics, which defines the object’s transformation between consecutive frames. The transformation is often described as (i) simple translation (Babenko et al. 2009), (ii) translation and scale (Babenko et al. 2011; Zhou et al. 2004), (iii) affine transformation (Chin and Suter 2007; Mei and Ling 2009; Ross et al. 2008) or (iv) projective transformation (Buenaposada and Baumela 2002). Here, more dimensions of freedom make the tracking results potentially more accurate, but at a cost of computational complexity.

The Kalman filter is a popular choice for modeling the dynamics of linear motion (Broida and Chellappa 1986; Metaxas and Terzopoulos 1993; Reynard et al. 1996; Rosales and Sclaroff 1999). Here, the target’s new location is estimated based on previously observed motion patterns. However, the target’s motion is not always linear. Non-linear motion is supported by the mean shift algorithm. In this method, histograms of features are computed for many candidate areas, and the location with the highest density of probability is picked as the target’s position (Collins 2003; Comaniciu et al. 2000, 2003; Vojir et al. 2013). However, if the appearance model is not based on histograms, mean shift is not applicable. Another option, and one of the most prominent models, is a particle filter (Isard and Blake 1998; Nikitidis et al. 2008; Patras and Pantic 2004; Yang et al. 2005a; Zhou et al. 2004). A particle filter estimates the posterior distribution of a system’s state recursively, using sequential Monte Carlo methods. The best particle as defined by the observation model (i.e. the appearance model) is selected to represent the new state of the system. While the estimation of the new state is often solely based on the previous state, some methods utilize all particles of the previous time step, weighted by their probability, for the Monte Carlo step (Ross et al. 2008). Details are provided in the following section, where I describe an example of adaptive tracking via online learning.

### 2.3.4 Incremental PCA for Tracking

Ross et al. (2008) describe an incremental PCA learning system, which is applied to tracking with an adaptive appearance model. In their framework, the features are vectors
of pixel intensities, and a particle filter with affine motion is used. The appearance model is learned as an incremental subspace.

Figure 2.6 illustrates the workings. Given the initial location $A_1^{(0)}$ at time $t = 0$, a particle is extracted to form a 32 × 32 image of the target. The image is converted into a lexicographically arranged vector $x_1^{(0)} \in \mathbb{R}^{1024}$, and the PCA model is initiated. Then, the particle filter applies a Gaussian to the initial location $A_1^{(0)}$, and picks 500 particles $\{A_m^{(1)}\}_{m=1}^{500}$ from the distribution

$$p(A_m^{(1)}|A_1^{(0)}) = \mathcal{N}(A_1^{(0)}, \Xi)|_{A_m^{(1)}},$$

(2.25)

where $\Xi$ is an independent covariance matrix, which represents the variance of the affine transformation, i.e. the horizontal and vertical displacement, in-plane rotation, scale, ratio and skew.

The corresponding vector representations $\{x_m^{(t)}\}_{m=1}^{500}$ of locations $\{A_m^{(t)}\}_{m=1}^{500}$ are extracted from the video frame at time-step $t = 1$. Their quality is then measured via the reconstruction distance to the subspace, given by

$$p(x_m^{(t)}) \propto e^{-\gamma\|x_m^{(t)} - U_tU_t^T x_m^{(t)}\|^2_F},$$

(2.26)

where $U_t$ represents the PCA’s subspace at time $t$, and $\gamma$ is the parameter that controls the spread.

The location related to the best quality $A_b^{(t)}$ is used to provide the estimated position

Chapter 2. Background
of the target, and the sample $x_k^{(t)}$ is employed to update the PCA model. Finally, the locations are diffused for the next frame, via a Gaussian mixture model (GMM), given by

$$p(A_m^{(t+1)}|\{A_m^{(t)}\}_{m=1}^{500}) = \sum_{m'=1}^{500} w_{m'}^{(t)} N(A_m^{(t)}, \Xi)|_{A_m^{(t+1)}},$$  \hspace{1cm} (2.27)$$

where the weights $\{w_{m'}^{(t)}\}_{m=1}^{500}$ are provided by the scaled probabilities in relation to eq. (2.26), such that $\sum_{m=1}^{500} w_{m'}^{(t)} = 1$. The process is now repeated for the rest of the sequence.

In the following chapters, I expand this framework to develop online learning systems with special kernel functions, which introduce robustness without loss of efficiency.
Principal Component Analysis (PCA) is perhaps the most prominent learning tool for dimensionality reduction in pattern recognition and computer vision. However, as outlined in chapter 2, the $\ell_2$-norm employed by standard PCA is not robust to outliers. A common alleviation to this is the introduction of alternative dissimilarity measures (Candès et al. 2009; Chin et al. 2006; de la Torre and Black 2003; Ding et al. 2006; He et al. 2011b; Ke and Kanade 2003, 2005; Kwak 2008). Methods in the literature, however, require computationally slow optimization algorithms to facilitate such setups.

In this chapter, a kernel PCA method for fast and robust PCA is introduced, called Euler PCA (e-PCA). The method is based on a dissimilarity measure which is originally proposed by Fitch et al. (2005) in the context of robust correlation-based estimation of large translational displacements. In e-PCA, pixel intensities are first normalized, and then converted into an Euler representation of complex numbers with unit norm. Then, the standard $\ell_2$-norm is applied. Overall, this is equivalent to applying a dissimilarity measure which is given by the cosine of the pixel differences as proposed by Fitch et al. (2005). Note, the mapping is explicit and thus the proposed kernel PCA is closely related to standard PCA and retains all the favorable properties (e.g. efficiency and rotational invariance). Furthermore, it offers a very efficient approximation of preimages without solving a separate optimization problem.

Additionally, due to the existence of an explicit mapping to feature space, e-PCA allows for an exact and efficient incremental implementation. Incremental PCA is known to be more efficient than batch PCA when applied to large training sets (Li 2004; Ross et al. 2008). Furthermore, incremental PCA allows for online learning. Overall, the proposed e-PCA forms a fast, direct and robust alternative to standard PCA. I summarize the favorable properties of the proposed e-PCA:

1. Contrary to the state-of-the-art linear robust PCA and kernel PCA approaches (Candès et al. 2009; Chin et al. 2006; de la Torre and Black 2003; Ding et al. 2006;

2. Contrary to kernel PCA approaches with standard kernels such as GRBFs, for e-PCA there exists an efficient way for preimage computation without solving an expensive optimization problem.

Tzimiropoulos et al. (2012a) present a similar learning framework to this proposed in the current chapter. In their work, rather than representing the image intensity as rotation via the Euler representation of complex numbers with unit norm, the gradient orientation values are directly employed. For this setup, however, a special property, only valid in the gradient domain, is exploited (Tzimiropoulos et al. 2010). As my work is a learning method directly derived from the correlation method of Fitch et al. (2005), e-PCA neither relies on the statistic of the gradient orientation differences, nor restricts itself to the domain of gradient orientations in general.

In section 3.1, I introduce the cosine-based dissimilarity measure that forms the basis of the proposed e-PCA. In section 3.2, I formulate e-PCA and discuss some of its properties. Experimental results are presented in section 3.3. Finally, conclusions are drawn in section 3.4.

### 3.1 Cosine-based Dissimilarity

Let us consider a $P$-dimensional vector, $z$, obtained by writing the pixel values of an image in lexicographic ordering. Motivated by the work of Fitch et al. (2005) on robust correlation-based translation estimation, the $\ell_2$-norm between two such samples $z_i$ and $z_j$ is replaced by the following dissimilarity measure

$$s(z_i, z_j) = \sum_{p=1}^{P} \left\{ 1 - \cos(\alpha \pi (z_i(p) - z_j(p))) \right\},$$

where the pixel values are represented in the range $[0, 1]$ and $\alpha \in \mathbb{R}_+$ (i.e. $\alpha$ is a positive, real number). Figure 3.1 visualizes the dissimilarity function with changing values for $\alpha$. Under consideration of the Taylor series for cosine, given by

$$\cos(x) = 1 - \frac{x^2}{2!} + \mathcal{O}(x^4)$$

where $\mathcal{O}(x^4)$ is an error of $x^4$ complexity, it is clear that an $\alpha$ value towards zero in eq. (3.1) results in a function which is proportional to the $\ell_2$-norm. With increasing $\alpha$, the effect
Figure 3.1: Cosine-based dissimilarity measure with varying \( \alpha \). The distances are normalized for illustration purposes.

of large distances, possibly caused by outliers, is reduced. In general, \( \alpha \) represents the frequency of the cosine and is optimized to suppress the values caused by outliers.

As noted by Fitch et al. (2005), for pixel intensities in the range \([0,1]\), eq. (3.1) is equivalent to Andrews’ M-Estimator. In particular, the influence function of the kernel (i.e. the derivative of the kernel) is equivalent to Andrews’ influence function, which is given by

\[
\psi(x) = \begin{cases} 
\sin(\pi x) & \text{if } -1 \leq x \leq 1 \\
0 & \text{otherwise}
\end{cases}
\]

The fast robust correlation (FRC) scheme of Fitch et al. (2005) employs eq. (3.1) and, unlike \( \ell_2 \)-based correlation, is able to estimate large translational displacements in real images while achieving the same computational complexity.

Prior to formulating the proposed PCA, let us consider a motivating example in which different dissimilarity measures are applied to the images shown in figure 3.2. As can be seen in table 3.1, the \( \ell_2 \)-norm associates a smaller distance between the original image and an image from a different subject. The distance between the original and the same
Figure 3.2: Motivating example for the use of the cosine-based dissimilarity measure. Shown from left to right are the original image, a second image of the same subject, an occluded version of the original image and an image of another subject.

<table>
<thead>
<tr>
<th></th>
<th>$\ell_2$-Norm</th>
<th>Cosine Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same Subject</td>
<td>$9.79 \times 10^{-4}$</td>
<td>$1.34 \times 10^{-2}$</td>
</tr>
<tr>
<td>Occluded</td>
<td>$1.96 \times 10^{-3}$</td>
<td>$3.36 \times 10^{-2}$</td>
</tr>
<tr>
<td>Different Subject</td>
<td>$1.63 \times 10^{-3}$</td>
<td>$3.46 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of normalized dissimilarity measures. The $\ell_2$-norm is more affected by random corruptions and thus reduces its score more drastically with random pixel noise. For the cosine measure with $\alpha = 1.9$, the occlusion is closer than an image of a random different subject.

image with occlusion is larger. In contrast, the use of the cosine-based measure results in a large distance between the original image and the image of a different person.\(^1\)

### 3.2 Euler PCA ($e$-PCA)

I first present $e$-PCA and introduce its representation as both a kernel PCA and a linear PCA with a special feature mapping. Then, inspired by the feature mapping, I develop a fast computation of preimages, which I validate through the kernel interpretation of $e$-PCA. Finally, building on the direct feature map, I propose a fast incremental update for online learning.

#### 3.2.1 Defining Euler PCA

Euler PCA is a kernel PCA which utilizes the robust dissimilarity in eq. (3.1). Moreover, it is also based on the Euler representation of complex numbers. Specifically, the normalized intensity values in sample $z$ (in range $[0, 1]$) are mapped onto the complex representation

\(^1\)After setting $\alpha = 1.9$, as will be discussed later, in section 3.3.
Figure 3.3: Visualization of the explicit mapping with different $\alpha$. A small value for $\alpha$ (a) leads to a near linear distribution. Medium $\alpha$ values (b) represent distances that fulfill a more circular distribution. Large $\alpha$ values (c) rate outliers less aggressively.

$x \in \mathbb{C}^P$, where

$$x = \begin{bmatrix} e^{i\alpha \pi z(1)} \\ \vdots \\ e^{i\alpha \pi z(P)} \end{bmatrix} = e^{i\alpha \pi z}.$$  

(3.4)

The value $x$ can be thought of as the special features in a version of PCA. Figure 3.3 visualizes the mapping onto the circular space for different values of $\alpha$. To show the relationship between eq. (3.4) and eq. (3.1), let us define the shorthand $\theta \triangleq \alpha \pi z$ and then apply the $\ell_2$-norm between two samples $x_i, x_j \in \mathbb{C}^P$

$$\|x_i - x_j\|_F^2 = \|(\cos (\alpha \pi z_i) + i \sin (\alpha \pi z_i)) - (\cos (\alpha \pi z_j) + i \sin (\alpha \pi z_j))\|_F^2$$

$$= \sum_{p=1}^P \{2 - 2 (\cos (\theta_i(p)) \cos (\theta_j(p)) + \sin (\theta_i(p)) \sin (\theta_j(p)))\}$$

$$= 2 \sum_{p=1}^P \{1 - \cos (\theta_i(p) - \theta_j(p))\}$$

$$\propto s(z_i, z_j).$$  

(3.5)

Eq. (3.5) suggests that a KPCA can be defined by first applying the explicit mapping of eq. (3.4) and then using standard linear PCA with complex numbers. Because of eq. (3.4), this approach is coined Euler PCA ($e$-PCA). Notice that for $\alpha < 2$, the mapping is one-to-one. In this case, preimages may be computed fast via the $\angle$-operator, which returns the angle of a complex number. More specifically, after reconstruction (or de-noising) in the feature space has been performed, we can go back to the pixel domain using the $\angle$-operator. Finally, high-dimensional data can be readily handled by using theorem 1.

**Theorem 1.** Define matrices $A$ and $B$ such that $A = XX^H$ and $B = X^HX$ are the scatter and kernel matrix of the samples in $X$, where $(\cdot)^H$ computes the complex conjugate transposition of a matrix. Let $U_A$ and $U_B$ be the eigenvectors corresponding to the non-
Algorithm 3.1 ESTIMATING THE PRINCIPAL SUBSPACE

**Input:** A set of $N$ images $\{I_n\}_{n=1}^N$ with $P$ pixels in each image, the number $R$ of principal components and parameter $\alpha$.

**Output:** The principal subspace $U$ and eigenvalues $\Sigma$ of the samples.

1. Represent the pixel values of $I_n$ in the range of $[0, 1]$ and obtain $z_n$ by writing $I_n$ in lexicographic ordering.
2. Compute $x_n$ using eq. (3.4) and form the matrix of the transformed data $X = [x_1 \cdots x_N] \in \mathbb{C}^{P \times N}$.
3. Compute the kernel matrix $K = X^H X \in \mathbb{C}^{N \times N}$ and find the eigenvalue decomposition of $K = \Omega \Lambda \Omega^H$.
4. Find the $R$-reduced set, $\Omega_R \in \mathbb{C}^{N \times R}$ and $\Lambda_R \in \mathbb{R}^{R \times R}$ with largest eigenvalues.
5. Compute $U = X \Omega_R \Lambda_R^{-\frac{1}{2}} \in \mathbb{C}^{P \times R}$ and $\Sigma = \Lambda_R^\frac{1}{2}$.
6. Reconstruct using $\hat{X} = UU^H X$.
7. Using the fast preimage computation, go back to the pixel domain via $\hat{Z} = \hat{X}^\alpha\pi$.

Algorithm 3.2 EMBEDDING OF NEW SAMPLES

**Input:** The principal subspace $U$ and $\alpha$ of algorithm 3.1, as well as a new image $I$ composed of $P$ pixels.

**Output:** The sample’s embedding in subspace and pixel domain, $\hat{x}$ and $\hat{z}$ respectively.

1. Represent the pixel values of $I$ in the range of $[0, 1]$ and obtain $z$ by writing $I$ in lexicographic ordering.
2. Find $x$ using eq. (3.4).
3. Reconstruct using $\hat{x} = UU^H x$.
4. Using the fast preimage computation, go back to the pixel domain via $\hat{Z} = \hat{X}^\alpha\pi$.

The complete computation of $e$-PCA is given in algorithm 3.1 and algorithm 3.2. Notice that in algorithm 3.1, $e$-PCA is formulated as a kernel method, because in computer
vision we mainly deal with small-sample-size problems and high-dimensional data (images). In general, however, the computation of the principal subspace could be equally obtained from the eigenvalue decomposition of the scatter matrix $XX^H$. Additionally, note that although the mapping in $e$-PCA is explicit, the calculation of preimages in the last step of algorithm 3.1 based on the $\angle$-operator is only an approximation. To this point, it would be interesting to compare this approach with standard calculation of preimages in KPCA algorithms using gradient descent optimization.

Let us discuss $e$-PCA’s interpretation as a KPCA with a complex kernel in an explicitly defined complex Hilbert space. For this, let $k : \mathbb{R}^P \times \mathbb{R}^P \rightarrow \mathbb{C}$ be a positive definite function that defines a RKHS $\mathcal{H}$ (the so-called feature space) through an implicit (or as in the proposed case, explicit) mapping $\phi : \mathbb{R}^P \rightarrow \mathcal{H}$ such that the kernel describes the inner product $k(z_i, z_j) = \langle \phi(z_j), \phi(z_i) \rangle$. Notice, the inner product may be written as dot-product between the mapped samples $\phi(z_i)^H\phi(z_j)$, with the property $k(z_i, z_j) = k(z_j, z_i)$, where $(\cdot)^\dagger$ is the complex conjugate operator. Using the complex feature space interpretation, the kernel that corresponds to the mapping in eq. (3.4) can be written as

$$k(z_i, z_j) = \sum_{p=1}^{P} \left\{ \cos(\alpha \pi (z_i(p) - z_j(p))) - i \sin(\alpha \pi (z_i(p) - z_j(p))) \right\}.$$

(3.7)

3.2.2 Preimage Computation

With the KPCA interpretation of the proposed method, the strategy for reconstruction in the input space becomes less apparent. In this section, I present the reconstruction by means of preimage computation (Kwok and Tsang 2004; Mika et al. 1999). In particular, I put the suggested approximation $\hat{z} = \frac{\hat{\Delta}}{\alpha \pi}$ of algorithm 3.1 and algorithm 3.2 for going back to the pixel domain (input space) in the context of preimage computation. Finally, I derive a closed form to the approximation error.

Following Mika et al. (1999), to de-noise a sample $z$ in the kernel-induced Hilbert space, the following optimization problem is solved:

$$\hat{z} = \arg \min_{\tilde{z}} \left\| \phi(\tilde{z}) - UU^H\phi(z) \right\|_F^2$$

$$= \arg \min_{\tilde{z}} \left\{ k(\tilde{z}, \tilde{z}) + \phi(z)^HUU^HUU^H\phi(z) \right.$$  
$$\left. - \phi(\tilde{z})^HUU^H\phi(z) - \phi(z)^HUU^H\phi(\tilde{z}) \right\}.$$  

(3.8)

Notice that for the proposed kernel in eq. (3.7), $k(z, z) = 1$ for any sample $z$. Therefore,
eq. (3.8) can be further simplified to

\[
\hat{z} = \arg \min_{\tilde{z}} \left\{ -\phi(\tilde{z})^H U U^H \phi(z) - \phi(z)^H U U^H \phi(\tilde{z}) \right\} \\
= \arg \max_{\tilde{z}} \Re \left( \phi(\tilde{z})^H U U^H \phi(z) \right),
\]

(3.9)

where \( \Re(.) \) extracts the real part of a complex number. Note, in KPCA the projection matrix is represented as a linear combination \( U = X \tilde{U} \), where \( X = \left[ \phi(z_1) \cdots \phi(z_N) \right] \) represents the (usually implicit) sample matrix of the mapped data in Hilbert space.

Let us define \( \text{Im}(.) \) as the operator that extracts the imaginary part of a complex number as a real number. Furthermore, let us represent the explicit reconstructed part of sample \( z \) in feature space by

\[
\tilde{x} = \tilde{U} \tilde{U}^H \begin{bmatrix} k(z_1, z) \\
\vdots \\
k(z_N, z) \end{bmatrix}.
\]

(3.10)

Then the optimization problem in eq. (3.9) can be reformulated as

\[
\hat{z} = \arg \max_{\tilde{z}} \Re \left( \begin{bmatrix} k(\tilde{z}, z_1) & \cdots & k(\tilde{z}, z_N) \end{bmatrix} \tilde{x} \right) \\
= \arg \max_{\tilde{z}} \left\{ \Re \left( \begin{bmatrix} k(\tilde{z}, z_1) & \cdots & k(\tilde{z}, z_N) \end{bmatrix} \right) \Re(\tilde{x}) \\
- \Im \left( \begin{bmatrix} k(\tilde{z}, z_1) & \cdots & k(\tilde{z}, z_N) \end{bmatrix} \right) \Im(\tilde{x}) \right\} \\
= \arg \max_{\tilde{z}} \sum_{n=1}^{N} \sum_{p=1}^{P} \{ \cos (\alpha \pi (\tilde{z}(p) - z_n(p))) \Re(\tilde{x}(n)) \\
+ \sin (\alpha \pi (\tilde{z}(p) - z_n(p))) \Im(\tilde{x}(n)) \} \\
= \arg \max_{\tilde{z}} f(\tilde{z}).
\]

(3.11)

The standard method of solving eq. (3.11) is by gradient ascent. In particular, an update of the form \( \tilde{z}_{t+1} = \tilde{z}_t + \nabla f(\tilde{z}_t) \) is required, where \( t \) corresponds to the iteration number (Mika et al. 1999). Hence, the partial derivatives \( \frac{\partial f}{\partial \tilde{z}(p)} \) are needed for all \( P \) pixels in an image. For any pixel \( \tilde{z}(p) \) we can write

3.2. Euler PCA (\( e \)-PCA)
\[ \frac{\partial f}{\partial \mathbf{z}(p)} = -\alpha \pi \sum_{n=1}^{N} \text{Re}(\tilde{x}(n)) \sin(\alpha \pi (\mathbf{z}(p)) - \mathbf{z}_n(p)) \]
\[ + \alpha \pi \sum_{n=1}^{N} \text{Im}(\tilde{x}(n)) \cos(\alpha \pi (\mathbf{z}(p)) - \mathbf{z}_n(p)) \]  

(3.12)

and \( \nabla f \) can be concisely written as

\[
\nabla f(\mathbf{z}) = - \begin{bmatrix} \text{Im}(\tilde{x} \odot x_1) & \cdots & \text{Im}(\tilde{x} \odot x_N) \end{bmatrix} \begin{bmatrix} \text{Re}(\tilde{x}(1)) \\ \vdots \\ \text{Re}(\tilde{x}(N)) \end{bmatrix} \\
+ \begin{bmatrix} \text{Re}(\tilde{x} \odot x_1) & \cdots & \text{Re}(\tilde{x} \odot x_N) \end{bmatrix} \begin{bmatrix} \text{Im}(\tilde{x}(1)) \\ \vdots \\ \text{Im}(\tilde{x}(N)) \end{bmatrix} ,
\]

(3.13)

where \( \odot \) is the element-wise product between vectors, and \( \tilde{x} \) is the transform \( \phi(\mathbf{z}) \) as given by eq. (3.4).

Notice, finding preimages via gradient ascent can be computationally expensive for large datasets. For example, the time complexity for recovering \( Q \) preimages is \( \mathcal{O}(TQPN) \) where \( T \) is the number of iterations until convergence. In contrast, for algorithm 3.1 and algorithm 3.2, a very fast way for approximating preimages is proposed by means of the \( \angle \)-operator, as preimages are approximated by

\[
\hat{\mathbf{z}} = \frac{1}{\alpha \pi} \angle (UU^H x) .
\]

(3.14)

The reasoning behind this back projection is illustrated in figure 3.4. In particular, eq. (3.14) finds the nearest point to the reconstruction, which lies on the circumference of the Euler induced circle. In the context of preimage computation the error of this approximation can be analytically computed by substituting eq. (3.14) into eq. (3.4), and the result into eq. (3.8). Let \( R(.) \) be a function that finds the element-wise magnitudes of a vector and \( \mathbf{1} \) is a vector of 1s. We find

Chapter 3. Euler Principal Component Analysis
Figure 3.4: Illustration of the preimage computation. The original sample $x$ is projected by the subspace $U$. The reconstruction could lie anywhere in the circle. The preimage $\hat{z}$ is the closest sample (in its mapped version), which is found on the circumference of the Euler mapping.

\[
\|\phi(\hat{z}) - UU^H \phi(z)\|_F^2 = \|e^{i\angle(UU^H x)} - UU^H x\|_F^2
\]
\[
= \|1 \odot e^{i\angle(UU^H x)} - R(UU^H x) \odot e^{i\angle(UU^H x)}\|_F^2
\]
\[
= \|1 - R(UU^H x)\|_F^2.
\] (3.15)

Finally, due to the invertibility of the proposed mapping eq. (3.4) for $0 \leq \alpha < 2$, in the case of $U$ containing all eigenvectors that correspond to non-zero eigenvalues, the preimage approximation using eq. (3.14) is optimal for the training set (i.e. eq. (3.15) is equal to zero).

3.2.3 Incremental Learning

I now introduce a fast incremental updating mechanism for online learning which exploits the direct feature mapping in eq. (3.4). The update method for the incremental $e$-PCA is based on standard incremental PCA (Levy and Lindenbaum 2000; Ross et al. 2008) – notice, $e$-PCA is formulated as a linear PCA in a non-linear subspace, defined by an explicit mapping. Let us assume the subspace $U_{t-1}$ of time step $t - 1$ is given by the SVD of the mapped sample matrix $X_{t-1}$, i.e.

\[
X_{t-1} \approx U_{t-1} \Sigma_{t-1} V_{t-1}^H = \left[ X_{t-1} \Omega_R \Lambda_R^{-\frac{1}{2}} \right] \left[ \Lambda_R^{\frac{1}{2}} \right]^H, \] (3.16)
Algorithm 3.3 Incremental Subspace Estimation

Input: The principal subspace $\mathbf{U}_{t-1} \in \mathbb{C}^{p \times R}$, the corresponding eigenvalues $\Sigma_{t-1} \in \mathbb{R}^{R \times R}$, a set of new images $\{\mathbf{I}_n\}_{n=N_{t-1}+1}^{N_t}$, the number $R$ of principal components and parameter $\alpha$.

Output: The new subspace $\mathbf{U}_t$ and eigenvalues $\Sigma_t$.

1: From set $\{\mathbf{I}_n\}_{n=N_{t-1}+1}^{N_t}$ compute the matrix of the transformed data $\mathbf{X}_\delta = \begin{bmatrix} \mathbf{x}_{N_{t-1}+1} & \cdots & \mathbf{x}_{N_{t-1}+N_\delta} \end{bmatrix}$.
2: Compute $\mathbf{U}_\delta = \text{orth}(\mathbf{X}_\delta - \mathbf{U}_{t-1} \mathbf{U}_{t-1}^H \mathbf{X}_\delta)$.
3: Form $\mathbf{L} = \begin{bmatrix} \Sigma_{t-1} & \mathbf{U}_{t-1}^H \mathbf{X}_\delta \\ 0 & \mathbf{U}_\delta^H \mathbf{X}_\delta \end{bmatrix}$.
4: Find $\mathbf{L} = \mathbf{U} \Sigma \mathbf{V}^H$ and obtain the $R$-reduced set $\mathbf{U}_R$ and $\Sigma_R$.
5: Set $\mathbf{U}_t = \begin{bmatrix} \mathbf{U}_{t-1} & \mathbf{U}_\delta \end{bmatrix} \mathbf{U}_R$ and $\Sigma_t = \Sigma_R$.

where $\Omega_R \mathbf{A}_R \Omega_R^H$ is the $R$-reduced eigenvalue decomposition of the kernel matrix $\mathbf{X}_{t-1}^H \mathbf{X}_{t-1}$.

For the update, the SVD of the concatenated sample matrix, built from $\mathbf{X}_{t-1}$ and the new mapped samples $\mathbf{X}_\delta$, is to be found. This matrix is given by

$$\mathbf{X}_t = \begin{bmatrix} \mathbf{X}_t & \mathbf{X}_\delta \end{bmatrix} \approx \begin{bmatrix} \mathbf{U}_{t-1} \Sigma_{t-1} \mathbf{V}_{t-1}^H & \mathbf{X}_\delta \end{bmatrix}.$$  \hspace{1cm} (3.17)

Let us reformulate eq. (3.17) as the product of three terms

$$\mathbf{X}_t \approx \begin{bmatrix} \mathbf{U}_{t-1} & \mathbf{U}_\delta \end{bmatrix} \begin{bmatrix} \Sigma_{t-1} & \mathbf{U}_{t-1}^H \mathbf{X}_\delta \\ 0 & \mathbf{U}_\delta^H \mathbf{X}_\delta \end{bmatrix} \begin{bmatrix} \mathbf{V}_{t-1} & 0 \\ 0 & \mathbf{I} \end{bmatrix}^H,$$  \hspace{1cm} (3.18)

where $\mathbf{U}_\delta = \text{orth}(\mathbf{X}_\delta - \mathbf{U} \mathbf{U}^H \mathbf{X}_\delta)$ contains the new components, which are not included in the current subspace $\mathbf{U}_{t-1}$ (here, $\text{orth}(.)$ finds the orthogonal components, and $\mathbf{0}$ implies zero padding). Notice, the SVD of the central term, \text{i.e.}

$$\mathbf{L} = \begin{bmatrix} \Sigma_{t-1} & \mathbf{U}_{t-1}^H \mathbf{X}_\delta \\ 0 & \mathbf{U}_\delta^H \mathbf{X}_\delta \end{bmatrix} = \mathbf{U} \Sigma \mathbf{V}^H,$$  \hspace{1cm} (3.19)

is all that is required for the update. With the $R$-reduced eigenspectrum $\mathbf{U}_R$ and $\Sigma_R$ of $\mathbf{L}$, we find $\mathbf{U}_t = \begin{bmatrix} \mathbf{U}_{t-1} & \mathbf{U}_\delta \end{bmatrix} \mathbf{U}_R$ and $\Sigma_t = \Sigma_R$. Algorithm 3.3 summarizes the update procedure of the setup.

Note that existing methods for incremental KPCA, for which the mapping is generally unknown, are computationally expensive and inexact. For example, Chin and Suter (2007) ensure constant execution speed by introducing a set of preimages to approximate the data matrix, and solve an extra optimization problem similar to eq. (3.8). However, the drawbacks of this method are twofold: (i) the reduced set representation provides only an estimate to the exact solution and (ii) the proposed optimization problem for finding
the reduced set inevitably increases the complexity of the algorithm. In contrast, since in the proposed case, the mapping is explicit and does not increase the dimensionality, the data can be represented directly in feature space. This eliminates the need to introduce an additional optimization problem, making the incremental version of e-PCA both fast and exact, while being robust.

3.3 Applications and Experiments

In this section, I present the evaluation of the proposed Euler PCA (e-PCA) in terms of its applications. In particular, experiments show the performance of e-PCA applied to image de-noising, visual object tracking and background subtraction.

3.3.1 Image Reconstruction under Noise

The robustness of e-PCA is tested for the application on image de-noising using subspace-based image reconstruction. For comparison, I select standard PCA, R1-PCA (Ding et al. 2006), PCA-L1 (Kwak 2008) and HQ-PCA (He et al. 2011b), which represents the state-of-the-art, as well as, standard kernel PCA de-noising with a GRBF KPCA (denoted by G-KPCA) and pre-image computation using eq. (3.14) (denoted by e-PCA-GA). The parameters of R1-PCA, PCA-L1 and HQ-PCA follow those of Ding et al. (2006), Kwak (2008) and He et al. (2011b) respectively. I choose the convergence criterion for R1-PCA, PCA-L1 and HQ-PCA to be based on the norm difference between two successive subspace estimations. The maximum difference is constrained not to exceed 10\(^{-8}\), unless a maximum of 50 iterations is reached. For the optimization of G-KPCA’s variance of the Gaussian kernel, I try two standard approaches, and I report on the best results. In the first approach I set the variance equal to \(\frac{1}{N(N-1)} \sum_{n=1}^{N} \sum_{n'=1}^{N} \|z_n - z_{n'}\|^2_F\) where \(\{z_n\}_{n=1}^{N}\) is the set of training samples (Kwok and Tsang 2004). In the second method I apply a cross-validation strategy in the training set for selecting the variance. For all methods that employ a gradient descent (or ascent) for preimage computation, the preimages are initialized with all pixel intensities equal to 0.5. Convergence is considered to be reached if two successive approximations are nearly equivalent, i.e. \(\|\hat{z}_t - \hat{z}_{t-1}\|^2_F \leq 10^{-4}\).

The dataset consists of a subset of the popular AR Database, produced by Martinez and Benavente (1998). In particular, I use a total of 100 images, size 101 \times 91 pixel, of different subjects. Examples are shown in figure 3.5.

Learning with and without corrupted data is compared in the following. The evaluation is based on the reconstruction error and the angular error (Gunawan et al. 2005; He et al. 2011b; Krzanowski 1979; Kwak 2008). The reconstruction error is commonly found
in previous approaches (He et al. 2011b; Kwak 2008). In particular, for \( N \) test samples, the reconstruction error computes

\[
e_r(R) = \frac{1}{N} \sum_{n=1}^{N} \left\| z^{(o)}_n - \sum_{r=1}^{R} u^{(c)}_r u^{(c)H}_r z^{(c)}_n \right\|_F^2,
\]

where \( z^{(o)} \) and \( z^{(c)} \) represent the original, uncorrupted images and the corrupted training image respectively, \( U^{(c)} = \begin{bmatrix} u^{(c)}_1 & \cdots & u^{(c)}_R \end{bmatrix} \in \mathbb{C}^{P \times R} \) is the estimated subspace of the corrupted data, and \( R \) denotes the number of components used. For the methods that use preimages for approximating the reconstruction error, eq. (3.20) is reformulated as

\[
e_r(R) = \frac{1}{N} \sum_{n=1}^{N} \left\| z^{(o)}_n - \tilde{z}^{(c(R))}_N \right\|_F^2,
\]

where \( \tilde{z}^{(c(R))}_N \) is the preimage associated with the reconstruction using \( R \) components in the feature space (i.e. solving optimization problems eq. (3.8) and eq. (3.9) using \( R \) components of matrix \( U^{(c)} \)). Note, that for \( e\)-PCA de-noising is performed by calculating preimages in two ways: (i) by applying the \( \angle \)-operator and (ii) by the gradient ascent optimization. I denote the former method as \( e\)-PCA and the latter method as \( e\)-PCA-GA. The calculation of preimages for \( G\)–KPCA is also performed in a similar fashion.

Additionally to the reconstruction error, the angular error between the corrupted subspace \( U^{(c)} \) (learned from the corrupted training set) and the uncorrupted subspace \( U^{(o)} = \begin{bmatrix} u^{(o)}_1 & \cdots & u^{(o)}_R \end{bmatrix} \in \mathbb{C}^{P \times R} \) (learned from the original, uncorrupted images) is used as follows (Gunawan et al. 2005; Krzanowski 1979)

\[
e_a(R) = R - \sum_{r=1}^{R} \sum_{r'=1}^{R} \left\| u^{(o)H}_r u^{(c)}_{r'} \right\|.
\]

Notice, for the nonlinear methods \( U^{(o)} \) and \( U^{(c)} \) are in the feature space. Here \( u^{(o)H}_r u^{(c)}_{r'} \) can be efficiently computed using the kernel. The main advantage of the angular error is provided when contrasting it to the reconstruction error: The reconstruction error introduces an inherent error due to the chosen number of components, the angle error
shows the difference caused by the outliers directly and without any offsets. I now conduct my experiments with synthetic occlusions and occlusions by hands.

**Synthetic Corruptions**

In this experiment, a percentage of training images is corrupted by randomly placed patches of random pixel noise as shown in figure 3.6. Both, the number of corrupted images and the size of the corrupted area is varied. For convenience, I say “a\% of images by b\%” to denote that a\% of the images in the training set is corrupted by randomly placed patches, which are the size of b\% of the total image. After training, the results are analyzed based on the reconstruction and angular errors. In particular, the five different methods are tested within six setups. These setups can be summarized into three categories:

- **Type (i):** *large occlusions on few images*, e.g.
  - 5\% of images by 30\%
  - 10\% of images by 30\%

- **Type (ii):** *medium sized occlusions on a few images*, e.g.
  - 15\% of images by 10\%
  - 25\% of images by 10\%

- **Type (iii):** *small occlusions on many images*, e.g.
  - 80\% of images by 5\%
  - 85\% of images by 5\%

Before comparing to other methods, the proposed e-PCA is analyzed individually. Prior to the experiments, \(\alpha\) of e-PCA is optimized with a validation set. For \(\alpha < 2\) the best performance is attained when \(\alpha = 1.9\). Figure 3.7 verifies these findings on the dataset. Note, \(\alpha\) is kept fixed to \(\alpha = 1.9\) for all the experiments in this chapter. This is in contrast with G-KPCA which includes an extra step for finding the optimum variance.

In order to test whether the preimage approximation using eq. (3.14) is a valid choice, the attained minimum of the optimization problem in eq. (3.9) is calculated after performing the gradient ascent, and compared to the error given in eq. (3.15). For all experiments, the error of the fast preimage approximation results in a similar or lower error. A representative example can be found in figure 3.8 where the preimage approximation error is plotted over the number of components. It is evident that both the preimage computation methods produce similar results. It is important to note here, that the preimage
Figure 3.6: Reconstructed images after learning with random corruptions. 80% of the images are occluded by an area of 5%, and 20 components are used for learning. The final row shows the corrupted training data. The uncorrupted images are shown in figure 3.5.
Figure 3.7: Reconstruction error and angular error for different values of $\alpha$. 

3.3. Applications and Experiments
approximation error is different to the reconstruction error in eq. (3.21), since the former is in the feature space while the latter is in the input space.

I now conduct the comparison between $e$-PCA and all other setups. Figure 3.9 shows the reconstruction errors of the tested methods. In type (i) and (ii), HQ-PCA performs well for few components, while R1-PCA performs worse than HQ-PCA, but better than standard PCA. As the number of components increases, PCA, R1-PCA and HQ-PCA perform similarly. PCA-L1 performs well only for a small number of components. G-KPCA performs as well as HQ-PCA. In all examples, both versions of $e$-PCA (i.e. $e$-PCA and $e$-PCA-GA) perform the best, even for a large number of components.

More distinctions between the tested methods are observable for type (iii). Here, PCA and R1-PCA have similar performance, up to 30 components, after which R1-PCA outperforms standard PCA. Similar conclusions can be drawn for HQ-PCA. Again, $e$-PCA and $e$-PCA-GA outperform all other methods. Figure 3.6 visualizes the qualitative reconstruction results. Most methods introduce significant corruptions to the images, e.g. noisy data artifacts from other training instances. As it can be seen, the proposed method is able to largely suppress such outliers.

The angular error results reveal different performance, as can be seen in figure 3.10. HQ-PCA outperforms PCA-L1 only for a large number of components. R1-PCA and standard PCA perform similarly. G-KPCA seems to perform the second best. This suggest that the performance improvement obtained for G-KPCA might be due to the fact that, for this experiment, the calculation of preimages is not necessary. It seems that this calculation, as required by the reconstruction experiment, might be problematic. Once more, the two versions of the proposed $e$-PCA ($e$-PCA and $e$-PCA-GA) perform best. Also notice, these methods are equivalent for the angular error, which is calculated in feature space.

Hand Occlusions

In the second part of the image reconstruction experiment, skin-like occlusions are used to verify the results of the previous section. In particular, a subset of the training data
Figure 3.9: Reconstruction error with different rates of occlusions. Here, the mean value over 10 executions with different random patches is shown. The variance is indicated by error bars.
Figure 3.10: Angular error with different rates of occlusions. Here, the mean value over 10 executions with different random patches is shown. The variance is indicated by error bars.
is occluded by hand signs, taken from the American fingerspelling alphabet.\footnote{The fingerspelling alphabet is a subset of sign language which is utilized for spelling names. Examples can be found at http://asl.ms/} The chosen sign (letter), its orientation and its position are randomized, and the skin color is adjusted to suit the subject. Some examples of the resulting corruptions can be seen in figure 3.11.

Figure 3.12 shows the reconstruction error and figure 3.13 the angular error. As before, HQ-PCA and G-KPCA outperform R1-PCA and standard PCA in terms of the reconstruction error. Again, PCA-L1 performs the worst. The two versions of e-PCA, e-PCA and e-PCA-GA perform the best. In terms of the angular error, the performances are slightly different. In contrast to our previous setup, here, R1-PCA, PCA, e-PCA and e-PCA-GA perform similarly well. Again, G-KPCA performs very well for this experiment. However, HQ-PCA and PCA-L1 perform worse. The corruption by hand occlusions is much more subtle than the one introduced by the random pixel patches. Therefore, PCA and R1-PCA achieve a similar performance to e-PCA in terms of angular error. Nonetheless, the general trend follows the results of the previous section: the versions of e-PCA work well in terms of both, the reconstruction error and the angular error. Figure 3.14 shows an example of the reconstruction quality.

### 3.3.2 Object Tracking

The incremental version of e-PCA is applied to the application of visual tracking. The aim of a visual tracking system is to locate a predefined target object on every frame of a video sequence. In this section, the tracking performance of the proposed e-PCA is evaluated, based on precision and accuracy, and compared to four other popular holistic trackers.

#### Framework

The appearance model learned by the incremental version of e-PCA is combined with a motion affine transformation and a particle filter in a similar fashion to the methods of Ross et al. (2008) and Chin and Suter (2007) which are discussed in chapter 2. Generally,
Figure 3.12: Reconstruction error with different rates of hand occluded images. Here, the mean value over 10 executions is shown. The variance is indicated by error bars.
Figure 3.3. Applications and Experiments
Figure 3.14: Reconstructed images after learning with skin-like corruptions. 50% of the images are corrupted by hand occlusions, and 20 components are used for learning. The final row shows the training data. The uncorrupted images are shown in figure 3.5.
a particle filter calculates the posterior of a system’s states based on a transition model and an observation model. In such tracking framework, the transition model is described as a GMM around an approximation of the state posterior distribution of the previous time step,

$$ p(A_m(t) | \{A_m(t-1)\}_{m=1}^M) = \sum_{m'=1}^M w_{m'}^{(t-1)} \mathcal{N}(A_{m'}^{(t-1)}, \Xi) | A_m(t), \quad (3.23) $$

where $A_m(t)$ is the affine transformation of the $m$th particle at time $t$, $\{A_m^{(t-1)}\}_{m=1}^M$ is the set of $M$ transformations of the previous time step, whose corresponding set of weights is denoted by $\{w_{m'}^{(t-1)}\}_{m=1}^M$, and $\Xi$ is an independent covariance matrix, which represents the variance of the affine transformations, i.e. the horizontal and vertical displacement, rotation, scale, ratio and skew.

The observation model computes the probability of a sample being generated by the learned eigenspectrum in the appearance model. As shown by Ross et al. (2008), for linear systems, this is proportional to an exponential. In particular, given the set of observations $\{z_m^{(t)}\}_{m=1}^M$ at time $t$, the probability of a sample is provided by the proportional equivalence

$$ p(z_m^{(t)}) \propto e^{-\gamma \|z_m^{(t)} - U_{t}z_m^{(t)}\|_F^2}, \quad (3.24) $$

where $\gamma$ is the parameter that controls the spread. Notice, the exponential is solely dependent on the reconstruction error. Equivalently, I assume the observation model for e-PCA to be proportional to the mapped sample’s reconstruction error in feature space.
Algorithm 3.4 Euler Tracker at Time $t$

**Input:** The previous eigenspectrum $U_{t-1}$, $U_{t-1}$, locations $\{A^{(t-1)}_m\}_{m=1}^M$, weights $\{w^{(t-1)}_m\}_{m=1}^M$, image frame $F_t$ and $\alpha$.

1. Draw $M$ particles $\{A^{(t)}_m\}_{m=1}^M$ from the transition model $p(A^{(t)}_m|\{A^{(t-1)}_m\}_{m=1}^M)$ as in eq. (3.23).
2. Take all image patches $\{I^{(t)}_m\}_{m=1}^M$ from $F_t$ which correspond to the particles in $\{A^{(t)}_m\}_{m=1}^M$.
3. Extract the pixel values (in the range $[0,1]$) in lexicographical order to form the observation vectors $\{z^{(t)}_m\}_{m=1}^M$.
4. Apply the mapping in eq. (3.4) to each observation vector, and form $\{x^{(t)}_m\}_{m=1}^M$.
5. Compute the exponential in eq. (3.25) for each particle and normalize these as weights $\{w^{(t)}_m\}_{m=1}^M$ to form a probability distribution.
6. Choose $A^{(t)}_b$ and $x^{(t)}_b$ as the affine transform and features of the particle with the largest weight.
7. Using $x^{(t)}_b$, update the subspace by applying algorithm 3.3 in a batch after a certain number of frames (5 in this implementation).

In particular, I compute

$$e^{-\gamma \|\phi(x^{(t)}_m) - U_{t}U_{t}^H\phi(z^{(t)}_m)\|^2_F} = e^{-\gamma \|x^{(t)}_m - U_{t}U_{t}^Hx^{(t)}_m\|^2_F}, \quad (3.25)$$

where $x^{(t)}_m$ is the observation’s mapping according to eq. (3.4).

The tracking framework consists of two stages. In the first phase, $M$ particles are drawn from eq. (3.23). In the second phase, the observation model is applied to estimate the weighting for the next iteration (the weights are normalized to ensure $\sum_{m=1}^M w^{(t)}_m = 1$). Furthermore, the most probable sample is selected as the estimated state at time $t$ (i.e. it is the computed guess of the target’s location). Let $A^{(t)}_b$ denote the best particle in terms of probability, then $e$-PCA is updated by its feature vector $x^{(t)}_b$. Notice, the estimation of the posterior distribution is an incremental process and utilizes a hidden Markov model (HMM) which only relies on the previous time step.

In chapter 2, I review the incremental visual tracking framework for standard PCA. Notice, the new setup differs only in the feature representation, as I exploit the direct mapping in eq. (3.4) to compute $e$-PCA. Figure 3.15 shows the main steps of the proposed tracking framework, and algorithm 3.4 describes its details. Finally, I coin the proposed system Euler tracker ($eT$).

**Results**

I present the performance evaluation results of the proposed Euler tracker ($eT$). The performance of $eT$ is compared with that of four other commonly used tracking approaches:

Chapter 3. Euler Principal Component Analysis 73
• IVT by Ross et al. (2008), which is publicly available as Matlab implementation at http://www.cs.toronto.edu/~dross/ivt/

• the incremental KPCA tracker (denoted IKPCA) of Chin and Suter (2007), for which the Matlab implementation was kindly provided by the authors

• the L1 tracker proposed by Mei and Ling (2009), which can publicly be downloaded from http://www.ist.temple.edu/~hbling/code_data.htm

• the MIL tracker by Babenko et al. (2011), which has an implementation only for a translation motion model, available at http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml. I carefully modified the code in order to support an affine motion model in a particle filter framework.

All methods are tested on eight very popular video sequences (subsets of which are used by Babenko et al. (2011); Comaniciu et al. (2003); Mei and Ling (2009); Ross et al. (2008)), Vid1 to Vid8, with drastic changes of the target’s appearance including pose variation, occlusions and non-uniform illumination. The videos Vid4 and Vid5 are available at http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml and the remaining videos are published at http://www.cs.toronto.edu/~dross/ivt/. Qualitative results are illustrated in figure 3.18 to figure 3.25.

Video Vid1 is provided along with seven annotated points which indicate the ground truth. I annotate three to seven fiducial points for the remaining sequences. The quantitative performance evaluation is based on the root mean square (RMS) error between the true and the estimated locations of these points, similar to Ross et al. (2008). Like Babenko et al. (2011), precision plots which visualize the quality of the tracking are additionally presented. Such graphs show the percentage of frames in which the target is tracked with an RMS error less than a certain threshold.

In the experiments, all trackers use an affine motion model with a fixed number of drawn particles (800 particles). I attempt to optimize the performance of all trackers using video-specific parameters. That is, for each tracker and video, I find the parameters which give the best performance in terms of robustness (i.e. how many times the tracker went completely off) and accuracy (measured by the RMS error).

Apart from the L1 tracker (for which the resolution of the template increases geometrically the complexity) the resolution of the tracking template is chosen to be of 32 × 32 pixel. All trackers are optimized with respect to the variance of the GMM from which I sample the particles. Additionally to the variance of the GMM, which is common for all the systems, I optimize $e_T$, IVT and IKPCA with respect to the number of components and the spread $\gamma$. For $e_T$ the value of $\alpha$ is fixed to 1.9. IKPCA is also optimized with respect
Table 3.2: Mean RMS error for general tracking, and tracking rate. I write “(lost)” to indicate sequences in which the tracker clearly does not follow the target throughout after visual inspection.

to the variance of the GRBF function. Furthermore, I optimize L1 with respect to the resolution of the templates (as the tracking becomes impractical for particles larger than 20 × 20). For MIL I optimize with respect to the parameters described by Babenko et al. (2011) (e.g. the number of positive patches in each frame, the number that controls the sampling of negative examples, the learning rate for the weak classifiers).

Table 3.2 lists the mean RMS error for all sequences, and the average frame rate of each tracker,\(^3\) while figure 3.16 plots the RMS error for each frame individually. Figure 3.17 shows the accuracy in terms of precision plots. Qualitative tracking results for all methods are shown in figure 3.18 to figure 3.25.

In general, the robustness of eT is similar to IVT, although, eT performs the best in terms of precision for most videos. MIL and L1 are more robust and track the target in Vid5 successfully. However, particularly visible in the results for Vid8, L1 is not precise for outliers caused by motion blur. MIL is based on a bag-of-features approach, and consequently is inherently imprecise. IKPCA fails for all sequences. On the contrary, eT performs very well particularly for Vid4 in which the target undergoes many prolonged partial occlusions. The robustness of e-PCA successfully suppresses the outliers caused by occlusions in this video sequence. In terms of efficiency, IVT and eT operate in the highest frame rate, while all other methods operate in less than one frame per second.

\(^3\)Tests were conducted in MATLAB on a desktop computer with an Intel core i7 870 processor at 2.93 GHz and 8 GB RAM.

<table>
<thead>
<tr>
<th>Videos</th>
<th>Vid1</th>
<th>Vid2</th>
<th>Vid3</th>
<th>Vid4</th>
<th>Vid5</th>
<th>Vid6</th>
<th>Vid7</th>
<th>Vid8</th>
<th>Frames/Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVT</td>
<td>6.82</td>
<td>(lost)</td>
<td>4.07</td>
<td>10.79</td>
<td>(lost)</td>
<td>3.31</td>
<td>1.78</td>
<td>2.62</td>
<td>3.157</td>
</tr>
<tr>
<td>IKPCA</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>0.832</td>
</tr>
<tr>
<td>L1</td>
<td>6.17</td>
<td>(lost)</td>
<td>2.87</td>
<td>11.10</td>
<td>12.68</td>
<td>9.53</td>
<td>1.62</td>
<td>13.58</td>
<td>0.076</td>
</tr>
<tr>
<td>MIL</td>
<td>16.95</td>
<td>(lost)</td>
<td>13.61</td>
<td>14.62</td>
<td>37.56</td>
<td>12.73</td>
<td>4.14</td>
<td>23.87</td>
<td>0.129</td>
</tr>
<tr>
<td>eT</td>
<td>5.14</td>
<td>(lost)</td>
<td>3.68</td>
<td>4.68</td>
<td>(lost)</td>
<td>3.04</td>
<td>1.73</td>
<td>2.44</td>
<td>2.935</td>
</tr>
</tbody>
</table>
Figure 3.16: Mean RMS error of all tested trackers at each frame. Vid1 to Vid8 are shown.
Figure 3.17: Tracking precision for each video sequence.
Figure 3.18: Qualitative tracking results for Vid1.

Figure 3.19: Qualitative tracking results for Vid2.

3.3. Applications and Experiments
Figure 3.20: Qualitative tracking results for Vid3.

Figure 3.21: Qualitative tracking results for Vid4.
Figure 3.22: Qualitative tracking results for Vid5.

Figure 3.23: Qualitative tracking results for Vid6.
Figure 3.24: Qualitative tracking results for Vid7.

Figure 3.25: Qualitative tracking results for Vid8.
3.3.3 Background Modeling

Background modeling algorithms aim to estimate the background of a scene from a video sequence usually captured with a static camera. This problem can be naturally tackled using PCA (Oliver et al. 2000). In particular, the frames of the video are used to estimate a low dimensional subspace. Then the background corresponding to each of the video frames is obtained by its reconstruction via the subspace. Once the background estimate is obtained, the foreground objects can be segmented typically by subtraction and thresholding.

For the evaluation, the popular data set by Li et al. (2004) is used. This set consists of nine videos including illumination changes, indoor and outdoor environments, as well as dynamic background changes. The ground truth for foreground and background pixels of 20 randomly selected frames for each video is also provided by Li et al. (2004). Standard
Table 3.3: Maximum similarity on background modeling.

PCA, PCA-L1, R1-PCA and HQ-PCA are used for comparison. Quantitative and qualitative results are presented. For the former, the common Jaccard similarity measure is used, given by (Maddalena and Petrosino 2008)

$$\text{Similarity} = \frac{tp}{tp + fp + fn}$$  \hspace{1cm} (3.26)

where $tp$, $fp$ and $fn$ are the numbers of correctly labeled foreground, falsely labeled background and falsely labeled foreground pixels respectively. The setup for PCA, PCA-L1, R1-PCA, HQ-PCA and $e$-PCA is similar to that in section 3.3.1. Furthermore, only PCA and $e$-PCA are updated incrementally for each frame during learning. All other methods are computed in their entirety at each frame to simulate online learning.

The complete set of preceding frames is employed to train the models (e.g. for frame 100, the preceding 99 frames are used for the appearance model), and for each video, the similarity for the frames in which the ground truth is provided is evaluated. The mean similarity over the number of components, is plotted in figure 3.26. The best similarity value for each method and video is summarized in table 3.3, while figure 3.27 shows the performance qualitatively. In general, $e$-PCA performs the best in five out of nine sequences, and the second best for three. The results of the other methods vary for each video. Also shown in table 3.3 are the results of the batch version for $e$-PCA ($e$-PCA-batch). Here the model is computed from the whole training data at each frame, and no incremental update is utilized. Note, $e$-PCA and $e$-PCA-batch perform with equivalent accuracy.

In table 3.4, the computation time of the appearance model for the final frame of each video sequence is shown.\(^4\) PCA and $e$-PCA can be updated incrementally, making their running time significantly less than 1 second for all sequences. In contrast, the other methods require a recalculation of the complete appearance model for each frame. Thus, these methods are much slower. Notice also, $e$-PCA-batch suffers from the non-incremental

---

\(^4\)Tests were conducted in MATLAB on a desktop computer with an Intel core i7 870 processor at 2.93 GHz and 8 GB RAM
<table>
<thead>
<tr>
<th>PCA</th>
<th>PCA-L1</th>
<th>R1-PCA</th>
<th>HQ-PCA</th>
<th>e-PCA</th>
<th>Original</th>
</tr>
</thead>
</table>

![Figure 3.27: Examples of background modeling for each video and each method. In the results, black indicates correctly predicted background, blue indicates correctly predicted foreground, red indicates misclassified background and white indicates misclassified foreground.](image)

3.3. Applications and Experiments
Table 3.4: Execution time required to compute (or update) the appearance model for the last frame of each video sequence (5 components).

<table>
<thead>
<tr>
<th></th>
<th>airport</th>
<th>bar</th>
<th>lobby</th>
<th>curtain</th>
<th>escalator</th>
<th>fountain</th>
<th>mall</th>
<th>campus</th>
<th>water</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>0.003s</td>
<td>0.002s</td>
<td>0.003s</td>
<td>0.002s</td>
<td>0.002s</td>
<td>0.002s</td>
<td>0.002s</td>
<td>0.010s</td>
<td>0.003s</td>
</tr>
<tr>
<td>PCA-L1</td>
<td>12.4s</td>
<td>9.5s</td>
<td>5.1s</td>
<td>6.3s</td>
<td>21.7s</td>
<td>1.2s</td>
<td>3.3s</td>
<td>9.7s</td>
<td>1.4s</td>
</tr>
<tr>
<td>R1-PCA</td>
<td>123.0s</td>
<td>120.0s</td>
<td>44.7s</td>
<td>124.9s</td>
<td>233.0s</td>
<td>12.4s</td>
<td>35.9s</td>
<td>182.8s</td>
<td>8.1s</td>
</tr>
<tr>
<td>HQ-PCA</td>
<td>2663.4s</td>
<td>930.8s</td>
<td>365.8s</td>
<td>3106.8s</td>
<td>2536.7s</td>
<td>36.7s</td>
<td>91.0s</td>
<td>282.3s</td>
<td>71.8s</td>
</tr>
<tr>
<td>$e$-PCA</td>
<td>0.007s</td>
<td>0.006s</td>
<td>0.006s</td>
<td>0.006s</td>
<td>0.006s</td>
<td>0.006s</td>
<td>0.006s</td>
<td>0.026s</td>
<td>0.006s</td>
</tr>
<tr>
<td>$e$-PCA-batch</td>
<td>178.2s</td>
<td>103.8s</td>
<td>19.3s</td>
<td>88.2s</td>
<td>148.7s</td>
<td>3.2s</td>
<td>25.4s</td>
<td>82.0s</td>
<td>5.3s</td>
</tr>
</tbody>
</table>

Figure 3.28: Per frame running time of $e$-PCA and its batch version $e$-PCA-batch.

update. Figure 3.28 shows the per frame running time of $e$-PCA and $e$-PCA-batch for the airport sequence in comparison. Clearly, $e$-PCA-batch is inferior as its execution increases exponentially.

3.4 Conclusions

A fast, direct and robust approach to PCA is introduced in this chapter. I call this approach Euler PCA ($e$-PCA) as it employs an explicit mapping with complex numbers, in a KPCA framework. The proposed $e$-PCA allows for fast incremental computation and retains the favorable properties of standard $\ell_2$-norm PCA, while suppressing outliers.

In the experiments, I show that $e$-PCA achieves promising results for the applications of face reconstruction, object tracking and background modeling.

Furthermore, I conclude that the use of an explicit mapping in a RKHS is advantageous, as it allows for robustness while remaining efficient. This realization will continue to build in this thesis. In the next chapter, I will introduce another useful KPCA with an explicit feature set, designed for comparing vectors of rotations in 3D.
The theories of kernel PCA (KPCA) are applied to formulate Euler PCA (e-PCA) in the previous chapter. Recall, the main advantage of e-PCA in comparison to standard KPCA is its explicit mapping into the reproducing kernel Hilbert space (RKHS). With this mapping, it is possible to produce a robust incremental PCA which allows for fast computation.

In this chapter, a similar approach is taken. First, a new distance measure which is designed for matching vectors of 3D rotations is introduced. The distance is a natural, but non-trivial extension of the cosine-based distance by Fitch et al. (2005) and the gradient-based distance of Tzimiropoulos et al. (2012b), and can be expressed as a kernel method. Second, the distance leads to an explicit mapping using a special formulation of unit length quaternion numbers, which I call the full-angle quaternion (FAQ) representation. Similarly to e-PCA, the mapping facilitates fast computation of the proposed measure. Finally, the proposed method is applied to 3D object recognition from point clouds, and object tracking from color video.

Notice, the matching of vectors consisting of 3D rotations is the chosen problem in this chapter, as it spans a wide range of applications. In 3D shape recognition, the problem of evaluating shape poses is converted into the problem of robustly matching vectors of direct similarities (i.e. transformations with a uniform scale, rotation, and translation (Coxeter 1961)). In particular, with the method for concurrent hashing of scale and translation proposed by Liwicki et al. (2014), the rotations of the similarities are extracted to form vectors of 3D rotations, which are then evaluated using the proposed distance. For object tracking, the problem of robustly matching color patches is formulated as an online subspace learning task for vectors of rotations.

The contributions in this chapter are summarized as follows:

1. A closed-form distance between 3D rotations is proposed, which allows for robust matching and subspace learning with vectors of 3D rotations.
2. A new representation for 3D rotations is introduced, called full-angle quaternion (FAQ), which allows for the distance to be expressed as Euclidean. Unlike the standard half-angle quaternion (HAQ) representation, where one rotation is represented by two quaternions, any 3D rotation is mapped to a single unit quaternion via the FAQ representation.

3. The presented framework is applied to 3D shape recognition, and 2D tracking with color images.

First, in section 4.1, the most common representations for 3D rotations are summarized. Section 4.2 reviews the robust cosine-based distance between 2D rotations from which the presented approach generalizes, and existing non-robust distances between 3D rotations. The proposed robust distance between 3D rotations and the FAQ representation is introduced in section 4.3, together with their properties. The application to 3D object recognition and 2D tracking in color images are presented in section 4.4. Experiments on these two setups are shown in section 4.5. Section 4.6 concludes this chapter.

4.1 Representing 3D Rotations

The representation of rotations in 3D is very diverse in the literature (Bauchau and Trainelli 2003; Shuster 1993). I present the most popular methods in the following. The notation assumes a rotation to be composed of a rotation angle \( \theta \in [-\pi, \pi] \) and a rotation vector \( \mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \in \mathbb{R}^3 \) of unit length. I write \( r = (\theta, \mathbf{v}) \in \mathcal{SO}(3) \) to represent the rotation in the space of 3D rotations.

4.1.1 Euler Angles

Euler angles describe a rotation in 3D fairly intuitively. In this representation a vector of angles \( \mathbf{u} = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T \) describes three consecutive rotations. However, many interpretations of Euler angles exist (Huynh 2009). Figure 4.1 visualizes one possible implementation. In this example, the azimuth and the altitude of the rotation vector provide \( \alpha \) and \( \beta \) via the \( \angle \)-operator, while \( \gamma \) is equivalent to the rotation angle. In particular, the vector of Euler angles is provided by

\[
\mathbf{u} = \begin{bmatrix} \angle(v_x + i v_y) \\ \angle(v_y + i v_z) \\ \theta \end{bmatrix}.
\]  

(4.1)
In computer vision, Euler angles are seldom selected, as redundancies may be introduced. For example, multiple vectors represent the same rotation. This is referred to as a gimbal lock (Huynh 2009).

### 4.1.2 Rotation Matrix

The rotation matrix (RM) representation of 3D rotation is probably the most common representation. Here, a 3D rotation is written as a $3 \times 3$ orthogonal matrix with determinant 1. In particular, the RM $\mathbf{R}$ of $\mathbf{r} = (\theta, \mathbf{v})$ is given by

$$
\mathbf{R} = \cos \theta \mathbf{I} + \sin \theta \begin{bmatrix}
0 & -v_z & v_y \\
v_z & 0 & -v_x \\
-v_y & v_x & 0
\end{bmatrix} + (1 - \cos \theta) \mathbf{vv}^T.
$$

One strength of the RM representation is that the calculation of transforming points can be computed very efficiently via a single matrix multiplication. It is, however, difficult to use rotation matrices in solving optimization problems. Notice, a RM can be thought of as a vector in 9-dimensional space, with six constraints: three for imposing unit length in the columns of $\mathbf{R}$, and three for pairwise orthogonality between columns. The additional
constraints are quadratic, thereby turning any optimization problem into a non-linearly constrained optimization problem.

### 4.1.3 Rotation Vector

Like Euler angles, rotation vectors (RV) define a rotation using minimal storage, as three parameters are required. In particular, the rotation vector $r$ for rotation $r$ is written as

$$ r = \theta v. \quad (4.3) $$

Due to their compact representation, RVs are sometimes found in computer vision. Their application, e.g. rotating a coordinate frame or concatenating multiple rotations, is however hard to compute. Here, HAQ provide a better compromise.

### 4.1.4 Half-Angle Quaternion

Before I present the quaternion representation of 3D rotations, let me introduce the concept of quaternions in general.

#### Quaternion Algebra

Quaternions form a natural extension of complex numbers (Zhang 1997). Similarly to complex numbers, they are composed of a real and an imaginary part. Let us denote the quaternion space (i.e. the Hyper-complex space) as $\mathbb{H}$. The numbers in this space, i.e. $q \in \mathbb{H}$, are written as

$$ q = a + ib + jc + kd, \quad (4.4) $$

where $a, b, c, d \in \mathbb{R}$, and $i, j$ and $k$ form the three imaginary bases, similar to the imaginary of a complex number. While addition simply adds the scalar values of each basis element, the multiplication is defined by

$$ i^2 = j^2 = k^2 = ijk = -1. \quad (4.5) $$

In analogy to complex numbers, the conjugate of $q$ is denoted by

$$ \overline{q} = a - ib - jc - kd \quad (4.6) $$
and the conjugate transpose of a quaternion matrix by \((\cdot)^\mathbb{H}\). The squared Frobenius norm of a \(P\)-dimensional vector of quaternions \(q = [q_1 \ldots q_P]^T \in \mathbb{H}^P\) is given by

\[
\|q\|_F^2 = \sum_{p=1}^{P} q_p q_p^\mathbb{H} = q^\mathbb{H} q.
\] (4.7)

### Unit Quaternion as 3D Rotation Representation

Unit quaternions can be used to represent 3D rotations \(\text{via}\) a standard map, given by

\[
q = \cos \frac{\theta}{2} + (i v_x + j v_y + k v_z) \sin \frac{\theta}{2} \tag{4.8}
\]

or its antipodal equivalent, as \(r\) is equivalent to the angle-axis pair \((2\pi + \theta, v)\),

\[
q = -\cos \frac{\theta}{2} - (i v_x + j v_y + k v_z) \sin \frac{\theta}{2}. \tag{4.9}
\]

As both mappings are composed out of the halved rotation angle \(\frac{\theta}{2}\), I refer to this representation as half-angle quaternion (HAQ). The HAQ representation is probably the most common representation of 3D rotations in computer vision. Compared to the RM representation, it is more suitable for optimization problems, as it consists of only one single constraint (\(i.e.\) the unit length constraint). The application of the rotation on points is easy to compute, as is the concatenation of two rotations (Zhang 1997). However, HAQ suffers from non-uniqueness, as a 3D rotation can be represented by exactly two antipodal unit quaternions.

### 4.2 Existing Rotation Distances

In the following sections, I briefly review existing distances for 2D and 3D rotations. In particular, I present the robust method by Tzimiropoulos et al. (2012b) in 2D, and distances derived from the 3D rotation representations above.

#### 4.2.1 Robust Rotation Distance in 2D

Tzimiropoulos et al. (2012b) introduce the robust cosine-based distance between gradient orientations to match vectors of 2D rotations for face recognition and tracking. Given an image \(\mathcal{I}_i\) with \(P\) pixels, Tzimiropoulos et al. (2012b) record the direction of the intensity gradient at each pixel as rotation angles \(\{\theta_{p}^{(i)}\}_{p=1}^{P}.\) The angles in \(\{\theta_{p}^{(i)}\}_{p=1}^{P}\) are then put into lexicographical ordering to form the vector \(x_i = [\theta_{1}^{(i)} \ldots \theta_{P}^{(i)}]^T \in \mathcal{SO}(2)^P,\) where
\(\cos(\theta_i - \theta_j)\) and \(\frac{-2\pi}{-\pi 0 \pi}\) are the highlighted areas, while outliers are uniformly distributed over the whole space.

The advantage lies in the sum of cosines. In particular, Tzimiropoulos et al. (2010) show that for an uncorrelated area \(P\), with random angle directions, the distance values are almost uniformly distributed. For example, given image \(I_i\) and its corrupted version \(I_j\), which is uncorrelated in \(P\), the sum of the corruptions cancel, i.e., \(\sum_{p \in P} \cos(\theta_i^{(p)} - \theta_j^{(p)}) \approx 0\). Thus, the distance tends towards the mean of its range. For highly correlated vectors of rotations, the distance is near 0. Hence, while inliers achieve the ideal distance of 0, outliers have less effect and only shift the distance towards the mean (and not the maximum value).

Figure 4.2 visualizes the distance function and its robust properties. The advantage lies in the sum of cosines. In particular, Tzimiropoulos et al. (2010) show that for an uncorrelated area \(P\), with random angle directions, the distance values are almost uniformly distributed. For example, given image \(I_i\) and its corrupted version \(I_j\), which is uncorrelated in \(P\), the sum of the corruptions cancel, i.e., \(\sum_{p \in P} \cos(\theta_i^{(p)} - \theta_j^{(p)}) \approx 0\). Thus, the distance tends towards the mean of its range. For highly correlated vectors of rotations, the distance is near 0. Hence, while inliers achieve the ideal distance of 0, outliers have less effect and only shift the distance towards the mean (and not the maximum value).

Beside 2D face recognition and tracking, the cosine-based distance has been employed to 3D algorithms by Marras et al. (2012). Here, by using the image and its depth map, the camera facing 3D surface normals of an object are computed, and the distance is applied to robustly match vectors of the surface normals’ azimuth angles. Again, the cosine-based kernel of angle differences is thought to be near zero for corrupted data.

\begin{equation}
\frac{-2\pi}{-\pi 0 \pi \theta_i - \theta_j}
\end{equation}

(4.10)
4.2.2 Rotation Distances in 3D

In the presented methods above, a special representation of 2D rotations enables a robust and efficient framework for comparing lists of 2D angles. Comparison of more general 3D rotations is, however, unsupported for the cosine-based kernel. In this section, dissimilarity measures for rotations in 3D are presented.

Numerous distances for 3D rotations have been proposed in the literature (Gramkow 2001; Hartley et al. 2013; Huynh 2009; Moakher). Most of these are Euclidean distances (and variants) under different representations of 3D rotations. The Euler angles distance is the Euclidean distance between Euler angles (Huynh 2009). In particular, given \( r_i \) and \( r_j \), or equivalently \( \mathbf{v}_i = \begin{bmatrix} \alpha_i & \beta_i & \gamma_i \end{bmatrix}^T \) and \( \mathbf{v}_j \), the distance is provided by

\[
d_1(r_i, r_j) = \sqrt{d_*(\alpha_i, \alpha_j)^2 + d_*(\beta_i, \beta_j)^2 + d_*(\gamma_i, \gamma_j)^2},
\]

(4.11)

where the distance between angles is given by

\[
d_*(\alpha_i, \alpha_j) = \min\{\|\alpha_i - \alpha_j\|, 2\pi - \|\alpha_i - \alpha_j\|\}.
\]

(4.12)

The Euclidean distance under the HAQ representation, \( q_i, q_j \in \mathbb{H} \) leads to the vectorial quaternion distance (Gramkow 2001; Hartley et al. 2013; Huynh 2009),

\[
d_2(q_i, q_j) = \min\{\|q_i - q_j\|, \|q_i + q_j\|\}
\]

(4.13)

and the inverse cosine quaternion distance (Huynh 2009),

\[
d_3(q_i, q_j) = \arccos(\text{Re}(q_i^H q_j)),
\]

(4.14)

where \( \text{Re}(\cdot) \) extracts the real part of the quaternion. Analysis of geodesics on \( \mathbb{SO}(3) \) leads to intrinsic distances (Hartley et al. 2013; Huynh 2009; Moakher), which are the Euclidean distance between RVs \( r_i \) and \( r_j \),

\[
d_4(r_i, r_j) = \sqrt{r_i^T r_j}.
\]

(4.15)

The Euclidean distance in the embedding space \( \mathbb{R}^9 \) of \( \mathbb{SO}(3) \) induces the extrinsic distance (Gramkow 2001; Huynh 2009; Moakher) between RMs \( R_i \) and \( R_j \).
4.3 Robust 3D Rotation Distance

The proposed distance is designed to compare vectors of 3D rotations robustly. Inspired by the cosine-based distance between 2D rotations in eq. (4.10), a distance for comparing 3D rotations is developed in the following. Notice, the extension is non-trivial, as the concept of rotation axes is non-existent in 2D.

4.3.1 Proposed Distance

In 2D, a rotation $r$ is solely defined as its rotation angle $\theta$. In 3D, rotations are given as an angle-axis pair $r = (\theta, v) \in SO(3)$. Notice, the 3D equivalent of a 2D rotation by angle $\theta$ is expressed as $r = (\theta, [0 \ 0 \ 1]^T)$, i.e. all rotations happen around the $z$-axis. In the following, a new distance for general 3D rotations with arbitrary rotation axes is
proposed. Given two vectors of rotations, denoted by $\mathbf{x}_i = \begin{bmatrix} r_1^{(i)} & \cdots & r_p^{(i)} \end{bmatrix}^T \in \mathcal{SO}(3)^P$ and $\mathbf{x}_j$, I define the distance function

$$
d(\mathbf{x}_i, \mathbf{x}_j)^2 = \sum_{p=1}^P \left( 1 - \left( \frac{1 + \mathbf{v}_p^{(i)} \mathbf{v}_p^{(j)}}{2} \right) \cos(\theta_p^{(i)} - \theta_p^{(j)}) \right.
\left. - \left( \frac{1 - \mathbf{v}_p^{(i)} \mathbf{v}_p^{(j)}}{2} \right) \cos(\theta_p^{(i)} + \theta_p^{(j)}) \right).
$$

(4.17)

Note that $\left( \frac{1 + \mathbf{v}_p^{(i)} \mathbf{v}_p^{(j)}}{2} \right) + \left( \frac{1 - \mathbf{v}_p^{(i)} \mathbf{v}_p^{(j)}}{2} \right) = 1$, i.e., the terms act as weights that depend on the angle between the rotations’ unit axes. Figure 4.3 visualizes the weights’ properties. Let us consider two rotations, $r_i$ and $r_j$. If both share the same axis $\mathbf{v}_i = \mathbf{v}_j$, then $\mathbf{v}_i^T \mathbf{v}_j = 1$ and the distance turns into its 2D counterpart in eq. (4.10). In the case of opposing axes, $\mathbf{v}_i = -\mathbf{v}_j$, $\mathbf{v}_i^T \mathbf{v}_j = -1$, thus the cosine-based term with flipped rotation angle $\theta_j$ is considered. Notice the equivalence of $(\theta_j, \mathbf{v}_j) = (-\theta_j, \mathbf{v}_i)$. Hence, again the problem is reduced to the distance in eq. (4.10). A combination of both parts is employed in the case where $-1 < \mathbf{v}_i^T \mathbf{v}_j < 1$.

Let us compare the proposed 3D cosine-based distance to the squared Euclidean distance with different 3D rotation representations: HAQ, RM and RV (figure 4.4). When similar rotations are compared (figure 4.4(a)), the RV representation is sensitive to rotations with angles close to 180°, here the normalized distance may jump from near 0 to near 1. All other methods identify close rotations successfully. When comparing random rotations (figure 4.4(b)), RM and RV strongly bias the results either towards small or large distances. The values under HAQ and the proposed distance are more evenly distributed. The proposed distance shows similar properties to the distance under RM when applied to rotations with similar rotation axes (figure 4.4(c)). Here HAQ produces overall smaller distances. The distance under RV is quite unstable for this setup, as no trend is observed. However, when exposed to similar rotation angles (figure 4.4(d)), RV behaves similarly to the proposed distance. RM shows a bias towards large distances, while HAQ exhibits an even distribution of distances.

The proposed distance and HAQ are least biased for random samples. In particular, the maximum count of a single bin is less than 20%, and the mean of outliers, i.e., random samples, is near 0.5. The distance for inliers is close to 0. This corresponds to the robust properties of the cosine distance in 2D by Tzimiropoulos et al. (2012b). The proposed method differs mainly in its preference when compared to HAQ. Both favor similar axes, but HAQ does so more significantly. The proposed distance rates the strength of rotations more highly, i.e., the rotation angle or the amount of displacement when applied to a scene.

Chapter 4. Full-Angle Quaternion 95
Figure 4.4: Histograms of the normalized squared distances for (a) similar and (b) uniformly distributed random rotations, rotations with (c) similar axes and (d) similar angles, each resulting from $10^5$ 3D rotation comparisons. Random rotations are taken from the left Haar measure on $SO(3)$ (Haar 1933; Pennec and Ayache 1998).
4.3.2 Full-Angle Quaternion

The proposed distance in eq. (4.17) leads to a new representation for 3D rotations, which allows for efficient comparison. I coin it the full-angle quaternion (FAQ) representation.

Let us rewrite the squared distance as follows

\[
d(x_i, x_j)^2 = \sum_{p=1}^{P} \left( 1 - \cos \theta_p^{(i)} \cos \theta_p^{(j)} - (v_p^{(i)} T v_p^{(j)}) \sin \theta_p^{(i)} \sin \theta_p^{(j)} \right)
\]

\[
= \sum_{p=1}^{P} \left( \frac{\cos^2 \theta_p^{(i)} + \sin^2 \theta_p^{(i)} + \cos^2 \theta_p^{(j)} + \sin^2 \theta_p^{(j)}}{2} - \cos \theta_p^{(i)} \cos \theta_p^{(j)} - \sin \theta_p^{(i)} v_p^{(i)} T v_p^{(j)} \sin \theta_p^{(j)} \right)
\]

\[
= \frac{1}{2} \sum_{p=1}^{P} \left( (\cos \theta_p^{(i)} - \cos \theta_p^{(j)})^2 + \|v_p^{(i)} \sin \theta_p^{(i)} - v_p^{(j)} \sin \theta_p^{(j)} \|^2_F \right)
\]

\[
= \frac{1}{2} \sum_{p=1}^{P} \|q_p^{(i)} - q_p^{(j)}\|^2_F , \quad (4.18)
\]

where \( q \) is a unit quaternion given by:

\[
q = \cos \theta + (i v_x + j v_y + k v_z) \sin \theta . \quad (4.19)
\]

Eq. (4.19) defines the proposed FAQ representation. The key difference to the HAQ representation is that the trigonometric functions \( \cos(\cdot) \) and \( \sin(\cdot) \) are applied to the full-angle \( \theta \) instead of the half angle \( \frac{\theta}{2} \). Hence, it avoids the double covering issue of HAQ, as each 3D rotation corresponds to exactly one unit quaternion under FAQ. In addition, eq. (4.18) reveals that the proposed distance is equivalent to the Euclidean distance under the FAQ representation. In particular, given a vector of 3D rotations under FAQ as \( q = \left[ q_1 \, \cdots \, q_P \right]^T \in \mathbb{H}^P \), the robust distance on vectors of 3D rotations is given with proportional equality by

\[
d(x_i, x_j)^2 \propto \|q_i - q_j\|_F^2 . \quad (4.20)
\]

In contrast to the HAQ representation, which returns non-global means (Gramkow 2001), the mean of 3D rotations under FAQ is global and easy to compute as the normalized sum of FAQs.

Notice however, the FAQ representation comes with a degenerate case. Every 3D rotation by \( 180^\circ \) maps to the same unit quaternion, \( q = -1 \). This, however, does not
affect the computation of the related distance nor its proposed applications, as no inverse mapping is required.

Conversion Formulae for 3D Rotations

Given rotation \( r = (\theta, v) \in SO(3) \), I present the conversion formulae from the FAQ representation (eq. (4.19)) to the HAQ (eq. (4.8) and eq. (4.9)) and RM representation (eq. (4.2)). Let us denote the rotation’s version of FAQ, HAQ and RM as \( q_1 \in \mathbb{H} \), \( q_2 \in \mathbb{H} \) and \( \mathbf{R} \in \mathbb{R}^{3 \times 3} \) respectively. The notation \( \mathbf{R}(i, j) \) represents the rotation matrix’s entry at the \( i \)th row and \( j \)th column, \( a_i, b_i, c_i, d_i \in \mathbb{R} \) represent the elements of the quaternion \( q_i \), and \( \text{tr}(.) \) computes the trace of a matrix. The conversions are as follows:

1. Rotation matrix to full-angle quaternion:

\[
q_1 = \frac{1}{2} \left( (\text{tr} (\mathbf{R}) - 1) + i (\mathbf{R}(3, 2) - \mathbf{R}(2, 3)) + j (\mathbf{R}(1, 3) - \mathbf{R}(3, 1)) + k (\mathbf{R}(2, 1) - \mathbf{R}(1, 2)) \right) \tag{4.21}
\]

2. Full-angle quaternion to rotation matrix:

\[
\mathbf{R} = \mathbf{I} + \begin{bmatrix}
0 & -d_1 & c_1 \\
d_1 & 0 & -b_1 \\
-c_1 & b_1 & 0
\end{bmatrix} + \frac{1}{a_1 + 1} \begin{bmatrix}
-c_1^2 - d_1^2 & c_1 b_1 & d_1 b_1 \\
-b_1^2 - d_1^2 & c_1 b_2 & d_1 c_1 \\
-b_1^2 & c_1 d_1 & b_1 d_1
\end{bmatrix} \tag{4.22}
\]

3. Half-angle quaternion to full-angle quaternion:

\[
q_1 = 2a_2 (a_2 + i b_2 + j c_2 + k d_2) - 1 \tag{4.23}
\]

4. Full-angle quaternion to half-angle quaternion:

\[
q_2 = \frac{1}{\pm \sqrt{2(a_1 + 1)}} ((a_1 + 1) + i b_1 + j c_1 + k d_1) \tag{4.24}
\]

4.4 Applications of Full-Angle Quaternions

The FAQ representation of 3D rotations is applied to two common tasks. In collaboration with Toshiba a method for concurrent hashing of scale and translation is introduced by Liwicki et al. (2014). I employ this framework to generate vectors of 3D rotations for shape recognition with FAQs. For tracking, I develop a setup, which translates color and gradient directions into a vector of rotations.
4.4.1 Matching Ball Features for 3D Recognition

In this section, the proposed distance is applied to the task of 3D shape recognition from unstructured point clouds. Commonly used for this problem are local features, which represent the appearances of an object or a scene (Chen and Bhanu 2007; Knopp et al. 2010; Pham et al. 2011; Tombari and Di Stefano 2010; Woodford et al. 2014). Figure 4.5 illustrates such methods. First, features in the form of 3D oriented balls are localized in the input scene using a standard multi-scale keypoint detector, similar to SIFT-3D (Flitton et al. 2010), and already provided by the dataset of Pham et al. (2011). Alternatively, SURF-3D (Knopp et al. 2010) or ISS (Zhong 2009) can be used. Then, 3D descriptors are extracted to describe the area of each oriented ball feature. Some popular descriptors are spin images (Johnson and Hebert 1999), LSP (Chen and Bhanu 2007) or BOARD (Petrelli and Di Stefano 2011). At test time, the extracted scene features are matched with features from training data by comparing their descriptions, generating an initial set of votes. These votes are hypotheses of the object’s identity and pose, consisting of a position and an orientation (Knopp et al. 2010; Tombari and Di Stefano 2010), and additionally a scale if the scale is not assumed to be fixed (Pham et al. 2011; Woodford et al. 2014). The hypothesis of the best vote is then selected as an estimate of pose and identity.

Real objects often have repetitive parts which map to the same descriptor, thus generating many votes. To select the best vote, existing methods either (i) group votes according to object identity and find the pose with most neighbors in each group (Knopp et al. 2010; Pham et al. 2011; Tombari and Di Stefano 2010; Woodford et al. 2014), or (ii) evaluate the votes by aligning the predicted objects’ surface to the scene (Chen and Bhanu 2007; Johnson and Hebert 1999; Mian et al. 2006). While the former approaches are fast, they often fail if the ground truth pose has too few neighbors. The latter frameworks are more robust, but finding the corresponding scene surface for each predicted object surface is computationally costly.

In this work, an alternative method for evaluating votes is proposed. Instead of aligning surfaces between the scene and the predicted objects, their features are aligned. Here I assume that, if a predicted object pose is close to its true pose, the scene feature locations and orientations match those of the predicted object’s training data when back-projected into the scene. Since the number of feature-to-feature correspondence is smaller than the number of point-to-point correspondences, matching is expected to be faster than methods using surfaces.

The general case of 3D recognition is considered in the following. In particular, object scale is assumed to be unknown, and object poses and their ball features are treated as
Figure 4.5: Object recognition based on oriented ball features. Interest features (b) are extracted from the input point cloud (a). Each scene feature (green) is matched with training features (gray), generating a number of votes (c). After inference or vote validation, the final hypothesized object pose and identity is returned (d).

direct similarities (Coxeter 1961). For notational convenience, let us denote by $S_s$, $S_r$, and $S_t$ the scale, rotation and translation part respectively of a direct similarity $S$, such that

$$S = \begin{bmatrix} S_s S_R & S_t \\ 0 & 1 \end{bmatrix},$$

(4.25)

where $S_R$ denotes the RM representation of $S_r$. The proposed method for object recognition and registration consists of two phases.

In the offline phase (algorithm 4.1), for each object all feature locations that occur in the training data are collected, and normalized via left-multiplication with their corresponding object pose’s inverse. The normalized feature locations of each object are then stored in a single hash table in which hash keys are computed based on the scale and translation components using the concurrent hashing function, proposed in collaboration.
Algorithm 4.1 Offline Phase: Creating Hash Tables

**Input:** The training features \( \mathcal{F} \) and their poses \( \mathcal{P} \) as multi-index lists, such that \( \mathcal{F}_{i,j,k} \) denotes the \( i \)th object’s \( j \)th training instance’s \( k \)th feature location, and \( \mathcal{P}_{i,j} \) denotes the \( i \)th object’s \( j \)th training instance’s pose.

**Output:** The list of hash tables \( \mathcal{T} \), where \( \mathcal{T}_i \) corresponds to the table generated from the instances of the \( i \)th object in the training samples.

1: for \( i \)th object do
   2: Create hash table \( \mathcal{T}_i \).
   3: for \( j \)th training instance of the \( i \)th object do
      4: for \( k \)th feature of the \( j \)th training instance do
         5: Extract normalized similarity \( S = \mathcal{P}_{i,j}^{-1} \mathcal{F}_{i,j,k} \).
         6: Find or insert hash entry \( E = \mathcal{T}_i(h(S)) \), where \( h(.) \) is a hashing function on \( S \).
         7: Add the similarity’s rotation to entry \( E = E \cup \{S_r\} \).
      8: end for
   9: end for
10: end for

with Toshiba (Liwicki et al. 2014). The content of a hash entry is the set of rotations of all normalized locations hashed to it.

In the online phase (algorithm 4.2), the evaluated space for each vote is restricted to the 3D ball features observed in the scene and a vector of 3D rotations is produced. Then, for each vote that is generated by the feature descriptor comparison, all scene feature locations are left-multiplied by the inverse of the vote’s hypothesized pose. Finally, the hash entry (via the hash table of the predicted object) of each transformed feature location is considered, and the nearest rotation is found. Thus, the vector of scene features, and their rotations in particular, are compared to the training data.

Notice, the proposed method for evaluating the votes does not involve any feature descriptions, as only poses are required. Therefore, it exploits the geometry of an object as a whole, and not the geometry of local features. I detail the applied hashing function, and the rotation comparison in the following.

Hashing Dilatations

A concurrent hashing function for scale and transformation is proposed in collaboration with Toshiba by Liwicki et al. (2014). In this section, I summarize its properties. The scale and translation parts of a direct similarity, \( s \) and \( t \) form a transformation called (direct) dilatation (Coxeter 1961) in the space

\[
\mathcal{DT}(3) = \left\{ \begin{bmatrix} s \mathbf{I} & t \\ 0 & 1 \end{bmatrix}, s \in \mathbb{R}_+, t \in \mathbb{R}^3 \right\}.
\]  

(4.26)
Algorithm 4.2 Online Phase: Vote Evaluation

Input: The list of hash tables $\mathcal{T}$, where $\mathcal{T}_i$ corresponds to the $i^{th}$ object, scene feature locations $\{S_n\}_{n=1}^N$, and vote with object identity $i$ and pose $\mathcal{P}$.

Output: Weight $w$, which reflects the value of the vote (higher is better).

1: Initialize $w = 0$.
2: for $n^{th}$ scene feature do
3: Extract normalized similarity $S = \mathcal{P}^{-1} S_n$.
4: Find hash entry $E = \mathcal{T}_i(h(S))$, where $h(.)$ is a hashing function on $S$.
5: if $E$ is not empty then
6: Add weight $w = w + 4 - \min_{r \in E} d(r, S_r)^2$.
7: end if
8: end for

Given a direct similarity $S$, Liwicki et al. (2014) map its dilatation part, denoted $S_D = \begin{bmatrix} S_s I & S_t \\ 0 & 1 \end{bmatrix}$, to a 4-dimensional point via the mapping $\vartheta : \mathcal{DT}(3) \rightarrow \mathbb{R}^4$ provided by

$$\vartheta(S_D) = \begin{bmatrix} \ln S_s \\ \frac{1}{S_s} S_t \end{bmatrix}.$$  \hspace{1cm} (4.27)

The resulting 4-dimensional point, denoted $S_v$, is then quantized to form a 4-dimensional integer vector, i.e. a hash key, via a quantizer $\eta : \mathbb{R}^4 \rightarrow \mathbb{Z}^4$, given by

$$\eta(S_v) = \begin{bmatrix} S_v(1)_{\sigma_s} \\ S_v(2)_{\sigma_t} \\ S_v(3)_{\sigma_t} \\ S_v(4)_{\sigma_t} \end{bmatrix}^T,$$  \hspace{1cm} (4.28)

where $\sigma_s$ and $\sigma_t$ are parameters that enable trade-offs between scale and translation, and operator $\lfloor \cdot \rfloor$ is the floor function that finds the integer part of a real number.

Note, an efficient hash table should ensure that every hash entry is accessed with similar probability so that collisions are minimized. Liwicki et al. (2014) prove that if dilatations in $\mathcal{DT}(3)$ are uniformly distributed, i.e. distributed by a (left-)Haar measure (Haar 1933; Pennec and Ayache 1998), their coordinates via $\vartheta$ are uniformly distributed in $\mathbb{R}^4$, and vice versa. Combining this fact with the fact that the quantizer $\eta$ partitions $\mathbb{R}^4$ into equally sized portions, Liwicki et al. (2014) have proposed a hashing table with concurrent hash keys.

The hashing function allows for efficient storage and retrieval of the feature locations, and in particular their similarities’ rotations. Consequently, the hashing function $h(\cdot)$ in algorithm 4.1 and algorithm 4.2 is defined as $h(S) = \eta \circ \vartheta(S_D)$. 

102 4.4. Applications of Full-Angle Quaternions
Comparing 3D Rotations for Object Recognition

Unlike the general case of robustly matching rotations where both inputs can be noisy, the rotation of a training feature is arguably an inlier (i.e. an expected value) since the training data is often clean, especially when generated from a 3D object model. Thus, the proposed method mostly compares rotations from the scene with inliers. To exploit this fact, apart from using eq. (4.17), its left-invariant version is proposed to be used. As mentioned by Pennec and Ayache (1998), a left-invariant version $d'$ of a distance function $d$ is given by

$$d'(r_i, r_j)^2 = d(I, r_i^{-1}r_j)^2,$$  \hfill (4.29)

where $I$ is a rotation with angle $0$, $r_i$ is the rotation of a training feature and $r_j$ is a rotation from a scene feature. The notation $r_i^{-1}r_j$ represents the application of $r_i$'s inverse rotation to $r_j$.

Interestingly, the left-invariant version of the proposed distance via the FAQ representation is equivalent to the Euclidean distance under the RM representation, as the following equality holds

$$\frac{1}{2} \| R_i - R_j \|^2_F = (1 - \cos \theta_\delta)^2 + (\sin \theta_\delta)^2$$  \hfill (4.30)

$$= (1 - \cos \theta_\delta)^2 + \| 0 - v_\delta \sin \theta_\delta \|^2$$  \hfill (4.31)

$$= \| \text{faq}(I) - \text{faq}(R_i^{-1}R_j) \|^2 = d'(r_i, r_j)^2,$$  \hfill (4.32)

where $\theta_\delta$ and $v_\delta$ are respectively the angle and axis of $r_i^{-1}r_j$, eq. (4.30) is taken from Hartley et al. (2013), and $\text{faq}(:)$ denotes the FAQ representation of a rotation matrix.

4.4.2 Object Tracking in Color Video

The FAQ representation is applied to the task of object tracking in this section. In the following, I discuss the feature representation, the appearance model acquisition and the tracking framework.

3D Rotation Features for 2D Color Images

With robustness in mind, a lighting-invariant color representation is designed for an image region. Due to their inherent structure, composed of three imaginary bases, quaternions have been employed to represent the three dimensions of red-green-blue (RGB) values in numerous applications. For example, Le Bihan and Sangwine (2003) use a pure quaternion $q_1 \in \mathbb{H}$ to represent color as

$$q_1 = ir + jg + kb,$$  \hfill (4.33)
where \( r, g \) and \( b \) are RGB color values respectively. Alternatively, a normalized quaternion \( q_2 \) can be adopted

\[
q_2 = \frac{ir + jg + kb}{\sqrt{r^2 + g^2 + b^2}}.
\]  

(4.34)

Note, research by van de Sande et al. (2010) indicate that normalized RGB values have some robustness towards lighting and cast shadows. Soriano et al. (2000) have shown that normalization is particularly suitable for human skin color under changing illumination.

While color presents valuable cues, the method by Tzimiropoulos et al. (2012b) employs image gradient directions, which were found to be particularly useful in their experiments. Let us also utilize gradient orientations in the proposed representation. In particular, the 3D rotation features of color images are formulated as the gradient orientation \( \theta \) around the axis \( v \), given by the normalized RGB color values. Therefore the features are expressed as the following FAQ

\[
q_3 = \cos \theta + \frac{ir + jg + kb}{\sqrt{r^2 + g^2 + b^2}} \sin \theta
\]

(4.35)

with gradient orientation \( \theta \in [-\pi, \pi] \).

### Adaptive Tracking with Full-Angle Quaternions

Again, an explicit mapping, in this case the rotation representation under FAQs, is applied to the task of object tracking. In contrast to the work in chapter 3, however, color is now incorporated. Note also that the incremental PCA is required to handle quaternions instead of complex numbers.

As emphasized throughout this thesis, the acquisition of online-learned appearance models in contrast to a priori learned models is considered advantageous for robust tracking in unknown scenes (Babenko et al. 2011). The explicitly computed FAQ representation in eq. (4.19) allows for the adoption of an incremental PCA strategy for online learning while utilizing the proposed robust distance in eq. (4.17) for matching 3D rotations. In particular, the vectorized pixel intensity values of an image are replaced with quaternions, and the update is formulated as an online appearance model using the tools of quaternion PCA (Le Bihan and Mars 2004) and incremental PCA (Levy and Lindenbaum 2000; Ross et al. 2008), as discussed in the following.

The strategy of Ross et al. (2008) for tracking with incremental PCA is applied. As values are quaternion numbers, a small alteration to the setup in chapter 2 is required. Let us denote by \( Q = \left[ q_1^{(0)} \ldots q_N^{(0)} \right] \in \mathbb{H}^{P \times N} \) the initial sample matrix of \( N \) \( P \)-dimensional images, mapped into vectors using the FAQ representation of eq. (4.35). To find the SVD of the samples, \( i.e. \) \( Q = U\Sigma V^H \) where \( \Sigma \in \mathbb{R}^{R \times R} \) and \( U \in \mathbb{H}^{P \times R} \) contains
the eigenvalues and subspace of $Q$ respectively, the quaternion SVD of Le Bihan and Mars (2004) is applied. In particular, Le Bihan and Mars (2004) first find the complex adjoint matrix

$$C = \begin{bmatrix} A & B \\ -\overline{B} & \overline{A} \end{bmatrix} \in \mathbb{C}^{2P \times 2N}, \quad (4.36)$$

where the quaternion samples follow the Charley-Dickson notation $Q = A + iB$ and $A, B \in \mathbb{C}^{P \times N}$ are complex matrices. The component analysis of $C$ yields the following structure

$$C = \tilde{U} \tilde{\Sigma} \tilde{V}^H, \quad (4.37)$$

where $\tilde{\Sigma} \in \mathbb{R}^{2R \times 2R}$ contains twice as many eigenvalues as $\Sigma$. The eigenvectors $\{\tilde{u}_r\}_{r=1}^{2R}$ of subspace $\tilde{U} \in \mathbb{C}^{2P \times 2R}$ exhibit a special format. In particular, the Charley-Dickson notation of the eigenvectors in the quaternions’ subspace, i.e. $\{u_r = a_r + i b_r\}_{r=1}^R$, is extracted via

$$\tilde{u}_{2r-1} = \begin{bmatrix} a_r \\ -\overline{b_r} \end{bmatrix}. \quad (4.38)$$

The eigenvalues $\{\lambda_r\}_{r=1}^R$ in $\Sigma$ are provided by

$$\tilde{\Sigma} = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \lambda_R & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \lambda_R \end{bmatrix}. \quad (4.39)$$

The complete computation of the incremental PCA with the proposed FAQ representation is given in algorithm 4.3 and algorithm 4.4.

Finally, the online PCA of quaternions is combined with a particle filter to calculate the posterior of the system’s state based on a transition model and an observation model. This approach is introduced in chapter 2 and previously used in chapter 3. In short, the transition model is described as a GMM around an approximation of the state posterior distribution of the previous time-step. Given an independent covariance matrix, which represents the variance in horizontal and vertical displacement, rotation, scale, ratio and skew, $M$ particles, i.e. image patches, are extracted and their feature vectors of rotations...
Algorithm 4.3 QUATERNION SUBSPACE ESTIMATION

**Input:** A set of $N$ images $\{I_n\}_{n=1}^N$ with $P$ pixels in each image and the number $R$ of principal components.

**Output:** The principal subspace $U$ and eigenvalues $\Sigma$ of the samples.

1. Represent the pixel values of $I_n$ in the range of $[0, 1]$ and obtain vector $q_n \in \mathbb{H}^P$ by applying the mapping in eq. (4.35) to the pixels in $I_n$.
2. Form the matrix $Q = [\begin{array}{c} q_1 \cdots q_N \end{array}] = A + iB$ and its complex adjoint matrix $C = \begin{bmatrix} A & B \\ -\overline{B} & \overline{A} \end{bmatrix}$.
3. Compute the complex SVD $C^\text{svd} \equiv \hat{U}\hat{\Sigma}\hat{V}^H$ and form the $2R$-reduced eigenspectrum $\bar{U}_{2R} \in \mathbb{C}^{2P \times 2R}$, $\bar{\Sigma}_{2R} \in \mathbb{R}^{2R \times 2R}$.
4. Set the eigenvectors $A_r = a_r + \overline{i}b_r$ of the quaternion’s subspace $\bar{U}_{2R}$ via the vectors $\{\hat{u}_{2r-1} = \begin{bmatrix} a_r \\ -\overline{b_r} \end{bmatrix}\}_{r=1}^R$ of $\bar{U}_{2R}$.
5. Find the eigenvalues $\lambda_r$ of the reduced eigenspectrum $\bar{\Sigma}_{2R}$ through the eigenvalues $\{\hat{\lambda}_r\}_{r=1}^R$ of $\hat{\Sigma}_{2R}$.

Algorithm 4.4 INCREMENTAL QUATERNION SUBSPACE ESTIMATION

**Input:** The principal subspace $U_{t-1} \in \mathbb{H}^{P \times R}$, the corresponding eigenvalues $\Sigma_{t-1} \in \mathbb{R}^{R \times R}$, a set of new images $\{I_n\}_{n=N_{t-1}+N_t}$, and the number $R$ of principal components.

**Output:** The new subspace $U_t$ and eigenvalues $\Sigma_t$.

1. From image set $\{I_n\}_{n=N_{t-1}+N_t}$ compute the matrix of the transformed data $Q_\delta = [\begin{array}{c} q_{N_{t-1}+1} \cdots q_{N_{t-1}+N_t} \end{array}]$ via the mapping in eq. (4.35).
2. Compute $U_\delta = \text{orth}(Q_\delta - U_{t-1}U_{t-1}^HQ_\delta)$.
3. Form $L = [\begin{array}{c} \Sigma_{t-1}^{-1} U_{t-1}^HQ_\delta \\ 0 \end{array}] = A + iB$ and its adjoint matrix $C = \begin{bmatrix} A & B \\ -\overline{B} & \overline{A} \end{bmatrix}$.
4. Find $C^\text{svd} \equiv \bar{U}\bar{\Sigma}\bar{V}^H$ and the $2R$-reduced eigenspectrum $\bar{U}_{2R} \in \mathbb{C}^{2P \times 2R}$, $\bar{\Sigma}_{2R} \in \mathbb{R}^{2R \times 2R}$.
5. From $\bar{U}_{2R}$ and $\bar{\Sigma}_{2R}$ extract the reduced eigenspectrum of $L$ as $\hat{U}_R$ and $\hat{\Sigma}_R$.
6. Set $U_t = \begin{bmatrix} U_{t-1} & U_\delta \end{bmatrix} \hat{U}_R$ and $\Sigma_t = \hat{\Sigma}_R$.

Given by eq. (4.35), is created. Then the observation model is applied to the extracted image features to find the best match, and to initialize the next GMM for the next frame of the video sequence. The observation model computes the probability of a sample being generated by the learned eigenspectrum in the appearance model, which is assumed to be proportional to

$$e^{-\gamma \|q_m^{(t)} - U_tU_t^Hq_m^{(t)}\|^2_F}$$

where $q_m^{(t)}$ is the FAQ representation vector of the tested particle at time $t$, $U_t$ is the current subspace, and $\gamma$ is a parameter that controls the spread. The update of the appearance model is performed with the best matching particle.
\begin{table}[h]
\centering
\begin{tabular}{|l|c|}
\hline
 & Weight \\
\hline
CNT & 1 \\
HAQ & $4 - \min_{R \in \mathcal{E}} \| \text{haq}(R) \pm \text{haq} (S_R) \|^2$ \\
RV & $4\pi^2 - \min_{R \in \mathcal{E}} \| \text{rv}(R) - \text{rv}(S_R) \|^2$ \\
LI-RV & $\pi^2 - \min_{R \in \mathcal{E}} \| \text{rv}(R^{-1}S_R) \|^2$ \\
FAQ & $4 - \min_{R \in \mathcal{E}} \| \text{faq}(R) - \text{faq}(S_R) \|^2$ \\
LI-FAQ & $4 - \min_{R \in \mathcal{E}} \| \text{faq}(I) - \text{faq}(R^{-1}S_R) \|^2$ \\
\hline
\end{tabular}
\caption{Weighting strategies for different methods. Functions $\text{haq} (\cdot)$, $\text{rv} (\cdot)$, $\text{faq} (\cdot)$ form the HAQ, RV and FAQ representations of a 3D rotation matrix.}
\end{table}

4.5 Experiments

The proposed FAQ representation of 3D rotations is evaluated on 3D shape recognition and 2D object tracking in color video.

4.5.1 3D Object Recognition and Registration

I evaluate the method in section 4.4.1 on 3D object recognition. The experiments employ the public Toshiba CAD model point clouds dataset by Pham et al. (2011). This dataset consists of 1000 test sets of votes, each computed from a point cloud containing a single rigid object – one of 10 test objects (table 4.2). Training and test features are provided.

The proposed method, using the left-invariant distance induced by FAQs, is compared to five variants. These methods differ in line 6 of algorithm 4.2, where different weighting strategies corresponding to different distances are adopted. Table 4.1 summarizes the setups. HAQ is based on the distance between HAQs. RV and LI-RV employ the distance between RVs, in its standard and left-invariant format. The proposed distance between FAQs is handled by FAQ and LI-FAQ. CNT is purely based on the count of matching dilatations, and thus is used as a baseline method. As CNT is rotation independent, I use this setup for finding $\sigma_s$ and $\sigma_t$ in eq. (4.28). The minimum-entropy Hough transform (Hough) by Woodford et al. (2014) is considered the state-of-the-art.

To optimize the values for $\sigma_s$ and $\sigma_t$, a grid search using leave-one-out cross validation is employed, similar to Pham et al. (2011); Woodford et al. (2014). I maximize the recognition rate, followed by the registration rate. In particular, the recognition rate is the rate of instances where the best vote’s hypothesized identity of the object corresponds to the ground truth. The registration rate is evaluated as a binary score on the displacement error of the best vote with correct object identity. In detail, this is computed as follows. Let $P$ be the predicted pose of the highest scoring vote with correct object identity. Following Pham et al. (2011) and given the true pose $G$, the following criterions need to be
<table>
<thead>
<tr>
<th>Method</th>
<th>Hough</th>
<th>CNT</th>
<th>HAQ</th>
<th>RV</th>
<th>LI-RV</th>
<th>FAQ</th>
<th>LI-FAQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Rate</td>
<td>79.6%</td>
<td>86.7%</td>
<td>87.7%</td>
<td>87.3%</td>
<td>87.7%</td>
<td>87.7%</td>
<td>87.9%</td>
</tr>
</tbody>
</table>

**Table 4.2:** Object level registration rates of all methods. Bold values indicate best results.
Table 4.3: Summary of the results for object recognition. Bold values indicate the best achieved results across all methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recognition Rate</th>
<th>Registration Rate</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hough</td>
<td>98.5%</td>
<td>79.6%</td>
<td>0.214s</td>
</tr>
<tr>
<td>CNT</td>
<td><strong>100.0%</strong></td>
<td>86.7%</td>
<td><strong>0.092s</strong></td>
</tr>
<tr>
<td>HAQ</td>
<td>100.0%</td>
<td>87.7%</td>
<td>0.103s</td>
</tr>
<tr>
<td>RV</td>
<td>100.0%</td>
<td>87.3%</td>
<td>0.117s</td>
</tr>
<tr>
<td>LI-RV</td>
<td>100.0%</td>
<td>87.7%</td>
<td>0.106s</td>
</tr>
<tr>
<td>FAQ</td>
<td>100.0%</td>
<td>87.7%</td>
<td>0.097s</td>
</tr>
<tr>
<td>LI-FAQ</td>
<td>100.0%</td>
<td><strong>87.9%</strong></td>
<td><strong>0.095s</strong></td>
</tr>
</tbody>
</table>

must hold. The best result for CNT is found at \((\sigma_s, \sigma_t) = (0.111, 0.92)\) where the recognition rate is 100.0% and the registration rate is 86.7%. This method alone outperforms the best-performing method on this dataset to date, i.e. Hough, which only achieves a recognition rate of 98.5% and a registration rate of 79.5%. It also produces a perfect recognition rate, leaving no room for improvement in this aspect.

I perform cross validation over the other five variants using the same values for \((\sigma_s, \sigma_t)\), to produce comparable results. Table 4.3 shows the performance of all methods. Generally, it is difficult to improve significantly upon the performance of CNT. A 100.0% recognition rate and a slightly higher registration rate than that of CNT is obtained in all cases. LI-FAQ gives the best registration rate, followed by HAQ, LI-RV and FAQ, and then RV. Notice, the left-invariant distances induced by the RV and FAQ representation outperform their non-invariant counterparts respectively. Detailed registration results at object level are shown in table 4.2.

Unlike existing vote-evaluation methods (Chen and Bhanu 2007; Johnson and Hebert 1999; Mian et al. 2006), the evaluation time per test set (10,000 votes on average) of the proposed methods are even faster than that of the fast minimum-entropy Hough transform approach by Woodford et al. (2014). Among them, distances based on the FAQ representation have a slight advantage.
4.5.2 Object Tracking in Color Video

I now evaluate the 3D rotation representation in the tracking setup of section 4.4.2 on data with outliers caused by varying lighting, occlusions and appearance changes. The proposed system (FAQ) with quaternions as in eq. (4.35) is compared to the competitive tracker in chapter 3 (eT), and the original 2D distance of Tzimiropoulos et al. (2012b) (Gradients), both with gray scale images. I also include the results of the RGB quaternion (RGB-Quat) in eq. (4.33) and its normalized version (N-RGB-Quat) in eq. (4.34). Finally, I replace the FAQ representation in FAQ with the same rotation, but represented by HAQ and RM. Table 4.4 summarizes the setups. Tests are performed on the color versions of the videos previously used in chapter 3. Note, only Vid2, Vid3, Vid5 and Vid7 have
Feature Representation | PCA Components
---|---
eT $\cos(\alpha \pi i) + i \sin(\alpha \pi i)$ | 15
Gradient $\cos(\theta) + i \sin(\theta)$ | 35
RGB-Quat $ir + jg + kb$ | 20
N-RGB-Quat $\frac{ir+jg+kb}{\sqrt{r^2+g^2+b^2}}$ | 10
RM vectorized RM for $r = (\theta, \frac{1}{\sqrt{r^2+g^2+b^2}} \begin{bmatrix} r & g & b \end{bmatrix}^T)$ | 40
HAQ $\cos(\frac{\theta}{2}) + \frac{ir+jg+kb}{\sqrt{r^2+g^2+b^2}} \sin(\frac{\theta}{2})$ | 20
FAQ $\cos(\theta) + \frac{ir+jg+kb}{\sqrt{r^2+g^2+b^2}} \sin(\theta)$ | 40

Table 4.4: The different tracking setups. The pixel intensity is given by $i$, while color is provided by $r$, $g$ and $b$. The gradient angle is denoted $\theta$ and $\alpha = 1.9$ as in chapter 3.

<table>
<thead>
<tr>
<th>Vid2</th>
<th>Vid3</th>
<th>Vid5</th>
<th>Vid7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$eT$</td>
<td>(lost)</td>
<td>(lost)</td>
<td>2.23</td>
</tr>
<tr>
<td>Gradient</td>
<td><strong>2.42</strong></td>
<td>4.94</td>
<td>(lost)</td>
</tr>
<tr>
<td>RGB-Quat</td>
<td>(lost)</td>
<td>(lost)</td>
<td>2.22</td>
</tr>
<tr>
<td>N-RGB-Quat</td>
<td>(lost)</td>
<td>(lost)</td>
<td><strong>1.79</strong></td>
</tr>
<tr>
<td>RM</td>
<td>2.48</td>
<td>2.52</td>
<td>(lost)</td>
</tr>
<tr>
<td>HAQ</td>
<td>2.90</td>
<td>2.93</td>
<td>(lost)</td>
</tr>
<tr>
<td>FAQ</td>
<td>2.55</td>
<td><strong>2.52</strong></td>
<td>(lost)</td>
</tr>
</tbody>
</table>

Table 4.5: Mean RMS error for all videos and trackers. Missing results indicate failed tracking, evaluated by inspection.

color versions, and are provided by Birchfield (1998) and Ross et al. (2008) (figure 4.6). In contrast to chapter 3, where video specific parameters for the particle filter are found, in this chapter, I fix the covariance to translation $x = 0.11$, $y = 0.09$, $scale = 1.4$, $rotation = 0.02$, $ratio = 0.002$ and $skew = 0.001$ for all videos and methods. Online learning is performed with optimal component numbers for each setup (see table 4.4). Again, 800 particles are extracted at each frame and a batch of five samples is used for the update. The tracking performance is evaluated based on accuracy (figure 4.7), i.e. RMS error between predicted and true landmarks, and precision (figure 4.8). Table 4.5 lists the mean RMS error for all videos and methods.

Gradient, RM, HAQ and FAQ track the target in Vid2 successfully, while the other methods which build upon color and intensity alone fail to track the video due to cast shadows and large pose variations. Here, gradients are an advantage. In Vid3, the rotation feature-based frameworks (RM, HAQ and FAQ) again outperform the other systems. N-RGB-Quat is able to track most of the sequence, but fails during the appearance change in frame 390. All other systems struggle during the motion blur and large pose variation around frame 159. The most challenging video is Vid5, in which the target performs two $360^\circ$ turns. FAQ is the only system, among the ones compared, to succeed in tracking
Figure 4.7: RMS error of each frame for Vid2, Vid3, Vid5 and Vid7 (top to bottom). Gaps indicate occlusions of the tracked object.
these appearance changes. The combination of robustness to varying appearance and the color and gradient cues make the tracking for FAQ possible. The final video Vid7 is a night-time recording of a rigid object in low light intensity. Such scene is considered easy to track, and all systems succeed. The low light however slightly reduces the performance of RM, HAQ and FAQ, as the color values for the rotation axes are less reliable.

With regards to precision (figure 4.8), the group of gradient-based methods (Gradient, RM, HAQ and FAQ) performs generally well, while non-gradient-based methods (eT, RGB–Quat, N–RGB–Quat) are less precise. Furthermore, systems with color and gradient outperform other methods for most videos – Vid7 is an exception, as the target is less difficult to track and the low lighting causes reduced performance with 3D rotation features.

In general, FAQ largely improves upon its 2D version, i.e. Gradient, and is among the best methods for Vid2, Vid3 and especially Vid5. In comparison to other 3D rotation
features, only RM performs similarly well in Vid2, Vid3 and Vid7 – note however, RM is slower as the dimensionality is increased by a factor of nine. Finally, I emphasize that FAQ’s unique representation of rotations is advantageous over HAQ’s dual mapping, as FAQ achieves higher precision. I conclude that the FAQ representation can be employed for fast and robust online subspace learning for object tracking.

4.6 Conclusions

In this chapter, a new distance measure for robustly matching vectors of 3D rotations is introduced. The distance leads to an efficient representation in which any 3D rotation maps uniquely to a unit quaternion. I coin this representation the full-angle quaternion (FAQ) representation of 3D rotations. Like in chapter 3, a robust PCA is formed through an explicit mapping, which facilitate fast and robust incremental PCA learning, while utilizing the proposed distance.

I apply the distance to 3D shape recognition and 2D tracking from color images. For the former, I incorporate the concurrent hashing function for dilatations proposed by Liwicki et al. (2014), in collaboration with Toshiba. After hashing the dilatation part of the similarities, rotations are compared using the proposed FAQ representation. Here, state-of-the-art recognition and registration results are achieved on a public dataset. In the application of 2D object tracking, online subspace learning is combined with the proposed FAQ representation to create a tracking system that tracks from color images.

In conclusion, this chapter reveals the importance of robust comparison of vectors composed of rotations. In particular, by combining gradient orientations of images with the pixels’ color values, promising performance improvements are achieved in the incremental tracking framework. In the next chapter, I will develop a domain-specific kernel function that is optimized to work with image intensity gradient orientations and their magnitudes.
Exploiting domain-specific knowledge, such as the advantages of gradient orientations in image processing, has proven promising in chapter 4. Now, I introduce an online learning framework which employs a kernel that uses intensity gradient magnitudes and orientations to represent its comparing function. As the kernel is non-positive definite, this task involves the generalization of KPCA from a reproducing kernel Hilbert space (RKHS) to Krein space, and the formulation of an incremental KPCA that exploits the special properties of the proposed kernel.

As outlined in chapter 2, one of the major challenges in online learning with kernels is the continuous growing of the support set during execution. In particular, as the online classification or regression function is typically written as a weighted sum of kernel combinations of samples, the set of all previously seen instances is required. This set is usually referred to as support or reduced set, and it grows at each time-step, as new instances are fed to the algorithm.

Although many methods have been proposed for online kernel learning for classification and regression, limited research has been conducted for online subspace learning with kernels. A summary of existing methods can be found in chapter 2. The most relevant work to the method in the current chapter is that by Chin and Suter (2007), which introduces an incremental kernel PCA (KPCA) algorithm with Hilbert spaces, that kernelizes the exact algorithm for incremental PCA by Levy and Lindenbaum (2000); Ross et al. (2008). In their work, in order to maintain constant update speeds, Chin and Suter (2007) introduce preimages of the kernel principal components and the mean. The main drawbacks of this method are, however, that (i) the reduced set representation provides only an approximation to the exact solution, and (ii) the proposed optimization problem for finding the expansions inevitably increases the complexity of their approach.

In this chapter, an exact framework for online learning with a special family of indefinite (i.e. non positive definite) kernels is introduced. As the case of non positive definite
kernels is studied, I first extend the general theory of KPCA from a RKHS to Krein space. Then an incremental KPCA in Krein space is formulated for a special type of kernels. As in previous chapters, the calculation of the proposed online version with the special property of the kernels does not require the computation of preimages and is therefore both efficient and exact.

Chapter 3 shows the disadvantages of using off-the-shelf kernels (such as the GRBFs), which do not incorporate any problem-specific prior knowledge for visual tracking, as such methods result in loss of robustness and accuracy. In chapter 4 the advantage of distance functions which combine gradient orientations with the robust sum of cosines of angle differences is seen. Motivated by applications in computer vision, I now wish to employ a robust gradient-based kernel and apply its non-linear appearance model, learned via KPCA in Krein space, for visual tracking and robust face recognition. The evaluation is performed on a number of popular and difficult tracking scenarios, and face recognition, where the proposed method outperforms KPCA with a GRBF kernel and standard \( \ell_2 \)-norm PCA.

In summary, the contributions in this chapter are as follows:

1. A robust non-positive definite kernel for measuring visual similarity is designed.

2. General kernel PCA is formulated in Krein space.

3. An accurate incremental KPCA in Krein space is proposed. This method exploits special properties of the employed kernel function, and does not require a reduced set representation.

4. The proposed learning framework is applied to visual tracking, for which very competitive performance is achieved.

5. The proposed KPCA is also applied to face recognition, where better class separation properties are seen, in comparison to KPCA in Hilbert space with a GRBF kernel, and standard \( \ell_2 \)-norm PCA.

One of the works, closely related to that proposed here, is presented by Pękalska and Haasdonk (2009). I would like to highlight that Pękalska and Haasdonk (2009) propose two-class classifiers based on a quadratic discriminant function in both Hilbert and Krein spaces. In this chapter, I take a different direction. In particular, a subspace learning algorithm in Krein space for feature extraction and object representation is proposed.

The rest of this chapter is organized as follows. I summarize the theory of Krein spaces and compare them to Hilbert spaces in section 5.1. The proposed domain-specific kernel is introduced in section 5.2. In section 5.3 I propose KPCA in Krein space and present
the direct incremental update of the non-linear subspace which exploits special properties of the kernel. The visual tracker is introduced in section 5.4, and section 5.5 presents experimental results. Section 5.6 concludes this chapter.

5.1 Kernel Functions in Krein Space

Recall from chapter 2, a Hilbert space $\mathcal{H}$ is a complete vector space which is defined by an inner product onto complex space $\langle ., . \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{C}$, which induces a norm and, thus, a metric. Generally, $\mathcal{H}$ is an implicitly defined infinite dimensional hyperspace. A kernel function $k_{\mathcal{H}} : \mathbb{C}^P \times \mathbb{C}^P \rightarrow \mathbb{C}$ defines the unknown mapping $\phi_{\mathcal{H}} : \mathbb{C}^P \rightarrow \mathcal{H}$ which transforms the original data into Hilbert space, and the inner product is realized by

$$\langle \phi_{\mathcal{H}}(z_j), \phi_{\mathcal{H}}(z_i) \rangle_{\mathcal{H}} = \phi_{\mathcal{H}}(z_i)^{\text{H}} \phi_{\mathcal{H}}(z_j) = k_{\mathcal{H}}(z_i, z_j) \quad (5.1)$$

where $z_i$ and $z_j$ are members of the original input space $\mathbb{C}^P$ and $()^\text{H}$ indicates the Hermitian transpose. For many applications, $k_{\mathcal{H}}$ is employed and $\phi_{\mathcal{H}}$ is never explicitly required (Böhmer et al. 2011; Schölkopf et al. 1997; Shawe-Taylor and Cristianini 2004). For example, the squared linear distance between two samples in the non-linear Hilbert space is given via the kernel as

$$d(\phi_{\mathcal{H}}(z_i), \phi_{\mathcal{H}}(z_j))^2 = k_{\mathcal{H}}(z_i, z_i) - k_{\mathcal{H}}(z_i, z_j) - k_{\mathcal{H}}(z_j, z_i) + k_{\mathcal{H}}(z_j, z_j). \quad (5.2)$$

The following properties are important for a Hilbert space ($\forall x_i, x_j, x_k \in \mathcal{H}, \forall a, b \in \mathbb{C}$):

$$\langle x_i, x_j \rangle_{\mathcal{H}} = \overline{\langle x_j, x_i \rangle_{\mathcal{H}}} \quad (5.3)$$
$$\langle ax_i + bx_j, x_k \rangle_{\mathcal{H}} = a\langle x_i, x_k \rangle_{\mathcal{H}} + b\langle x_j, x_k \rangle_{\mathcal{H}} \quad (5.4)$$
$$d(x_i, x_k) \leq d(x_i, x_j) + d(x_j, x_k) \quad (5.5)$$

where $\overline{(.)}$ denotes the complex conjugate. Therefore, if a positive definite kernel is selected, the related vector space is Hilbert.

Let us now discuss kernels in Krein space, denoted by $\mathcal{K}$. Similar to Hilbert space, kernels in Krein space define an implicit mapping $\phi_{\mathcal{K}} : \mathbb{C}^P \rightarrow \mathcal{K}$ from feature space onto $\mathcal{K}$, and provide the inner product $\langle ., . \rangle_{\mathcal{K}} : \mathcal{K} \times \mathcal{K} \rightarrow \mathbb{C}$ such that

$$\langle \phi_{\mathcal{K}}(z_j), \phi_{\mathcal{K}}(z_i) \rangle_{\mathcal{K}} = k_{\mathcal{K}}(z_i, z_j) \quad (5.6)$$

for input samples $z_i, z_j \in \mathbb{C}^P$. A Krein space, too, satisfies the analogous properties for eq. (5.3) and eq. (5.4). However, as a distance is given similarly to eq. (5.2), the triangular
inequality may not hold (Hassibi et al. 1996; Pękalska and Haasdonk 2009).

A Krein space is composed out of two Hilbert spaces, $\mathcal{K}_+$ and $\mathcal{K}_-$, such that $\mathcal{K} = \mathcal{K}_+ \oplus \mathcal{K}_-$, where $\oplus$ denotes the direct sum (i.e. $\mathcal{K}_+$ and $\mathcal{K}_-$ are orthogonal in terms of $\langle ., . \rangle_\mathcal{K}$). Thus, two orthogonal projections can be extracted from $\mathcal{K}$: $\mathbf{F}^+$ onto $\mathcal{K}_+$ and $\mathbf{F}^-$ onto $\mathcal{K}_-$, known as fundamental projections. Furthermore, the positive portion $\mathbf{x}^+ \in \mathcal{K}_+$ of sample $\mathbf{x} \in \mathcal{K}$ can be extracted via $\mathbf{x}^+ = \mathbf{F}^+ \mathbf{x}$, and similarly the negative part $\mathbf{x}^- \in \mathcal{K}_-$ via $\mathbf{x}^- = \mathbf{F}^- \mathbf{x}$. Notice, $\langle \mathbf{x}^+_j, \mathbf{x}^-_i \rangle_\mathcal{K} = 0$ for any two samples $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{K}$. Moreover, $\langle \mathbf{x}^+_j, \mathbf{x}^+_i \rangle_\mathcal{K} > 0$ and $\langle \mathbf{x}^-_j, \mathbf{x}^-_i \rangle_\mathcal{K} < 0$ for any non-zero vectors $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{K}$. Hence, the inner product of $\mathcal{K}$ is defined as the difference between the products of $\mathcal{K}_+$ and $\mathcal{K}_-$, respectively, and we may write

$$\langle \mathbf{x}_j, \mathbf{x}_i \rangle_\mathcal{K} = \langle \mathbf{x}^+_j, \mathbf{x}^+_i \rangle_{\mathcal{K}_+} - \langle \mathbf{x}^-_j, \mathbf{x}^-_i \rangle_{\mathcal{K}_-}.$$  \hfill (5.7)

By use of the so-called fundamental symmetry $\mathbf{J} = \mathbf{F}^+ - \mathbf{F}^-$, an associated Hilbert space, denoted $|\mathcal{K}|$, is found. The fundamental symmetry $\mathbf{J}$ satisfies $\mathbf{J}^3 = \mathbf{J}$. The relationship between $\mathcal{K}$ and $|\mathcal{K}|$ can be written in terms of a “conjugate” (denoted $(.)^*$), defined as

$$\mathbf{x}^*_i \mathbf{x}_j \triangleq \langle \mathbf{x}_j, \mathbf{x}_i \rangle_\mathcal{K} = \mathbf{x}^*_i \mathbf{J} \mathbf{x}_j = \langle \mathbf{J} \mathbf{x}_j, \mathbf{x}_i \rangle_{|\mathcal{K}|}$$  \hfill (5.8)

where $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{K}$. That is, $\mathcal{K}$ can be turned into its associated Hilbert space $|\mathcal{K}|$ by using its positive definite inner product $\langle ., . \rangle_{|\mathcal{K}|}$, as $\langle \mathbf{x}_j, \mathbf{x}_i \rangle_{|\mathcal{K}|} = \langle \mathbf{x}_j, \mathbf{J} \mathbf{x}_i \rangle_{\mathcal{K}}$.

In the following I am particularly interested in finite dimensional Krein spaces (i.e. $\dim(\mathcal{K}_) + \dim(\mathcal{K}_-) < \infty$, where $\dim(.)$ finds the dimensionality). Such a Krein space describes a pseudo-Euclidean space and is characterized by its so-called signature, i.e. $[\dim(\mathcal{K}_+) \; \dim(\mathcal{K}_-)]^T \in \mathbb{N}^2$, which indicates the dimensionality of the positive and negative subspaces (Pękalska and Haasdonk 2009). In such cases, the fundamental symmetry is given by

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_{\dim(\mathcal{K}_+)} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_{\dim(\mathcal{K}_-)} \end{bmatrix}$$  \hfill (5.9)

where $\mathbf{I}_a$ is the $a \times a$ identity matrix, and $\mathbf{0}$ implies zero padding.

Kernels in Hilbert or Krein spaces are important tools, as they provide feature representations for dissimilarities in non-linear spaces. Plentiful successful applications exist in the literature for Hilbert kernels (Böhmer et al. 2011; Schölkopf et al. 1997; Shawe-Taylor and Cristianini 2004). The use of Krein kernels is more seldom found (Pękalska and Haasdonk 2009). In the following I will propose a domain-specific indefinite kernel (i.e. a kernel that builds on image gradient features) with special properties, that allow for efficient incremental subspace learning techniques.

5.1. Kernel Functions in Krein Space
5.2 Proposed Indefinite Kernel for Computer Vision

In this section, I will propose an indefinite gradient-based kernel for image processing. I first detail its design in the following. I then introduce the special kernel properties that allow for an explicit representation which is equivalent to the kernel’s implicit feature space.

5.2.1 Robust Gradient-based Kernel Function

Off-the-shelf kernels (such as GRBF kernels), which do not incorporate any problem-specific prior knowledge to the domain of visual tracking, often result in loss of robustness and accuracy. To tackle this deficiency, an indefinite robust gradient-based kernel is employed. This kernel is inspired by Tzimiropoulos et al. (2010), who propose a scheme for the robust estimation of large translational displacements.

Their method works as follows. Let us assume an image $I$, with normalized pixel values in the range $[0, 1]$. The gradient-based representation of $I$ is defined as

$$G = F_h \ast I + iF_v \ast I,$$  \hspace{1cm} (5.10)

where $F_h$ and $F_v$ are linear filters which approximate the ideal differentiator in the image’s horizontal and vertical directions, and $\ast$ is the operator that applies a filter to an image. Let $z \in \mathbb{C}^P$ be the $P$-dimensional vector obtained by writing $G$ in lexicographical order. Tzimiropoulos et al. (2010) provide the standard gradient correlation coefficient between samples $z_i$ and $z_j$ by

$$s_1(z_i, z_j) = \text{Re} \left( z_i^H z_j \right) = \sum_{p=1}^{P} r_i(p) r_j(p) \cos(\theta_i(p) - \theta_j(p)), \hspace{1cm} (5.11)$$

where $\text{Re}(.)$ extracts the real value of a complex number, $r = R(z)$ is a vector containing the element-wise magnitudes of $z$, and $\theta = \angle(z)$ contains the element-wise angles of the complex number in $z$. Tzimiropoulos et al. (2010, 2011, 2012a) argue that the use of the cosine on the difference of gradient orientations induces a robustness to outliers. In particular, given an image area of outliers $P$, the sum of the corruptions cancel, as $\sum_{p\in P} \cos(\theta_i(p) - \theta_j(p)) \approx 0.$

A modification of this correlation is now proposed as a new kernel function. As shown by Tzimiropoulos et al. (2010), the gradient magnitudes are more sensitive to outliers. Thus it is very likely that their product (i.e. $r_i(p)r_j(p)$) will be affected most. One solution to circumvent this issue may be to remove the gradient magnitude from
I
i
G
i
I
j
G
j
r
j
= 1
r
i
= 1

Figure 5.1: Visualization of the proposed kernel when corruption is present. Notice, the corruption in image \( I_j \) largely only affects one term (where \( r_i = 1 \)), while the other retains some useful edge information.

eq. (5.11), as proposed in the learning framework of Tzimiropoulos et al. (2012a) or the alignment framework by Tzimiropoulos et al. (2011). Notice however, this results in loss of information, provided by the magnitudes. I take a different approach. In particular, a split of the correlation in eq. (5.11) is proposed. That is, by expressing the correlation with two terms, for which \( r_i = 1 \) for one term and \( r_j = 1 \) for the other, essentially replacing the product by a sum. The new dissimilarity measure is thus given by

\[
s_2(z_i, z_j) = \sum_{p=1}^{P} (r_i(p) + r_j(p)) \cos (\theta_i(p) - \theta_j(p))
\]

\[
= \sum_{p=1}^{P} r_i(p) \cos (\theta_i(p) - \theta_j(p)) + \sum_{p=1}^{P} r_j(p) \cos (\theta_i(p) - \theta_j(p)).
\]

(5.12)

Figure 5.1 shows how this approach reduces the effect of outliers to some extent while maintaining some information from the magnitudes. A normalized version of the above correlation is achieved via

\[
s_3(z_i, z_j) = \frac{\sum_{p=1}^{P} r_i(p) \cos (\theta_i(p) - \theta_j(p))}{2 \sqrt{\sum_{p=1}^{P} r_i^2(p) P}} + \frac{\sum_{p=1}^{P} r_j(p) \cos (\theta_i(p) - \theta_j(p))}{2 \sqrt{\sum_{p=1}^{P} r_j^2(p) P}}.
\]

(5.13)

For this, a detailed derivation can be found in appendix A. Finally, the complex equivalent

120 5.2. Proposed Indefinite Kernel for Computer Vision
(in terms of induced distance) forms the proposed kernel function \( k \) as

\[
k(z_i, z_j) = \frac{\sum_{p=1}^{P} r_i(p) (\cos (\theta_i(p) - \theta_j(p)) - i \sin (\theta_i(p) - \theta_j(p)))}{2 \sqrt{\sum_{p=1}^{P} r_i^2(p) P}} + \frac{\sum_{p=1}^{P} r_j(p) (\cos (\theta_i(p) - \theta_j(p)) - i \sin (\theta_i(p) - \theta_j(p)))}{2 \sqrt{\sum_{p=1}^{P} r_j^2(p) P}}.
\] (5.14)

The robust properties of the proposed kernel derive from (i) the use of illumination invariant gradient orientation features, (ii) the way we split the magnitude, and (iii), as shown by Tzimiropoulos et al. (2010, 2011, 2012a), the use of the cosine on the difference of gradient orientations.

5.2.2 Special Kernel Properties

The proposed kernel is a member of a special family of kernel functions. In particular, after simple manipulations, the introduced kernel can be expressed via

\[
k(z_i, z_j) = \begin{bmatrix} r_i \odot e^{i \theta_i} \\ 2 \sqrt{r_i^T r_i P} \end{bmatrix}^H \begin{bmatrix} e^{i \theta_j} \\ r_j \odot e^{i \theta_j} \\ 2 \sqrt{r_j^T r_j P} \end{bmatrix}
\] (5.15)

where \( e^{i \theta} = \begin{bmatrix} e^{i \theta_1} & \cdots & e^{i \theta_P} \end{bmatrix}^T \) and \( \odot \) computes the element-wise product. Notice, from eq. (5.15) two explicit mappings \( a : \mathbb{C}^P \to \mathbb{C}^{2P} \) and \( b : \mathbb{C}^P \to \mathbb{C}^{2P} \) can be extracted, and are given by

\[
a(z) = \begin{bmatrix} r \odot e^{i \theta} \\ 2 \sqrt{r^T r P} \end{bmatrix} \quad \text{and} \quad b(z) = \begin{bmatrix} e^{i \theta} \\ r \odot e^{i \theta} \\ 2 \sqrt{r^T r P} \end{bmatrix}.
\] (5.16)

Therefore, the kernel \( k \) in eq. (5.14) is in fact a member of a special family of kernels, for which there exist two mappings \( a \) and \( b \) that can express the kernel via

\[
k(z_i, z_j) = a(z_i)^H b(z_j) = b(z_i)^H a(z_j).
\] (5.17)

When \( a \neq b \), kernels of the form shown in eq. (5.17) which also satisfy \( k(z_i, z_i) \geq 0 \) are in general non-positive definite, as the triangular inequality may not hold. Notice, there could exist two vectors \( z_i \) and \( z_j \) such that \( \text{Re}(k(z_i, z_j)) > \sqrt{k(z_i, z_i) k(z_j, z_j)} \). In fact, this is the case for the proposed kernel \( k \) in eq. (5.14), which satisfies \( k(z_i, z_i) = a(z_i)^H b(z_i) \geq 0 \). Thus, this kernel is indefinite and defines an implicit mapping \( \phi : \mathbb{C}^P \to \mathcal{K} \) into a finite dimensional Krein space.

Chapter 5. Tracking with a Non-positive Kernel 121
Analogous to Hilbert space, the proposed kernel is equivalent to the dot-product in feature space, i.e. \( k(z_i, z_j) = \langle \phi(z_j), \phi(z_i) \rangle_k \). Moreover, the kernel induced squared distance in feature space is given by

\[
d(z_i, z_j)^2 = k(z_i, z_i) + k(z_j, z_j) - 2k(z_i, z_j) = k(z_i, z_i) - k(z_i, z_j) - k(z_j, z_i) + k(z_j, z_j).
\]

(5.18)

Notice, while the new kernel is non-positive definite, its induced squared distance is always positive. Theorem 2 proves this statement.

**Theorem 2.** The squared distance is always positive, i.e. \( d(z_i, z_j)^2 \geq 0 \).

**Proof.** The distance is denoted by \( d: \mathbb{C}^P \times \mathbb{C}^P \rightarrow \mathbb{R} \), and induced by kernel \( k: \mathbb{C}^P \times \mathbb{C}^P \rightarrow \mathbb{C} \). Let us write

\[
d(z_i, z_j)^2 = \frac{2 \sum_{p=1}^{P} r_i(p)}{2 \sqrt{\sum_{p=1}^{P} r_i^2(p)P}} + \frac{2 \sum_{p=1}^{P} r_j(p)}{2 \sqrt{\sum_{p=1}^{P} r_j^2(p)P}} - 2 \left( \frac{\sum_{p=1}^{P} r_i(p) \cos(\theta_i(p) - \theta_j(p))}{2 \sqrt{\sum_{p=1}^{P} r_i^2(p)P}} + \sum_{p=1}^{P} r_j(p) \cos(\theta_i(p) - \theta_j(p)) \right) + \sum_{p=1}^{P} r_j(p)
\]

(5.19)

Thus the kernel induced squared distance is always positive.

A final remark goes towards the complexity of mappings \( a \) and \( b \), which may degrade the systems performance if ill-formed. In my case however, the computation of the proposed kernel \( \text{via} \ a \) and \( b \) is of the same complexity as the computation of the kernel alone. Both are of time complexity \( \mathcal{O}(P) \), as the \( P \)-dimensional input samples are extended by a constant factor of 2 for each mapping.

### 5.3 Direct Incremental KPCA in Krein Space

The proposed kernel is non-positive definite. Consequently, an implicit Hilbert feature space is not applicable. In this case, the appropriate vector space where the kernel represents a dot-product is a Krein space (Pęckalska and Haasdonk 2009). In this section I present a direct \( \text{i.e. explicit} \) incremental KPCA in Krein space which is specifically
designed to make use of the special properties of the proposed kernel. First however, an offline KPCA with arbitrary non-positive definite kernels in Krein space is developed. Then the special form of the new kernel is exploited to form a direct version of KPCA. Finally, the direct incremental KPCA is proposed.

### 5.3.1 General KPCA with Arbitrary Krein Space Kernels

Let us denote the set of input samples as \( \{ z_n \}_{n=1}^{N} \), where \( z_n \in \mathbb{C}^P \) is a \( P \)-dimensional vector composed of complex numbers. Their implicitly mapped sample matrix is composed as \( X = \left[ \phi(z_1) \cdots \phi(z_N) \right] \). Motivated by KPCA and pseudo-Euclidean embedding proposed by Pękalska and Duin (2005); Pękalska and Haasdonk (2009), KPCA for Krein spaces is developed in the following.

Let us write the implicit mean vector \( \mu \) and centralized matrix \( \tilde{X} \) of the input as

\[
\mu = \frac{1}{N}X1_{N \times 1} = XM
\]

\[
\tilde{X} = \tilde{X}(I_N - \frac{1}{N}1_{N \times N}) = XC
\]

where \( 1_{a \times b} \) is an \( a \times b \) matrix with all elements equal to 1. The matrices \( M = \frac{1}{N}1_{N \times 1} \) and \( C = I_N - \frac{1}{N}1_{N \times N} \) are linear expansions to the implicitly mapped sample matrix, and are designed to find the mean and centralized matrix respectively. Let us now define the total scatter matrix in Krein space \( \mathcal{K} \) as

\[
S_{\mathcal{K}} \triangleq \frac{1}{N} \sum_{n=1}^{N} (\phi(z_n) - \mu)(\phi(z_n) - \mu)^* = \frac{1}{N} \tilde{X} \tilde{X}^* = \frac{1}{N} \tilde{X} \tilde{X}^H J = S_{|\mathcal{K}|} J
\]

where \( S_{|\mathcal{K}|} \) is the total scatter matrix in the associated Hilbert space \( |\mathcal{K}| \).

**Optimization Problem**

Analogously to KPCA in Hilbert space, KPCA in Krein space is generalized as follows. To compute the eigenvectors \( \{ u_r \}_{r=1}^{R} \), which are in Krein space (i.e. \( u_r \in \mathcal{K} \)), and the projection \( U = \begin{bmatrix} u_1 & \cdots & u_R \end{bmatrix} \), the following optimization is solved

\[
U = \underset{\hat{U}}{\arg\max} \text{tr} \left( \hat{U}^* S_{\mathcal{K}} \hat{U} \right) \quad (5.23)
\]

subject to \( U^* U = J \),
where \( \text{tr}(\cdot) \) computes the trace of a matrix. Let us rewrite the projection as a linear combination of the samples, i.e. \( U = \tilde{X} \hat{U} \). Then the optimization in eq. (5.23) becomes

\[
\hat{U} = \arg \max_U \text{tr} \left( \tilde{U}^H \tilde{X}^H J \tilde{X} \tilde{X}^H J \tilde{X} \hat{U} \right) = \arg \max_U \text{tr} \left( \tilde{U}^H K \hat{U} \right) \tag{5.24}
\]

subject to \( \tilde{U}^H \tilde{X}^H J \tilde{X} \tilde{U} = \tilde{U}^H K \tilde{U} = J \),

where \( K = \tilde{X}^* \tilde{X} \) is the centralized kernel matrix. The eigenvalue decomposition of \( K \) then yields the solution of the above, as

\[
K = \Omega \Lambda \Omega^H = \Omega |\Lambda|^{\frac{1}{2}} J |\Lambda|^{\frac{1}{2}} \Omega^H, \tag{5.25}
\]

where \( \Lambda \) is a diagonal matrix whose main diagonal consists of \( R_+ \) positive and \( R_- \) negative eigenvalues \( (R_+ + R_- = R \leq N) \) in the following order: first, positive eigenvalues with decreasing values, then negative ones with decreasing absolute values and finally zero values. The notation \( |\Lambda| \triangleq R(\Lambda) \) is to mean the diagonal matrix containing the magnitudes of the eigenvalues. The fictitious fundamental symmetry \( J \) in eq. (5.25) may differ from its previous values, but is defined as before in eq. (5.9). The signature of the pseudo-Euclidian space is given by the vector \( [R_+ R_-]^T \). Using the kernel’s decomposition in eq. (5.25), the optimal solution for eq. (5.24) is obtained via \( \hat{U}_R = \Omega_R |\Lambda_R|^{-\frac{1}{2}} \), where \( \Lambda_R \) contains the non-zero eigenvalues and \( \Omega_R \) the corresponding eigenvectors of the kernel matrix. Thus, the implicit subspace of the KPCA is provided by the linear combination \( U_R = \tilde{X} \Omega_R |\Lambda_R|^{-\frac{1}{2}} \).

### Sample Projection

Let there be a new sample \( z \in \mathbb{C}^P \), whose implicit mapping is denoted by \( x = \phi(z) \in \mathcal{K} \). Notice, the part of \( x \) which belongs to the positive subspace \( U_{R_+} \) in \( \mathcal{K}_+ \) is given by

\[
x^+ = |\Lambda_{R_+}|^{-\frac{1}{2}} \Omega_{R_+}^H C^T \tilde{X}^* \phi(z) = |\Lambda_{R_+}|^{-\frac{1}{2}} \Omega_{R_+}^H C^T \begin{bmatrix} k(z_1, z) \\ \vdots \\ k(z_N, z) \end{bmatrix}, \tag{5.26}
\]

where \( \Lambda_{R_+} \) contains only the positive eigenvalues, and \( \Omega_{R_+} \) denotes the corresponding eigenvectors. Similarly, the features \( x^- \) which belong to the negative subspace \( U_{R_-} \) in \( \mathcal{K}_- \) are found as

\[
x^- = |\Lambda_{R_-}|^{-\frac{1}{2}} \Omega_{R_-}^H C^T \tilde{X}^* \phi(z), \tag{5.27}
\]

124 5.3. Direct Incremental KPCA in Krein Space
where $\Lambda_{R_-}$ and $\Omega_{R_-}$ corresponds to the negative eigenvalues. Furthermore, the inner product of two arbitrary implicit samples $x_i, x_j \in \mathcal{K}$, obtained from $z_i, z_j \in \mathbb{C}^P$, can be verified to be equivalent to the kernel, as

$$
\langle x_j, x_i \rangle_{\mathcal{K}} = x_i^* x_j = x_i^* J x_j
= \phi(z_i)^* \tilde{X} \Omega [-\frac{1}{2} J |\Lambda|^{-\frac{1}{2}} \Omega] \tilde{X} \phi(z_j)
= \phi(z_i)^* J U^* U \phi(z_j) = \phi(z_i)^* J \phi(z_j)
= \langle \phi(z_j), \phi(z_i) \rangle_{\mathcal{K}} = k(z_i, z_j).
$$

(5.28)

Reduction Strategy

In order to establish a dimensionality reduction strategy, let us expand the objective function of the optimization problem from eq. (5.23)

$$
\text{tr} (U^* S_{\mathcal{K}} U) = \text{tr} \left( \tilde{U}^H K K \tilde{U} \right)
= \text{tr} \left( |\Lambda|^{-\frac{1}{2}} \Omega \Lambda \Omega^H \Omega |\Lambda|^{-\frac{1}{2}} \right)
= \text{tr} \left( |\Lambda|^{-\frac{1}{2}} \Lambda \Lambda |\Lambda|^{-\frac{1}{2}} \right)
= \text{tr} (|\Lambda|) = \sum_{r=1}^{R} |\lambda_r|,
$$

(5.29)

where $\{\lambda_r\}_{r=1}^{R}$ is the set of eigenvalues. As it can be observed, the actual functional to be minimized is based on the absolute eigenvalues, $\{|\lambda_r|\}_{r=1}^{R}$. Hence, the dimensionality reduction may be performed by removing the eigenvectors that correspond to the smallest in terms of eigenvalues magnitude. The signature of the reduced Krein space is then given by $[ R_+ \  R_- ]^T$ with $R_+ \leq R_+$, $R_- \leq R_-$ and $R_+ + R_- = R$.

5.3.2 Direct KPCA in Krein Spaces

In this section I capitalize on the properties of the proposed kernel in order to define a special version of KPCA in Krein spaces. In particular, the proposed KPCA does not require the computation of preimages and as such is called a direct KPCA. The properties of the direct KPCA are then used as the basis for an exact incremental KPCA in the following section. First, however, a batch version is proposed here.

As before, $X = \left[ \phi(z_1) \ \cdots \ \phi(z_N) \right]$ denotes the matrix of $N$ unknown sample mappings in Krein space $\mathcal{K}$ (zero mean is assumed for simplicity, without loss of generalization). Let us define a matrix $X^{(a)}$ related to the explicit mapping $a$, such that $X^{(a)} = \left[ a(z_1) \ \cdots \ a(z_N) \right]$. Similarly, $X^{(b)} = \left[ b(z_1) \ \cdots \ b(z_N) \right]$ forms the explicit
matrix for mapping $b$. Notice here, even though mappings $a$ and $b$ are known, mapping $\phi$ is not known and neither can be explicitly defined. Furthermore, both explicit mappings are non-equivalent, i.e. $a \neq b$.

Let us develop direct KPCA by employing matrices $X^{(a)}$ and $X^{(b)}$. From the eigenvalue decomposition of the kernel matrix $K = X^* X$, we get

$$X^* X = X^{(a)H} X^{(b)} = \Omega \Lambda \Omega^H.$$ (5.30)

Hence, under consideration of eq. (5.25), the eigenspectrum of the KPCA is given by the eigenvectors $U = X\Omega|\Lambda|^{-\frac{1}{2}}$ and the non-zero eigenvalues $\Sigma = |\Lambda|^\frac{1}{2}$. We can now define two explicit projections which provide the direct properties, i.e.

$$A \triangleq X^{(a)} \Omega |\Lambda|^{-\frac{1}{2}},$$ (5.31)
$$B \triangleq X^{(b)} \Omega |\Lambda|^{-\frac{1}{2}}.$$ (5.32)

With the pseudo-subspaces $A$ and $B$, the mapped matrices $X^{(a)}$ and $X^{(b)}$ are given by $X^{(a)} = A \Sigma \Omega^H$ and $X^{(b)} = B \Sigma \Omega^H$ respectively. Additionally, the direct KPCA is satisfied as the following properties hold (with sample $z \in \mathbb{C}^P$ in input space):

$$U^* \phi(z) = \left| \Lambda \right|^{-\frac{1}{2}} \Omega^H \begin{bmatrix} k(z_1, z) & \cdots & a(z_1)^H b(z) \\ k(z_N, z) & \cdots & a(z_N)^H b(z) \end{bmatrix} = A^H b(z)$$ (5.33)
$$A^H b(z) = \left| \Lambda \right|^{-\frac{1}{2}} \Omega^H \begin{bmatrix} a(z_1)^H b(z) \\ \cdots \\ a(z_N)^H b(z) \end{bmatrix} = B^H a(z)$$ (5.34)
$$A^H B = \left| \Lambda \right|^{-\frac{1}{2}} \Omega^H X^{(a)H} X^{(b)} \Omega |\Lambda|^{-\frac{1}{2}} = U^* U = J.$$ (5.35)

In particular, the implicit subspace $U$ is directly representable by two explicit pseudo-subspaces $A$ and $B$. Notice, the columns of $A$ and $B$ still relate to the eigenvalues in $\Sigma$. The procedure for computing this method is summarized in algorithm 5.1.

### 5.3.3 Direct Incremental KPCA

Finally, to conclude the current section, I present the direct incremental update procedure for online subspace learning with kernels that satisfy the properties in eq. (5.17). In particular, I show that the incremental version of the proposed direct KPCA above does not require the computation of preimages, as the use of the explicit equivalent of the implicit
kernel mapping allows for an exact incremental KPCA. Notice, the KPCA proposed by Chin and Suter (2007) requires an approximate reduced set expansion of preimages to satisfy online learning requirements.

Let us denote by $X_{t-1}$ the implicitly mapped sample matrix at the previous time step. Given a set of $N_0$ new observations $\{z_n\}_{n=N_0}^{N_0+N_t}$, and the implicitly mapped sample matrix $X_\delta = \begin{bmatrix} \phi(z_{N_{t-1}+1}) & \cdots & \phi(z_{N_{t-1}+N_t}) \end{bmatrix}$ the subspace of the combined sample matrix $[X_{t-1} \quad X_\delta]$ is to be found. By representing the old mapped sample matrix by its decomposed format $X_{t-1} \approx U_{t-1} \Sigma_{t-1} \Omega_{t-1}^H$, the combined matrix can be written as

$$
\begin{bmatrix}
U_{t-1} \Sigma_{t-1} \Omega_{t-1}^H & U_{t-1} U_{t-1}^* X_\delta + QR 
\end{bmatrix},
$$

(5.36)

where $Q$ is an orthogonal matrix, and $R$ ensures that the product is equivalent to the subspace complementary of the new data, i.e., $QR = X_\delta - U_{t-1} U_{t-1}^* X_\delta$. For the sake of notation, I denote this complementary by $H = QR$ in the following. The eigenvalue decomposition of its kernel matrix, i.e., $H^* H = \hat{\Omega} \hat{\Lambda} \hat{\Omega}^H$, finds $Q = H \hat{\Omega} |\hat{\Lambda}|^{-\frac{1}{2}}$ and $R = |\hat{\Lambda}|^{\frac{1}{2}} \hat{\Omega}^H$.

Chapter 3 reveals how the new subspace of eq. (5.36) is computed for incremental KPCA with a known, explicit mapping and kernels in Hilbert space. In particular, recall the formulation of eq. (5.36) as

$$
\begin{bmatrix}
U_{t-1} & Q \\
\Sigma_{t-1} & U_{t-1}^* X_\delta \\
0 & R \\
V_{t-1} & 0 & I
\end{bmatrix}^H,
$$

(5.37)

where the SVD of $L = \begin{bmatrix} \Sigma_{t-1} & U_{t-1}^* X_\delta \\
0 & R \end{bmatrix}$ provides the main step towards the solution. In this chapter, however, the kernel-induced mapping $\phi$ is implicit and indefinite. Hence, $Q$ and $R$ is only indirectly provided. Nonetheless, by utilizing the explicit mappings $a$
and $b$, a direct incremental update can still be achieved.

Given the mapping matrices $X^{(a)}_\delta = \begin{bmatrix} a(z_{Nt-1+1}) & \cdots & a(z_{Nt-1+N_t}) \end{bmatrix}$ and $X^{(b)}_\delta = \begin{bmatrix} b(z_{Nt-1+1}) & \cdots & b(z_{Nt-1+N_t}) \end{bmatrix}$ and the pseudo-subspaces $A_{t-1}$ and $B_{t-1}$, let us define $H^{(a)} = X^{(a)}_\delta - A_{t-1}B^H_{t-1}X^{(a)}_\delta$ and $H^{(b)} = X^{(b)}_\delta - B_{t-1}A^H_{t-1}X^{(b)}_\delta$. Now, the kernel matrix of the unknown complementary matrix and its decomposition is computed via

$$H^{(a)H}H^{(b)} = (X^{(a)}_\delta - A_{t-1}B^H_{t-1}X^{(a)}_\delta)^H(X^{(b)}_\delta - B_{t-1}A^H_{t-1}X^{(b)}_\delta)$$

$$= (X_\delta - U_{t-1}U^*_tX_\delta)^*(X_\delta - U_{t-1}U^*_tX_\delta)$$

$$= H^*H = \hat{\Omega}\hat{\Lambda}\hat{\Omega}^H$$

and the SVD of the inner term of eq. (5.37) can be computed

$$L = \begin{bmatrix} \Sigma_{t-1} & A^H_{t-1}X^{(b)}_\delta \\ 0 & |\hat{\Lambda}|^{\frac{1}{2}}\hat{\Omega}^H \end{bmatrix} = \hat{U}\Sigma\hat{V}^H.$$  \hspace{1cm} (5.39)

As the kernel resides in a Krein space, this matrix is indefinite and may contain positive and negative eigenvalues.

Notice, the SVD of the combined matrix is given by

$$X_t = \begin{bmatrix} X_{t-1} & X_\delta \end{bmatrix} = \begin{bmatrix} [U_{t-1} & Q] \hat{U} \\ \hat{\Sigma} \end{bmatrix} \begin{bmatrix} \hat{V}^H_{t-1} & \begin{bmatrix} V^H & 0 \\ 0 & I \end{bmatrix} \end{bmatrix},$$  \hspace{1cm} (5.40)

where only $\begin{bmatrix} U_{t-1} & Q \end{bmatrix}$ is implicit (while $V_{t-1}$ and $\hat{V}$ are not required). Let us define $Q^{(a)} = H^{(a)}\hat{\Omega}|\hat{\Lambda}|^{-\frac{1}{2}}$ and $Q^{(b)} = H^{(b)}\hat{\Omega}|\hat{\Lambda}|^{-\frac{1}{2}}$, and express the new pseudo-subspaces at time $t$ as $A_t = \begin{bmatrix} A_{t-1} & Q^{(a)} \end{bmatrix} \hat{U}$ and $B_t = \begin{bmatrix} B_{t-1} & Q^{(b)} \end{bmatrix} \hat{U}$ with eigenvalues $\Sigma_t = |\hat{\Sigma}|$. Note that this choice satisfies the properties in eq. (5.33), eq. (5.34) and eq. (5.35).

Algorithm 5.2 summarizes the proposed incremental update. Due to the direct approach to KPCA, the storage requirements for the online update is of fixed complexity and the complexity of the update is also bound. In particular, with the proposed kernel in eq. (5.14), the storage requirements are of $O(2 \times 2P(R + N_\delta))$ while the computation of the update is of $O(4PN_\delta^2)$. Hence, the overall running speed is similar to that of the incremental linear PCA by Ross et al. (2008) and incremental Euler PCA in chapter 3.

Summarizing, this section presents a general approach for incremental KPCA with non positive definite kernels in Krein space that can be described by two explicitly given mappings $a$ and $b$ such that eq. (5.17) holds. This approach is coined the direct incremental KPCA (DIKPCA). DIKPCA facilitates fast update of an online learned eigenvalue decomposition. In contrast to the incremental version of KPCA proposed by Chin and
$C$ is the parameter that controls the spread. Note, the distribution of the samples’ probabilities can be calculated via mappings $a$ and $b$, and pseudo-subspaces $A$ and $B_t$ to avoid the unknown subspace $U$.

Suter (2007) which deals with positive definite kernels, the presented approach uses a class of special indefinite kernels which renders finding preimages unnecessary. Therefore, the proposed method is faster, exact and robust.

### 5.4 Visual Object Tracking

I apply the robust and efficient online learning framework to the application of visual object tracking. In particular, the online appearance model is combined with a motion affine transformation in a particle filter framework. The overall methodology is similar to that of chapter 3 and follows the work of Ross et al. (2008) and Chin and Suter (2007) which is introduced in chapter 2.

As in previous chapters, the transition model is provided by the GMM around the approximation of the state posterior distribution of the previous time step. The observation model is based on the probability of a sample being generated by the online appearance model. As a Krein space is employed, this probability is assumed to be analogous to an exponential. Specifically, given the set of observations $\{z_m^{(t)}\}_{m=1}^M$ at time $t$, the distribution is provided by

$$e^{-\gamma \left( \phi(z_m^{(t)}) - U_t \bar{U}_t^* \phi(z_m^{(t)}) \right)^* \left( \phi(z_m^{(t)}) - U_t \bar{U}_t^* \phi(z_m^{(t)}) \right)} = e^{-\gamma \left( a(z_m^{(t)}) - A_t B_t^* a(z_m^{(t)}) \right)^* \left( b(z_m^{(t)}) - B_t A_t^* b(z_m^{(t)}) \right)}$$

where $\gamma$ is the parameter that controls the spread. Note, the distribution of the samples’ probabilities can be calculated via mappings $a$ and $b$, and pseudo-subspaces $A$ and $B_t$ to avoid the unknown subspace $U$.

---

**Algorithm 5.2 Update Procedure of Direct Incremental KPCA**

**Input:** The previous eigenspectrum $A_{t-1}$, $B_{t-1}$ and $\Sigma_{t-1}$, the number of previous samples $N_{t-1}$, the set of $N_\delta$ new observations $\{z_n\}_{n=N_{t-1}+1}^{N_{t}}$ with $z_n \in \mathbb{C}^P$, the two mappings $a : \mathbb{C}^P \rightarrow \mathbb{C}^{2P}$ and $b : \mathbb{C}^P \rightarrow \mathbb{C}^{2P}$, that satisfy eq. (5.17), and the number of components $R$.

**Output:** The updated eigenspectrum $A_t \in \mathbb{C}^{2P \times R}$, $B_t \in \mathbb{C}^{2P \times R}$ and $\Sigma_t \in \mathbb{R}^{R \times R}$.

1. Calculate the mapped sample matrices $X_{\delta}^{(a)}$ and $X_{\delta}^{(b)}$, of the samples $\{z_n\}_{n=N_{t-1}+1}^{N_{t}}$.
2. Find $H^{(a)} = X_{\delta}^{(a)} - A_{t-1} B_{t-1}^* X_{\delta}^{(a)}$ and $H^{(b)} = X_{\delta}^{(b)} - B A_{t-1}^* X_{\delta}^{(b)}$.
3. Compute $H^{(a)^*} H^{(b)} = H^* H = \hat{\Lambda} \hat{\Lambda}^*$ and set $R = |\hat{\Lambda}|^{\frac{1}{2}}$, $Q^{(a)} = H^{(a)} \hat{\Omega} |\hat{\Lambda}|^{-\frac{1}{2}}$ and $Q^{(b)} = H^{(b)} \hat{\Omega} |\hat{\Lambda}|^{-\frac{1}{2}}$.
4. Form $L = \begin{bmatrix} \Sigma_{t-1} & A_{t-1}^* X_{\delta}^{(b)} \\ 0 & R \end{bmatrix}$ and compute $L = \tilde{U} \hat{\Sigma} \tilde{V}^*.$
5. Set $A_t = [A_{t-1} \; Q^{(a)}] \tilde{U}$, $B_t = [B_{t-1} \; Q^{(b)}] \tilde{U}$ and $\Sigma_t = |\tilde{\Sigma}|$.
6. Obtain $R$-reduced set of $A_t$ and $B_t$ via $R$ largest eigenvalue magnitudes in $\Sigma_t$.  

Chapter 5. Tracking with a Non-positive Kernel
Algorithm 5.3 describes the proposed visual tracking framework. Figure 5.2 visualizes the setup. The main differences to the Euler tracker (eT) from chapter 3 is that gradient features are extracted from the particles and two mappings are applied, and the PCA of the appearance model is now composed out of two pseudo-subspaces. The proposed setup is coined direct incremental KPCA tracker (DIKT) in the following.

5.5 Results and Discussion

The evaluation is pursued in two stages. First, the performance of DIKT is tested against other popular holistic online tracking algorithms. Second, the general robustness of the direct KPCA framework with the proposed kernel is evaluated, and compared to standard PCA and KPCA in Hilbert space with a GRBF kernel.

5.5.1 Object Tracking

In this section, the performance evaluation results of the proposed direct incremental KPCA tracker (DIKT) is presented. The proposed method is again compared with that of the four other commonly found tracking approaches, previously used in chapter 3:

- IVT by Ross et al. (2008), the Matlab implementation of which is publicly available at [http://www.cs.toronto.edu/~dross/ivt/](http://www.cs.toronto.edu/~dross/ivt/)
Algorithm 5.3 DIKT at Time $t$

**Input:** The previous pseudo-subspaces $A_{t-1}$ and $B_{t-1}$, the eigenvalues $\Sigma_{t-1}$, locations $\{A^{(t-1)}_m\}_{m=1}^M$, weights $\{w^{(t-1)}_m\}_{m=1}^M$, image frame $F_t$ and the explicit mappings $a$ and $b$ of the kernel.

1: Draw $M$ particles $\{A^{(t)}_m\}_{m=1}^M$ from the transition model
2: Take all image patches $\{I^{(t)}_m\}_{m=1}^M$ from $F_t$ which correspond to the particles in $\{A^{(t)}_m\}_{m=1}^M$.
3: Extract the gradient-based representations (eq. (5.10)) and store them in lexicographical order to form the observation vectors $\{z^{(t)}_m\}_{m=1}^M$.
4: Compute the exponential in eq. (5.41) for each particle and normalize these as weights $\{w^{(t)}_m\}_{m=1}^M$ to form a probability distribution.
5: Choose $A^{(t)}_b$ and $z^{(t)}_b$ as the affine transform and features of the particle with the largest weight.
6: Using $a(z^{(t)}_b)$ and $a(z^{(t)}_b)$, update the subspace by applying algorithm 5.2 in a batch after a certain number of frames (5 in this implementation).

- IKPCA by Chin and Suter (2007), the Matlab implementation of the incremental KPCA is kindly provided by the authors of the paper

- L1 tracker proposed by Mei and Ling (2009), the implementation of which is publicly available at http://www.ist.temple.edu/~hbling/code_data.htm

- MIL tracker by Babenko et al. (2011), the implementation only with translation motion model of which, is publicly available at http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml. As in chapter 3 a carefully modified version that supports an affine motion model in a particle filter framework is employed.

The performance of all methods is evaluated on nine very popular video sequences, subsets of which are used by Babenko et al. (2009, 2011); Comaniciu et al. (2003); Mei and Ling (2009); Ross et al. (2008). These videos contain drastic changes of the target’s appearance, including pose variation, occlusions and non-uniform illumination. Representative frames of the video sequences are illustrated in figure 5.5 to figure 5.13. While videos Vid1 to Vid5 illustrates the performance for the application of face tracking, tracking of vehicles is assessed using Vid6 and Vid7. Finally, other objects are tracked in Vid8 and Vid9. Videos Vid4 and Vid5 are available at http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml and the remaining videos are published at http://www.cs.toronto.edu/~dross/ivt/.

The annotated points, introduced in chapter 3 are also used in this chapter. In particular, three to seven fiducial points are given for each video. As usual, the quantitative performance evaluation is based on the RMS errors between the true and the estimated locations of these points. Precision plots which show the quality of the tracking, i.e.
the percentage of frames in which the target was tracked with an RMS error less than a certain threshold, are also provided.

In the experiments, all trackers use an affine motion model with a fixed number of drawn particles (800 particles). The results obtained for two versions of the experimental settings are presented in the following. The first version tests the performance of all trackers using video-specific parameters. That is, for each tracker and video pair, the parameters which give the best performance in terms of robustness (i.e. how many times the tracker lost the target completely after visual inspection) and accuracy (measured by the mean RMS error) are found and their results compared. In the second and most interesting version, the parameters remain fixed in all videos for each tracker. Specifically, the parameters which give the best performance in terms of robustness and accuracy for all videos are found – i.e. tracker-specific parameters are used.

Tracking with Video-specific Parameters

The notation DIKT-specific, IVT-specific, IKPCA-specific, L1-specific and MIL-specific relates to the video-specific versions of the trackers. The optimization criterion is based on the minimization of the RMS error between the true and the estimated location of the points. Apart from the L1-specific tracker (for which the resolution of the template increases geometrically the complexity) the tracking template is chosen to be of resolution 32 × 32 pixels. Similar to chapter 3, all trackers are optimized with respect to the variance of the GMM from which the particles are sampled. Except for the variance of the GMM, which is common for all the systems, further optimization include the number of the components in the appearance model and the spread γ of DIKT-specific, IVT-specific and IKPCA-specific. For IKPCA-specific the GRBF is optimized with respect to its radius. With L1-specific the tracking becomes impractical when the templates’ resolution is more than 20 × 20 pixels, thus L1-specific is optimized with respect to the number of templates and their resolution. MIL-specific is optimized with respect to the parameters described by Babenko et al. (2011), which include the number of positive patches in each frame, the number that controls the sampling of negative examples, and the learning rate for the weak classifiers. The tracking rates for the tested systems are: for IVT 3.2 frames/second, DIKT 3 frames/second, IKPCA 0.7-1 frame/second, MIL 0.15 frames/second and L1 less than 0.1 frames/second.¹

For the video-specific version of the trackers, table 5.1 lists the mean RMS error for all sequences, while figure 5.3 plots the RMS error for each frame number. Figure 5.4 shows

¹Tests were conducted in MATLAB on a desktop computer with an Intel core i7 870 processor at 2.93 GHz and 8 GB RAM.
Figure 5.3: Mean RMS error for video-specific trackers at each frame. The tuned DIKT-specific is shown against IVT-specific, IKPCA-specific, L1-specific and MIL-specific for Vid1 to Vid9.
Figure 5.4: Tracking precision of video-specific trackers for each video sequence. The tuned DIKT-specific is shown against IVT-specific, IKPCA-specific, L1-specific and MIL-specific for Vid1 to Vid9.
Table 5.1: Mean RMS error for tracking with video-specific parameters. I write "(lost)" to indicate sequences in which the tracker clearly does not follow the target throughout after visual inspection. The results of $\epsilon$T-specific from chapter 3 are included as further comparison.

<table>
<thead>
<tr>
<th></th>
<th>$\text{Vid1}$</th>
<th>$\text{Vid2}$</th>
<th>$\text{Vid3}$</th>
<th>$\text{Vid4}$</th>
<th>$\text{Vid5}$</th>
<th>$\text{Vid6}$</th>
<th>$\text{Vid7}$</th>
<th>$\text{Vid8}$</th>
<th>$\text{Vid9}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVT-specific</td>
<td>6.82</td>
<td>(lost)</td>
<td>4.07</td>
<td>10.79</td>
<td>(lost)</td>
<td>3.31</td>
<td>1.78</td>
<td>2.62</td>
<td>(lost)</td>
</tr>
<tr>
<td>IKPCA-specific</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
<td>(lost)</td>
</tr>
<tr>
<td>L1-specific</td>
<td>6.17</td>
<td>(lost)</td>
<td>2.87</td>
<td>11.10</td>
<td>12.68</td>
<td>9.53</td>
<td>1.62</td>
<td>13.58</td>
<td>(lost)</td>
</tr>
<tr>
<td>MIL-specific</td>
<td>16.95</td>
<td>(lost)</td>
<td>13.61</td>
<td>14.62</td>
<td>37.56</td>
<td>12.73</td>
<td>4.14</td>
<td>23.87</td>
<td>17.62</td>
</tr>
<tr>
<td>DIKT-specific</td>
<td>4.06</td>
<td>2.27</td>
<td>2.40</td>
<td>5.62</td>
<td>11.28</td>
<td>3.40</td>
<td>1.80</td>
<td>1.96</td>
<td>5.90</td>
</tr>
<tr>
<td>$\epsilon$T-specific</td>
<td>5.14</td>
<td>(lost)</td>
<td>3.68</td>
<td>4.68</td>
<td>(lost)</td>
<td>3.04</td>
<td>1.73</td>
<td>2.44</td>
<td>(lost)</td>
</tr>
</tbody>
</table>

In general, DIKT-specific outperforms all other trackers in terms of robustness, accuracy and precision. In terms of robustness, DIKT-specific successfully follows the target for all sequences, including Vid2, where all other trackers fail. In terms of RMS error, DIKT-specific achieves by far the best results for all videos with the exception of video Vid7, where it is slightly outperformed by IVT-specific and L1-specific. While MIL-specific appears to be robust (it loses the target for one video sequence only), it is generally not as precise as the other trackers. Clearly, IKPCA-specific is inferior to the other trackers. It is believed that this performance degradation is induced by the search for preimages which accumulates errors and eventually makes the tracker lose the target in prolonged and challenging video sequences.

Finally, I include the results of $\epsilon$T-specific from chapter 3 as additional comparison in table 5.1. DIKT-specific mainly improves upon robustness, as $\epsilon$T-specific fails on three video sequences.
Figure 5.5: Example frames of the tracking with video-specific parameters for Vid1. The results of DIKT-specific (○) versus IVT-specific (□), IKPCA-specific (∗), L1-specific (▷) and MIL-specific (◁) are shown. The ground truth is indicated by ×. The tracked area of DIKT-specific is visualized by a magenta bounding box. – DIKT-specific is clearly more robust to pose variations and motion blur than the other trackers, as the target is tracked with high precision throughout the video sequence.
Figure 5.6: Example frames of the tracking with video-specific parameters for Vid2. Annotation follows that of figure 5.5. – Vid2 illustrates the robust performance of DIKT-specific under challenging pose changes, variations of illumination and cast shadows.
Figure 5.7: Example frames of the tracking with video-specific parameters for Vid3. Annotation follows that of figure 5.5. – Another example of successful tracking under challenging pose changes, variations of illumination and cast shadows.
Figure 5.8: Example frames of the tracking with video-specific parameters for Vid4. Annotation follows that of figure 5.5. – In many frames of Vid4, the subject is not fully visible. DIKT-specific is more robust and tracks the target more accurately and precisely.
Figure 5.9: Example frames of the tracking with video-specific parameters for Vid5. Annotation follows that of figure 5.5. – DIKT-specific copes with the changes and adapts to the targets different visual properties during the two 360° rotations successfully.
Figure 5.10: Example frames of the tracking with video-specific parameters for Vid6. Annotation follows that of figure 5.5. – DIKT-specific successfully performs tracking under severe illumination changes and background clutter. The illumination changes mostly affect MIL-specific and L1-specific.
Figure 5.11: Example frames of the tracking with video-specific parameters for Vid7. Annotation follows that of figure 5.5. – All trackers, apart from IKPCA-specific, perform with high precision and accuracy, although MIL-specific is least precise.
Figure 5.12: Example frames of the tracking with video-specific parameters for Vid8. Annotation follows that of figure 5.5. – With the exception of DIKT-specific, illumination changes and motion blur result in performance degradation for all trackers. However, DIKT-specific tracks the target successfully, and with high accuracy and precision.
Figure 5.13: Example frames of the tracking with video-specific parameters for Vid9. Annotation follows that of figure 5.5. – Drastic pose variations and illumination changes make tracking in this video sequence challenging. DIKT-specific performs the best in terms of accuracy.
Table 5.2: Mean RMS error for tracking with fixed parameters. I write “(lost)” to indicate sequences in which the tracker clearly does not follow the target throughout after visual inspection. The results of $e_T$ from chapter 3 are included as further comparison.

## Tracking with Fixed Parameters

I denote by DIKT, IVT, IKPCA, L1 and MIL the versions of the trackers which use fixed parameters for all videos. The parameters are tuned to optimize robustness and accuracy for each tracker – i.e. tracker-specific parameters are found. For all the test videos the mean RMS errors is listed in table 5.2, while figure 5.14 and figure 5.15 illustrate the accuracy and precision of the trackers respectively.

Compared to the performance of all other trackers, the performance of DIKT appears to be significantly less affected by video-specific parameter fine tuning. In terms of robustness, DIKT is still among the most robust trackers, as in six out of nine videos the target was successfully tracked. In terms of accuracy and precision, DIKT appears to be the only tracker which performs almost equally well to its video-specific version DIKT-specific from the previous section. Furthermore, notice that the $e_T$ framework from chapter 3 has significantly reduced robustness, as the kernel in $e$-PCA is not optimum for tracking with illumination and pose variation when parameters are fixed.
Figure 5.14: Mean RMS error for all trackers with fixed parameters at each frame. DIKT is shown against IVT, IKPCA, L1 and MIL for Vid1 to Vid9.
Figure 5.15: Tracking precision of each video sequence with fixed tracking parameters. DIKT is shown against IVT, IKPCA, L1 and MIL for Vid1 to Vid9.
Figure 5.16: The data, as projected by the subspaces of PCA, Gaussian and DKPCA. Training is performed on 3 randomly selected classes (i.e. subjects) in the usual manner (trained on 5 images). The corresponding projections of the test samples are shown.

<table>
<thead>
<tr>
<th></th>
<th>trained on 5</th>
<th>trained on 10</th>
<th>trained on 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>31.1%</td>
<td>45.2%</td>
<td>59.3%</td>
</tr>
<tr>
<td>Gaussian</td>
<td>31.6%</td>
<td>45.5%</td>
<td>59.4%</td>
</tr>
<tr>
<td>DKPCA</td>
<td>62.1%</td>
<td>77.7%</td>
<td>88.4%</td>
</tr>
</tbody>
</table>

Table 5.3: Average recognition rate with Yale B Database.

5.5.2 Face Recognition

In this section the performance of the direct KPCA with the presented kernel (DKPCA) is evaluated on face recognition with illumination changes and pose variations. Note, as the number of training sample is small, the batch version of the proposed method, as seen in algorithm 5.1, suffices. The results are compared to standard PCA (PCA) and KPCA with a GRBF kernel (Gaussian). I optimize the GRBF kernel’s deviation for each experiment.

Extended Yale B Database

The extended Yale B database by Lee et al. (2005) contains 16,128 images of 38 subjects in 9 poses and 64 illumination conditions. A subset consisting of 64 near frontal images for each subject is used. For training, I randomly select 5, 10 or 20 images per subject. For testing, the remaining images are used. 20 different random realizations of the training and test sets are employed. Table 5.3 shows the average recognition rate of the tested methods. The direct KPCA performs best for all setups.

Each method is additionally trained with the same random selection of 3 classes, each class with the same 5 random training images. After computing the subspace of each
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>trained on 5</th>
<th>trained on 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>26.1%</td>
<td>39.1%</td>
</tr>
<tr>
<td>Gaussian</td>
<td>26.1%</td>
<td>39.1%</td>
</tr>
<tr>
<td>DKPCA</td>
<td>52.3%</td>
<td>69.4%</td>
</tr>
</tbody>
</table>

**Table 5.4:** Average recognition rate with CMU PIE Database.

The proposed DKPCA algorithm (linear PCA, KPCA using Gaussian and the proposed DKPCA), I project the samples of the test set onto it (the deviation of the GRBF kernel is set to the one that gives the best performance in the recognition experiment). Figure 5.16 plots the features corresponding to the two largest eigenvalues along with some ellipsis which describes the distribution of each facial class. Standard PCA and Gaussian cluster the training data poorly. The proposed DKPCA is more successful in separating the classes.

The enhanced class-separability achieved by the proposed method can be explained by the metric multidimensional scaling (MMS) perspective of PCA which can be provided through the methods of Borg and Groenen (2005); Pekalska et al. (2002). Under this perspective, standard $\ell_2$-norm PCA finds the optimal linear projections that best preserve the Euclidean distances. As is well known, these distances can be arbitrarily biased by the presence of outliers (the same holds for KPCA with GRBF kernels). Therefore, in the presence of outliers, PCA and Gaussian are not suitable for providing a consistent way of measuring distances in a facial class. On the other hand, under the MMS perspective, the proposed DKPCA finds the optimal linear projections that best preserve the proposed robust distance. And because, as argued in this chapter, this distance is robust to outliers, the proposed DKPCA provides a more consistent way of representing the samples in a facial class.

**CMU PIE database**

The final evaluation is conducted on the CMU PIE database created by Sim et al. (2003). The data set consists of more than 41,000 face images of 68 subjects, and contains faces under varying pose, illumination, and expression. Again, a subset of the near frontal poses (C05, C07, C09, C27, C29) is extracted to create a total of 170 images for each subject. For training, a random selection of 5 or 10 images per subject is used. For testing, the remaining images are employed. 20 different random realizations of the training and test sets are produced. The results are shown in table 5.4. Again, the proposed DKPCA outperforms PCA and Gaussian.
5.6 Conclusions

A robust online kernel learning framework for efficient visual tracking is proposed. The main properties to be emphasized is that the kernel follows a domain-specific design and is indefinite (i.e. non positive definite). In particular, a non-linear appearance model, learned via kernel PCA (KPCA) and a robust gradient-based kernel is used.

To allow for indefinite kernels, KPCA with a RKHS is extended to Krein space. To allow for online learning, a special property of the proposed kernel is exploited to formulate a direct version of KPCA, which does not require the calculation of preimages. Based on this property, an efficient and exact incremental KPCA is then developed. By combining the proposed appearance model with a particle filter, a tracking framework is designed. In the evaluation, this framework achieves very competitive performance in many popular difficult tracking scenarios. Finally, further applications of the kernel framework are presented as the direct KPCA is applied to face recognition, for which it improves upon the standard $\ell_2$-norm PCA and KPCA in Hilbert space with a GRBF kernel.

In this chapter, a domain-specific kernel is developed and efficiently applied to online subspace learning. The presented method is exact as well as robust. As in previous chapters, an explicit equivalent of the kernel function is utilized to facilitate fast computation. However, thus far only PCA learning is supported. In the next chapter, I will introduce an incremental algorithm for online learning of the general eigenvalue problem, applied to slow feature analysis (SFA).
Chapter 6

Incremental Slow Feature Analysis

In previous chapters, the focus is directed towards robust online subspace learning in PCA settings. I now expand this theory to the generalized eigenvalue problem. Specifically, I propose an incremental framework for slow feature analysis (SFA).

SFA originates from theories related to neural networks (Wiskott and Sejnowski 2002) and has since been correlated to the learning properties of human brain cells in the visual cortex (Berkes and Wiskott 2005; Wiskott and Sejnowski 2002). As described in chapter 2, SFA more recently found its way into the computer vision community (Franzius et al. 2008; Kompella et al. 2012; Nater et al. 2011; Zafeiriou et al. 2013).

In general, SFA extracts projections which consider the temporal information to find the most descriptive components that vary slowest over time (Wiskott and Sejnowski 2002). The intuition is linked to the assumption that the information contained in a signal do not change suddenly, but slowly, in sequential data. Note, an input signal generally contains high variation, often caused by noise. Nonetheless, it is the seldom varying features of the sequence that mark the separation between informative changes. SFA extracts these features, as it selects the important attributes of the video which change least when time moves forward.

This chapter is motivated by the work of Nater et al. (2011), which exploits the properties of SFA to segment video data temporally. In their study Nater et al. (2011) extract unknown different actions from an image sequence. Here, the individual segments are thought to be the activities in the examined video. Recall from chapter 2, Nater et al. (2011) perform SFA on the complete video and therefore require the whole sequence a priori. Notice, the system of Nater et al. (2011) works by detecting change in video data, as it uses the frame with most variance in the slow features as cutting point for the segmentation. It is an iterative offline setup. However, with an ever increasing importance of realizing online applications, I now wish to propose an online learning setup for SFA, which I apply to the application of temporal video segmentation and change detection.
The need for change detection stems from the results in previous chapters. In particular, previous chapters and the work of Babenko et al. (2011) show that online learned appearance models are commonly susceptible to drift during tracking. With change detection, and SFA in particular, a tracking system can detect when drift is likely (i.e. during changes). I want to combine online tracking systems with an incremental approach to SFA to improve drift suppression.

An incremental learning system to find slow features is required for detecting changes in video streams. However, the methods presented in chapter 2 require the complete video a priori. To my knowledge, the only incremental version of SFA in current literature is proposed by Kompella et al. (2012) and coined IncSFA. As SFA can be solved in two stages via PCA and minor component analysis (MCA), IncSFA utilizes a combination of the candid covariance-free incremental PCA (CCIPCA) by Weng et al. (2003), and the sequential extraction of minor components proposed by Chen et al. (2001). CCIPCA is based on statistical efficiency and incrementally estimates the data distribution by means of scale and mean. Consequently, IncSFA learns a rough estimation of the true slow features and converges to the true solution only over time. Because of this, IncSFA is designed to learn from multiple complete videos of similar motions, rather than the data points of a single video. Therefore, IncSFA is very suitable for learning features of sequences that reoccur in their entirety, e.g. complete activities. However, online learning for change detection, where a scene is seen only once, is not supported by this approach.

SFA is originally designed for data signals with quadratic expansions. In this case, since SFA is a statistical component analysis, it is quite likely to suffer from the so-called “curse of dimensionality.” Hierarchical SFA, as proposed by Franzius et al. (2008), somewhat improves this situation, but important properties could be missed. More recently, Böhmer et al. (2011) introduce kernel SFA (KSFA) for standard positive definite kernels. Similarly to a quadratic expansion, kernel methods allow for a mapping of features into a high-dimensional feature space. However, this feature space is never required to be computed explicitly, as it is realized by the kernel which represents the dot-product of two samples in such space. The computation of kernels is often more efficient than non-linear expansions – at least in the amount of computer memory used. Furthermore, kernel functions allow for more flexible feature spaces.

Typically, as also suggested by Böhmer et al. (2011), the selection of standard kernels such as GRBFs is encountered. As emphasized throughout this thesis, such kernels seldom utilize the domain-dependent properties of the data. In chapter 5, I present a kernel specifically designed for image gradients. Now, I aim to apply the same kernel function to SFA. However, as the kernel is not positive definite, an extension of KSFA into Krein space is required.
In the first part of this chapter, I introduce a general kernel framework for SFA with positive definite and indefinite kernels. Furthermore, I formulate an exact online learning algorithm for the proposed KSFA which computes the slow components at each given time step incrementally. Notice, in contrast to IncSFA by Kompella et al. (2012) which learns from multiple videos, the proposed incremental version of SFA in this chapter is designed to learn from individual data points of a single data stream.

Recall, the classifications of online kernel methods are commonly written as a weighted sum of kernel combinations, taken from a set of stored instances, usually referred to as the support or reduced set. At each step a new instance is fed to the algorithm and, depending on the update criterion, the algorithm adds the instance to the support set. One of the major challenges in online learning is that the support set may grow arbitrarily large over time (Chin and Suter 2007; Günter et al. 2007; Kim et al. 2005). Inspired by the incremental KPCA of Chin and Suter (2007), a reduced set expansion in Krein space is proposed, which is expressed through preimages, and designed to bound the memory requirements.

While a reduced set expansion with preimages allows for arbitrary kernels, the computation of preimages has drawbacks: (i) the reduced set representation provides only an approximation to the exact solution and (ii) the optimization problem for finding the expansion inevitably increases the complexity of the algorithm. Hence, in the second part of this chapter, the special properties of the kernel family also used in chapter 5 are utilized to formulate a special case of exact online KSFA without preimage computation. Finally, SFA’s first real-time change detection algorithm is proposed. This new method is then employed to temporal video segmentation and visual tracking with change detection. In summary, the contributions are:

1. An exact KSFA algorithm for arbitrary derivative approximations and any kernel in Hilbert or Krein space is proposed.

2. A framework for general online KSFA, for which a reduced set expansion in Krein space is employed, is introduced. Notice, in contrast to Kompella et al. (2012), the proposed online learning system computes the slow features at each time step, from individual data points of a single video sequence.

3. Incremental KSFA without reduced set expansion is then formulated, by exploiting the properties of the special domain-specific kernel from chapter 5.

4. Finally, SFA’s first online change detection is implemented and employed to temporal video segmentation and tracking with change detection.
Some related research in the broader area of unsupervised video segmentation is conducted by Hoai et al. (2011); Turaga et al. (2009); Zhou et al. (2010). Zhou et al. (2010) propose a method for clustering facial events. Their work is only suitable for offline processing and requires the number of clusters \textit{a priori}. Similarly, the clustering algorithm by Turaga et al. (2009) also requires the number of expected clusters. A method for joint segmentation and classification of human actions in video is proposed by Hoai et al. (2011). Their method is supervised, \textit{i.e.} a model for human actions is learned from a set of labeled training samples. Then, given a testing video with a continuous stream of human activities, the algorithm by Hoai et al. (2011) finds the globally optimal temporal segmentation (\textit{i.e.} the change points between actions) and class labels.

The proposed methodology in this chapter takes a different direction. In particular, it detects the temporal changes in video streams online. I emphasize, neither the number of clusters, nor any training to a predefined set of examples, is required before execution. Thus, the methods by Hoai et al. (2011); Turaga et al. (2009); Zhou et al. (2010) constitute excellent post-processing tools for clustering or classifying the events. In contrast to the proposed method, however, initial segmentation of unknown videos with unpredicted activities is not possible.

### 6.1 Slow Feature Analysis with Indefinite Kernels

In chapter 5 the concept of Krein space is discussed in relation to Hilbert space, denoted \( \mathcal{K} \) and \( \mathcal{H} \) respectively. Recall, both spaces are implicit and provide a hyper-dimensional vector space. A kernel function \( k : \mathbb{C}^P \times \mathbb{C}^P \rightarrow \mathbb{C} \) defines the unknown mapping, denoted \( \phi_\mathcal{H} : \mathbb{C}^P \rightarrow \mathcal{H} \) for \( \mathcal{H} \) or \( \phi_\mathcal{K} : \mathbb{C}^P \rightarrow \mathcal{K} \) for \( \mathcal{K} \) which transforms the original data into the relevant space. While kernels in Hilbert space are semi-positive definite, for kernels in Krein space, the fundamental symmetry \( J \) is used to find an associated Hilbert space, denoted \( |\mathcal{K}| \), \emph{via} a conjugate operator denoted \((\cdot)^*\), as introduced in the previous chapter. For many applications, the implicit feature space \( \phi \) is never explicitly required as the kernel \( k \) provides the necessary computations (Böhmer et al. 2011; Schölkopf et al. 1997; Shawe-Taylor and Cristianini 2004). For example, the linear distance between two samples in the non-linear vector space is given as

\[
d(\phi(z_i), \phi(z_j))^2 = k(z_i, z_i) - k(z_i, z_j) - k(z_j, z_i) + k(z_j, z_j). \tag{6.1}
\]

I now wish to apply the indefinite Krein kernel from the previous chapter to SFA, due to its success in visual applications. SFA and its kernelized optimization problem is presented in the following. Then, the algorithm for SFA with kernels in Hilbert and Krein
space is proposed. It is important to note here, that methods which employ Krein space are also valid for Hilbert space, as $\mathcal{K} \supset \mathcal{H}$.

### 6.1.1 SFA’s Optimization Problem

Let me reiterate the optimization problem of SFA in the following. Given $N$ sequential observation vectors $\{x_n\}_{n=1}^{N}$ as columns of the sample matrix $X = \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix} \in \mathbb{C}^{P \times N}$, SFA finds a descriptive output vector $o_n$ for each $x_n$ such that the output signal $O = \begin{bmatrix} o_1 & \cdots & o_N \end{bmatrix} \in \mathbb{C}^{M \times N}$ changes slowest over time (Wiskott and Sejnowski 2002).\(^1\)

The output of each individual sample is formed as the concatenation of $M$ mappings $\{f_m\}_{m=1}^{M}$, where $f_m : \mathbb{C}^{P} \rightarrow \mathbb{C}$ transforms the input samples, such that the output vector for $x_n$ becomes $o_n = \left[ f_1(x_n) \quad \cdots \quad f_M(x_n) \right]^T$. The slowness for function $f_m$ is given by

$$\Delta(f_m) \triangleq \frac{1}{N} \sum_{n=1}^{N} |\dot{f}_m(x_n)|^2.$$  \hfill (6.2)

Hence, SFA minimizes the time derivative $\dot{f}_m$ of function $f_m$ for the whole video sequence. Often, as shown by Böhmer et al. (2011); Nater et al. (2011), $\dot{f}_m$ is represented as the difference between consecutive time steps, i.e. $\dot{f}_m(x_n) = f_m(x_n) - f_m(x_{n-1})$. However, any derivative may be utilized.

Wiskott and Sejnowski (2002) introduce further constraints to avoid the trivial solution and prevent information redundancy: The output signals of each mapping $f_m$, denoted here as $f_m = \left[ f_m(x_1) \quad \cdots \quad f_m(x_N) \right]^T$, are required to have zero mean (eq. (6.3)) and unit variance (eq. (6.4)). Moreover, the vectors $f_m$ are constrained to be uncorrelated (eq. (6.5)). Finally, an ordering of the components according to their slowness is employed (eq. (6.6)). In summary

$$\forall m \quad f_m^H 1_{N \times 1} = 0 \quad \hfill (6.3)$$
$$\forall m \quad f_m^H f_m = 1 \quad \hfill (6.4)$$
$$\forall m \neq m' \quad f_m^H f_{m'} = 0 \quad \hfill (6.5)$$
$$\forall m < m' \quad \Delta(f_m) < \Delta(f_{m'}) \quad \hfill (6.6)$$

where $1_{a \times b}$ is an $a \times b$ matrix with all elements equal to 1.

Often, the features of the samples in $X \in \mathbb{C}^{P \times N}$ are assumed to be linear and directly taken from the input values in $Z = \begin{bmatrix} z_1 & \cdots & z_N \end{bmatrix} \in \mathbb{C}^{P \times N}$, i.e. $x_n = z_n$ where $P = P'$.

\(^1\)The original SFA is defined only in the real valued space, however, an extension into complex space is trivially achieved by replacing the transpose operation with the Hermitian transpose.
or a result of a nonlinear expansion is used, e.g. a quadratic expansion where \( z_n \in \mathbb{R}^P \) is transformed by

\[
    x_n = \begin{bmatrix}
        z_n(1) \\
        \vdots \\
        z_n(P) \\
        z_n(1)z_n(1) \\
        \vdots \\
        z_n(1)z_n(P) \\
        \vdots \\
        z_n(P)z_n(P)
    \end{bmatrix},
\]

where \( P' = P + P\frac{P+1}{2} \), and \( a(b) \) denotes the \( b \)th element of vector \( a \). Generally, however, any mapping \( \phi : \mathbb{C}^P \to \mathcal{K} \), such that \( x_n = \phi(z_n) \), may be utilized.

Berkes and Wiskott (2005) show how SFA can be solved by means of the generalized eigenvalue problem. In particular, let the scatter of the data be

\[
    S = \sum_{n=1}^{N}(x_n - \mu)(x_n - \mu)^H, \tag{6.8}
\]

where \( \mu \) is the sample mean. The scatter of the data’s derivative is

\[
    \dot{S} = \sum_{n=1}^{N}(\dot{x}_n - \dot{\mu})(\dot{x}_n - \dot{\mu})^H, \tag{6.9}
\]

where \( \dot{x}_n \) is the derivative of \( x_n \), and \( \dot{\mu} \) is the derivatives’ mean. Then, SFA finds a projection matrix \( V = \begin{bmatrix} v_1 & \cdots & v_M \end{bmatrix} \in \mathbb{C}^{P' \times M} \) such that

\[
    V = \arg \min_{\tilde{V}} \text{tr} \left( (\tilde{V}^H S \tilde{V})^{-1} \tilde{V}^H S \tilde{V} \right), \tag{6.10}
\]

where \( \text{tr}(.) \) computes the trace of a matrix. After ordering, the functions in \( \{f_m\}_{m=1}^{M} \) are realized by \( f_m(x_n) = v_m^H(x_n - \mu - \dot{\mu}) \).

### 6.1.2 Solving Kernel SFA in Krein Space

In this section, SFA is formulated as a general kernel problem. Let \( \phi : \mathbb{C}^P \to \mathcal{K} \) be an unknown mapping into Krein space whose inner product is equivalent to a known kernel, \( k : \mathbb{C}^P \times \mathbb{C}^P \to \mathbb{C} \). Notice, although the notation includes mapping \( \phi \) to describe the method, the implicit function is never evaluated, as \( k \) is employed.
Consider the mapped sample matrix $X = \left[ \phi(z_1) \cdots \phi(z_N) \right]$ as the implicitly given matrix of the original features in $Z$. The samples’ mean and centralized matrix can be respectively computed by

$$\mu = \frac{1}{N}X1_{N \times 1} = XM$$  \hspace{1cm} (6.11)$$
$$\tilde{X} = X \left( I_N - \frac{1}{N}1_{N \times N} \right) = XC,$$ \hspace{1cm} (6.12)

where $M = \frac{1}{N}1_{N \times 1}$ and $C = I_N - \frac{1}{N}1_{N \times N}$ (Chin and Suter 2007; Pękalska and Haasdonk 2009). Furthermore, let us define the total scatter matrix by

$$S = \sum_{n=1}^{N} (\phi(z_n) - \mu)(\phi(z_n) - \mu)^* = \tilde{X}\tilde{X}^* = \tilde{X}\tilde{X}^HJ$$  \hspace{1cm} (6.13)

for some fundamental symmetry matrix $J$. Note, this matrix describes the scatter in the associated Hilbert space $|K|$, similar to the proposed PCA with Krein kernels in chapter 5.

Similarly to the work of Böhmer et al. (2011); Nater et al. (2011), the signal derivative is expressed as

$$\dot{X} = X \begin{bmatrix} 0 & -1 & 0 & \cdots & 0 & \cdots & 0 & -1 \end{bmatrix} = XD$$ \hspace{1cm} (6.14)

where $D \in \mathbb{R}^{N \times N}$ describes the backward difference. However, a more general representation of the derivative may be desired. In fact, the derivative may be represented as the product of any two matrices $\dot{X} = [\phi(\dot{z}_1) \cdots \phi(\dot{z}_N)]$ and $D \in \mathbb{R}^{N \times N}$ such that $\dot{X} = XD$. The derivatives’ mean $\dot{\mu}$ and centralized matrix $\tilde{\dot{X}}$ are respectively found analogously to eq. (6.11) and eq. (6.12), and a scatter matrix of the derivatives is define by

$$\dot{S} = \tilde{\dot{X}}\tilde{\dot{X}}^* = [XDC][C^T D^TX^H]J.$$  \hspace{1cm} (6.15)

Based on the KSFA in Hilbert space by Böhmer et al. (2011) and KPCA in Krein space in chapter 5, KSFA’s optimization problem in Krein space is provided by

$$V = \arg \min_{\tilde{V}} \operatorname{tr} \left( (\tilde{\tilde{V}}^*S\tilde{V})^{-1} \tilde{\tilde{V}}^*\dot{S}\tilde{V} \right).$$  \hspace{1cm} (6.16)
By formulating the projection as linear combination, i.e. \( \mathbf{V} = \hat{\mathbf{X}} \hat{\mathbf{V}} \), eq. (6.16) becomes

\[
\hat{\mathbf{V}} = \arg \min_{\mathbf{V}} \text{tr} \left( \left( \mathbf{V}^H \hat{\mathbf{X}}^* \hat{\mathbf{X}}^T \hat{\mathbf{X}} \mathbf{V} \right)^{-1} \mathbf{V}^H \hat{\mathbf{X}}^* \hat{\mathbf{X}} \mathbf{V} \right)
= \arg \min_{\mathbf{V}} \text{tr} \left( \left( \mathbf{V}^H \tilde{\mathbf{K}}^2 \mathbf{V} \right)^{-1} \mathbf{V}^H \mathbf{K} \mathbf{C} \mathbf{D} \mathbf{C}^T \mathbf{K}^H \mathbf{V} \right) \quad (6.17)
\]

where \( \mathbf{K} = \mathbf{X}^\ast \mathbf{X} \) is the kernel matrix, and \( \tilde{\mathbf{K}} = \hat{\mathbf{X}}^* \hat{\mathbf{X}} = \mathbf{C}^T \mathbf{K} \mathbf{C} \) its centralized version. The optimum of eq. (6.17) is found via a two step approach. First, ensure that

\[
\hat{\mathbf{V}}^H \tilde{\mathbf{K}}^2 \hat{\mathbf{V}} = \mathbf{I} \quad (6.18)
\]
and then find the optimum for

\[
\hat{\mathbf{V}} = \arg \min_{\mathbf{V}} \text{tr} \left( \hat{\mathbf{V}}^H \mathbf{K} \mathbf{C} \mathbf{D} \mathbf{C}^T \mathbf{K}^H \mathbf{V} \right). \quad (6.19)
\]

Eq. (6.18) is achieved by whitening the squared kernel matrix \( \tilde{\mathbf{K}}^2 \). For this, let us compute the eigenvalue decomposition, \( \tilde{\mathbf{K}}^2 = \mathbf{\Omega} \mathbf{\Lambda}^2 \mathbf{\Omega}^H \), and define \( \hat{\mathbf{V}} = \mathbf{W} \mathbf{\Theta} \) such that \( \mathbf{W} = \mathbf{\Omega} |\mathbf{\Lambda}|^{-1} \) and \( \mathbf{\Theta}^H \mathbf{\Theta} = \mathbf{I} \) holds (eigenvalues with zero magnitude and the corresponding eigenvectors are implicitly removed). Then we obtain

\[
\hat{\mathbf{V}}^H \tilde{\mathbf{K}}^2 \hat{\mathbf{V}} = \mathbf{\Theta}^H \mathbf{W}^H \tilde{\mathbf{K}}^2 \mathbf{W} \mathbf{\Theta}
= \mathbf{\Theta}^H |\mathbf{\Lambda}|^{-1} \mathbf{\Omega}^H \mathbf{\Lambda}^2 \mathbf{\Omega}^H |\mathbf{\Lambda}|^{-1} \mathbf{\Theta}
= \mathbf{\Theta}^H |\mathbf{\Lambda}|^{-1} \mathbf{\Lambda}^2 |\mathbf{\Lambda}|^{-1} \mathbf{\Theta}
= \mathbf{\Theta}^H \mathbf{\Theta} = \mathbf{I}. \quad (6.20)
\]

Eq. (6.19) is now reformulated as

\[
\mathbf{\Theta} = \arg \min_{\mathbf{\Theta}} \text{tr} \left( \hat{\mathbf{\Theta}}^H \mathbf{W}^H \mathbf{K} \mathbf{C} \mathbf{D} \mathbf{C}^T \mathbf{K}^H \mathbf{W} \hat{\mathbf{\Theta}} \right)
\text{subject to } \mathbf{\Theta}^H \mathbf{\Theta} = \mathbf{I} \quad (6.21)
\]

which is solved via the eigenvalue decomposition of \( \mathbf{W}^H \mathbf{K} \mathbf{C} \mathbf{D} \mathbf{C}^T \mathbf{K}^H \mathbf{W} = \mathbf{\Theta} \mathbf{\Sigma} \mathbf{\Theta}^H \). Finally, the sought projection is provided by \( \mathbf{V} = \hat{\mathbf{X}} \hat{\mathbf{V}} = \hat{\mathbf{X}} \mathbf{W} \mathbf{\Theta} \).

Often, high dimensional data yields some dimensions which virtually never vary. Unfortunately, in these circumstances such data points are likely to provide the slowest features. Therefore, a dimensionality reduction on the scatter matrix of the data is often advantageous. The eigenvalue decomposition of \( \tilde{\mathbf{K}}^2 \) is ideal for this task, as the eigenvectors in \( \mathbf{\Omega}_R \in \mathbb{C}^{N \times R} \) related to the largest absolute eigenvalues in \( |\mathbf{\Lambda}_R| \in \mathbb{R}^{R \times R} \) can be extracted (here the notation \( |\mathbf{\Lambda}| \triangleq \mathbb{R}(\mathbf{\Lambda}) \) is to mean the diagonal matrix containing...
Algorithm 6.1 Batch SFA in Krein Space

**Input:** The sequential training data \( Z \in \mathbb{C}^{P \times N} \), the derivative matrix \( D \in \mathbb{R}^{N \times N} \), the number of components kept after whitening \( R \), and the kernel function \( k : \mathbb{C}^P \times \mathbb{C}^P \to \mathbb{C} \).

**Output:** The data projection \( \tilde{V} \) with sorted components according to slowness.

1. Compute the kernel matrix \( K = \bar{X}^* \bar{X} \), utilizing \( k \).
2. Find \( C^H K C = \Omega \Lambda \Omega^H \).
3. Form the reduced set \( \Omega_R \in \mathbb{C}^{N \times R} \) and \( \Lambda_R \in \mathbb{R}^{R \times R} \) which is related to the \( R \) eigenvalues with largest magnitude in \( |\Lambda| \).
4. Set \( W_R = \Omega_R |\Lambda_R|^{-1} \).
5. Compute \( W_R^H C^T K D \bar{C}^T K W_R = \Theta \Sigma \Theta^H \).
6. Reorganize the components in \( \Theta \) in relation to the ascending eigenvalues in \( \Sigma \).
7. Set \( \tilde{V} = W_R \Theta \).

Algorithm 6.2 Testing SFA in Krein Space

**Input:** The to-be-tested sample \( z \in \mathbb{C}^P \), the data projection \( \tilde{V} \) with sorted components, the number \( M \) of slow features to be used, the training data \( Z \in \mathbb{C}^{P \times N} \), the derivative matrix \( D \in \mathbb{R}^{N \times N} \), and the kernel function \( k : \mathbb{C}^P \times \mathbb{C}^P \to \mathbb{C} \).

**Output:** The output signal \( o \in \mathbb{C}^M \).

1. Compute \( \dot{o} = \tilde{X}^* x - \tilde{X}^* \mu - \tilde{X}^* \dot{\mu} \), where \( x = \phi(z) \), using the kernel \( k \).
2. Find \( o = \tilde{V}_M^H \dot{o} \) where \( \tilde{V}_M \in \mathbb{C}^{N \times M} \) consists of the first \( M \) rows of \( \tilde{V} \).

When arbitrary derivatives are used, projecting the scatter matrix of the derivatives \( (i.e. W^H K D \bar{C}^T K^H W) \) may introduce a new null-space, which will inadvertently result in incorrect slow feature functions. Fortunately, when the derivation is approximated directly from sample data (as is the case for the backward difference in eq. (6.14)) this is not happening above. However, if the components in \( S \) are insufficient, an alternative optimization problem can be employed. As discussed by Fukunaga (1990); Kim et al. (2007), equivalent to the optimization problem in eq. (6.10) is the solution of

\[
V = \arg \min_V \text{tr} \left( (\tilde{V}^H \bar{S} \tilde{V})^{-1} \tilde{V}^H \bar{S} \tilde{V} \right), \tag{6.22}
\]

where \( \bar{S} = \bar{X} \bar{X}^* \) represents the total scatter matrix of the samples and their derivative.
The total sample matrix $\bar{X}$ is computed as

$$\bar{X} = \begin{bmatrix} X & \dot{X} \end{bmatrix} \begin{bmatrix} I_N & 0 \\ 0 & D \end{bmatrix} \left( I_{2N} - \frac{1}{2N} I_{2N \times 2N} \right).$$  \hfill (6.23)

Thus for this special case, simple substitution of $S$ by $\bar{S}$ yields the solution. Note, the implicit mapping remains not required, as the kernel function $k$ can be used analogously to before. By solving eq. (6.22), it is ensured that the projection $W$ does not introduce a null-space, as the data derivatives are considered when $W$ is formed.

### 6.2 A Framework for Online Kernel SFA

Online KSFA that updates from novel data points is now introduced. First, online SFA with arbitrary Krein space kernels is formulated. A reduced set representation is required to employ general kernels. Then, a special family of kernels is presented, which allows for a representation which does not required a reduced set expansion.

#### 6.2.1 General Online Kernel SFA

As seen in section 6.1, SFA can be solved in two stages, the data whitening in eq. (6.18) and the decomposition in eq. (6.19). In the following, it is detailed how both steps are computed incrementally. Figure 6.1 illustrates the incremental setup. The notation is as before, while time steps and new data is indicated by subscripts (e.g. $X_t$ is the implicitly mapped sample matrix at time $t$, and $N_\delta$ is the number of new samples).

**Indefinite KPCA Update for Online Whitening**

The update of the whitening projections can be performed with online KPCA. Thus, the incremental KPCA by Chin and Suter (2007) is extended to allow for indefinite kernels in Krein space.

Let $X_\delta$ be the matrix containing the mapping of the new samples in $\{z_n\}_{n=N_{t-1}+1}^{N_t}$, such that $X_t = \begin{bmatrix} X_{t-1} & X_\delta \end{bmatrix}$ provides the updated data set. The eigenvectors $\Omega_{t-1}$ and the eigenvalue magnitudes $|\Lambda_{t-1}|$ of the previous time step $t - 1$ is to be updated to find the new whitening projection $W_t = \Omega_t |\Lambda_t|^{-1}$ which incorporates the additional information of the new samples.

Similarly to eq. (6.11) and eq. (6.12), let us denote the new data’s mean as $\mu_\delta = X_\delta M_\delta$ and its centered sample matrix as $\tilde{X}_\delta = X_\delta C_\delta$, where $M_\delta = \frac{1}{N_\delta} I_{N_\delta \times N_\delta}$ and $C_\delta = I_{N_\delta} - \frac{1}{N_\delta} I_{N_\delta \times N_\delta}$ (or equivalently the respective version for method based on the total scatter)
Figure 6.1: Illustration of incremental SFA at initialization (time 0) and time step t. Incremental kernel PCA and the incremental derivative projection is combined to find the slow features at each time-step. Details are provided in the text.

matrix $\tilde{S}_t$). For the sake of simplification, let us first assume the data mean is unchanged (i.e. $\mu_{t-1} = \mu_\delta$). We define the kernel matrix of the new data as $\tilde{K}_t = \tilde{X}_t \tilde{X}_\delta$ and find the eigenvalue decomposition $\tilde{K}_t^2 = \Omega_t \Lambda_t^2 \Omega_t^H$. Then in respect to eq. (6.18), the eigenvalue decomposition of $\tilde{K}_t^2$ is to be found, i.e.

$$
\tilde{K}_t^2 = (\tilde{X}_t \tilde{X}_\delta)^* (\tilde{X}_t \tilde{X}_\delta)^2 = \hat{\Omega}_t \left[ \begin{array}{c|c} \Lambda_t \Lambda_t^H & \Omega_t \hat{X}_t \tilde{X}_\delta \end{array} \right] = \hat{\Omega}_t \hat{\Omega}_t^H = \Omega_t \Lambda_t^2 \Omega_t^H,
$$

(6.24)

where $\hat{\Omega}_t = \left[ \begin{array}{c} \Omega_{t-1} \ 0 \\ 0 \ \Omega_\delta \end{array} \right]$ is an orthogonal matrix, composed of the eigenvectors in $\Omega_{t-1}$ and $\Omega_\delta$. Notice, eq. (6.24) can be solved by utilizing the decomposition of the inner matrix, $\hat{\Omega}_t \Lambda_t^2 \hat{\Omega}_t^H$. The size of this matrix is independent of the sample number. The projected eigenvectors provide the solution, as

$$
\Omega_t = \left[ \begin{array}{c} \Omega_{t-1} \ 0 \\ 0 \ \Omega_\delta \end{array} \right] \hat{\Omega}_t = \hat{\Omega}_t \hat{\Omega}_t.
$$

(6.25)
Thus, the new projection is given by $W_t = \Omega_t|\Lambda_t|^{-1}$ (zero eigenvalues and their eigenvectors are removed implicitly). A dimensionality reduction may be applied as outlined in section 6.1.2. It is important to note that $\Omega_t$ may grow arbitrarily large over time. Later, I will detail how this problem is overcome by introducing a reduced set representation that bounds the growth by means of approximation.

I now continue the analysis. Thus far, only the case with an unchanged data mean is considered. A simple modification of $\tilde{X}_\delta$ allows for changing means (Chin and Suter 2007; Ross et al. 2008). In particular, an additional data point is added to the new samples, whose sole purpose is the correction of the mean subtraction

$$\tilde{X}_\delta' = \left[ \begin{array}{c} \tilde{X}_\delta \\ \sqrt{\frac{N_{t-1}N_{\delta}}{N_{t-1}+N_{\delta}}}(\mu_{t-1} - \mu_\delta) \end{array} \right]$$

$$= X_t \left[ \begin{array}{c} 0 \\ C_\delta \end{array} \right] \sqrt{\frac{N_{t-1}N_{\delta}}{N_{t-1}+N_{\delta}}} \left[ \begin{array}{c} M_{t-1} \\ -M_\delta \end{array} \right].$$

(6.26)

Now, eq. (6.24) is computed with $\tilde{X}_\delta'$ in place of $\tilde{X}_\delta$. Finally, the mean of the overall data is updated by

$$\mu_t = \frac{N_{t-1}}{N_{t-1}+N_{\delta}}X_{t-1}M_{t-1} + \frac{N_{\delta}}{N_{t-1}+N_{\delta}}X_\delta M_\delta$$

$$= X_t M_t$$

(6.27)

where $M_t = \frac{1}{N_{t-1}+N_{\delta}} \left[ \begin{array}{c} N_{t-1}M_{t-1} \\ N_\delta M_\delta \end{array} \right]$.

It is important to note that the proposed complete incremental update never requires the explicit calculation of the unknown mapping $\phi$, as the kernel trick using the kernel $k$ is employed. Furthermore, ignoring the growth of the kernel support set, the framework for updating general KPCA in Krein space computes in constant time and memory. Later, I introduce a reduced set representation for constant execution of the whole algorithm. First, however, I complete the online KSFA framework with the second part of the algorithm, related to eq. (6.19).

**Slow Feature Update**

After finding the whitening matrix, the slow features in $\tilde{V}_{t-1} = W_{t-1}\Theta_{t-1}$ require update. More precisely, the eigenvalue decomposition of the projected scatter matrix, $W_{t-1}^H\tilde{X}_{t-1}^H\tilde{S}_{t-1}\tilde{X}_{t-1}W_{t-1} = \Theta_{t-1} \Sigma_{t-1} \Theta_{t-1}^H$, needs to be renewed with regards to the new derivatives and the projection, which are denoted by $\tilde{X}_\delta$ and $W_t$ respectively.
Again, let us first assume unchanged means (i.e. $\bar{\mu}_{t-1} = \bar{\mu}_{\delta}$). The new projection of the new scatter matrix is then provided as

$$W_t^H \tilde{X}_t \hat{S}_{t-1} \tilde{X}_t W_t = W_t^H \tilde{X}_t \hat{S}_{t-1} \tilde{X}_t W_t + W_t^H \tilde{X}_t \hat{S}_{\delta} \tilde{X}_t W_t$$

$$= W_t^H \begin{bmatrix} \tilde{X}_t^* \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \tilde{X}_t \nu_{t-1} \nu_{t-1} \nu_\delta \end{bmatrix}^H W_t$$

$$+ W_t^H \tilde{X}_t^* \tilde{X}_t \nu_\delta \tilde{X}_t^* \tilde{X}_t \nu_\delta \tilde{X}_t W_t$$

(6.28)

where $\hat{S}_\delta$ is the scatter of the new derivatives.

The second term of eq. (6.28) can be directly computed, as the new data points in $\tilde{X}_\delta$ are readily available. More complicated, however, is the calculation of the first term when constant running time is required for online learning. Notice that $\tilde{X}_t W_t$ can be expressed through the samples’ subspaces for $\tilde{X}_{t-1}$ and $\tilde{X}_\delta$, given by $U_{t-1} = \tilde{X}_{t-1} \Omega_{t-1} |\Lambda_{t-1}|^{-\frac{1}{2}}$ and $U_\delta = \tilde{X}_\delta \Omega_\delta |\Lambda_\delta|^{-\frac{1}{2}}$ respectively. Therefore, let us write

$$\tilde{X}_t W_t = \begin{bmatrix} U_{t-1} |\Lambda_{t-1}|^{\frac{1}{2}} & U_\delta |\Lambda_\delta|^{\frac{1}{2}} \end{bmatrix} \tilde{\Omega}_t |\Lambda_t|^{-\frac{1}{2}}.$$ (6.29)

It is reasonable to assume that $\tilde{X}_{t-1}$ is well represented by the subspace $U_{t-1}$ alone, as the projection’s components were selected to reproduce both $\tilde{X}_{t-1}$ and $\tilde{X}_\delta$ (or $\tilde{X}$ respectively). The new components which are introduced by the new samples in $\tilde{X}_\delta$ are unlikely to be of significance to $\tilde{X}_{t-1}$. Hence their contribution is omitted,\(^2\) and the first term of eq. (6.28) is rewritten as

$$W_t^H \tilde{X}_t^* \hat{S}_{t-1} \tilde{X}_t W_t = |\Lambda_t|^{-\frac{1}{2}} \tilde{\Omega}_t^H \tilde{X}_{t-1}^* \hat{S}_{t-1} \tilde{X}_t W_t$$

$$= |\Lambda_t|^{-\frac{1}{2}} \tilde{\Omega}_t^H \begin{bmatrix} \Omega_{t-1} & 0 \end{bmatrix}^H \tilde{X}_{t-1}^* \hat{S}_{t-1} \tilde{X}_t W_t$$

$$= P_t^H W_{t-1}^H \tilde{X}_{t-1}^* \hat{S} \tilde{X}_{t-1}^* W_{t-1} P_t,$$ (6.30)

where $P_t \in \mathbb{C}^{R_t \times R_t}$ is a matrix given by

$$P_t = \begin{bmatrix} |\Lambda_{t-1}| & 0 \end{bmatrix} \tilde{\Omega}_t |\Lambda_t|^{-\frac{1}{2}}$$ (6.31)

and $\tilde{\Omega}_t$ corresponds to the non-zero eigenvalues in $\Lambda_t$.

Let us represent the old data mean in its projected form, $W_{t-1}^H \tilde{X}_{t-1}^* \hat{\mu}_{t-1}$. Using $P_t$, its update is approximated by

$$W_t^H \tilde{X}_t^* \hat{\mu}_{t-1} \approx P_t^H W_{t-1}^H \tilde{X}_{t-1}^* \hat{\mu}_{t-1}.$$ (6.32)

\(^2\)The forgetting factor, introduced in the following section, reduces the effect of errors even further.
Similarly to eq. (6.26), the step to correct the mean substraction is performed by adding a correction sample to the new data, or, equivalently, by adding the following to eq. (6.28)

$$\frac{N_{t-1}N_\delta}{N_{t-1} + N_\delta} \mathbf{W}_t^H \tilde{\mathbf{X}}_t^* (\tilde{\mu}_{t-1} - \tilde{\mu}_\delta)(\tilde{\mu}_{t-1} - \tilde{\mu}_\delta)^* \tilde{\mathbf{X}}_t \mathbf{W}_t$$  \hspace{1cm} (6.33)

and the new mean is provided by

$$\mathbf{W}_t^H \tilde{\mathbf{X}}_t^* \tilde{\mu}_t = \frac{N_{t-1} \mathbf{W}_t^H \tilde{\mathbf{X}}_t^* \tilde{\mu}_{t-1} + N_\delta \mathbf{W}_t^H \tilde{\mathbf{X}}_t^* \tilde{\mu}_\delta}{N_{t-1} + N_\delta}.$$  \hspace{1cm} (6.34)

Finally, after computing the new projected scatter matrix \( \mathbf{W}_t^H \tilde{\mathbf{X}}_t^* \tilde{\mathbf{S}}_t \tilde{\mathbf{X}}_t \mathbf{W}_t \), the slow features are obtained through eigenvalue decomposition as described in section 6.1.2. Note, the size of this matrix is bound by the number of components in \( \mathbf{W}_t \). Hence, the computation time and memory requirements remain constant.

**Forgetting Factor**

In many online systems it is beneficial to attach a higher weight to recent data. A common approach to moderate the balance between old and new data is a forgetting factor, also employed by Levy and Lindenbaum (2000); Ross et al. (2008). This value acts as a weight on the known data, and reduces their impact by a factor \( w \in [0, 1] \) (usually \( 0 \ll w < 1 \)). In the following, I describe how \( w \) is applied to both steps of the incremental update.

Essentially, it is desired to reduce the effect of the values in \( \mathbf{X}_{t-1} \) by \( w \). In eq. (6.24), old and new data is combined via the squared kernel matrix \( \left[ \begin{array}{cc} \tilde{\mathbf{X}}_{t-1} & \tilde{\mathbf{X}}_\delta \\ \tilde{\mathbf{X}}_{t-1}^* & \tilde{\mathbf{X}}_\delta^* \end{array} \right]^2 \). The forgetting factor can be directly applied to the eigenvalues of the old data, as we find

$$\left( w \tilde{\mathbf{X}}_{t-1}^* \tilde{\mathbf{X}}_{t-1} w \right)^2 = \Omega_{t-1} (w^2 \Lambda_{t-1})^2 \Omega_{t-1}^H.$$  \hspace{1cm} (6.35)

Hence, eq. (6.24) can be modified to reflect \( w \) as follows

$$\tilde{\mathbf{K}}_t^2 = \tilde{\Omega}_t \left[ \begin{array}{cc} w^2 |\Lambda_{t-1}| & \Omega_{t-1}^H \tilde{\mathbf{X}}_{t-1}^* \tilde{\mathbf{X}}_{t-1} \Omega_{t-1} \delta \\ \Omega_{\delta}^H \tilde{\mathbf{X}}_{\delta} \tilde{\mathbf{X}}_{t-1} \Omega_{t-1} \delta \\ |\Lambda_{\delta}| \end{array} \right]^2 \tilde{\Omega}_t^H.$$  \hspace{1cm} (6.36)

The forgetting factor also affects the old data’s mean \( \tilde{\mu}_{t-1} \), which is changed by multiplying \( w \). Analogously to Ross et al. (2008), the mean is updated using

$$\mathbf{M}_t = \frac{1}{w N_{t-1} + N_\delta} \left[ \begin{array}{c} w N_{t-1} \mathbf{M}_{t-1} \\ N_\delta \mathbf{M}_\delta \end{array} \right].$$  \hspace{1cm} (6.37)
Finally, the correction term for the mean subtraction in $\tilde{X}_\delta'$ (in eq. (6.26)) is also modified to reflect $w$. In particular, using the derivation in appendix B, $\tilde{X}_\delta'$ is now given by

$$
\tilde{X}_\delta' = X_t \left[ \begin{array}{c} 0 \\ C_\delta \end{array} \right] \sqrt{w^2 N_{t-1} N_\delta (N_{t-1} + N_\delta)} \begin{bmatrix} M_{t-1} \\ -M_\delta \end{bmatrix}.
$$

(6.38)

Similar modifications are employed for the second stage of the update. With analogous derivations to above, we find

$$
W_t^H \tilde{X}_t^* S_t \tilde{X}_t W_t = \frac{w^2 N_{t-1} N_\delta (N_{t-1} + N_\delta)}{(w N_{t-1} + N_\delta)^2} W_t^H \tilde{X}_t^* (\dot{\mu}_{t-1} - \dot{\mu}_\delta)(\dot{\mu}_{t-1} - \dot{\mu}_\delta)^* \tilde{X}_t W_t \\
+ w^2 W_t^H \tilde{X}_t^* S_{t-1} \tilde{X}_t W_t + W_t^H \tilde{X}_t^* \tilde{X}_t W_t,
$$

(6.39)

which can be computed as described above. The mean update is now given by

$$
W_t^H \tilde{X}_t^* \dot{\mu}_t = \frac{w N_{t-1} W_t^H \tilde{X}_t^* \dot{\mu}_{t-1} + N_\delta W_t^H \tilde{X}_t^* \dot{\mu}_\delta}{w N_{t-1} + N_\delta}.
$$

(6.40)

Finally, the number of elements at time $t$ is set to $N_t = w N_{t-1} + N_\delta$.

**Introducing Constant Running Time**

Like most learning systems which employ the kernel trick, the proposed online SFA with an arbitrary Krein space kernel depends on a support set of the previously encountered data. One of the major challenges is that this set may grow to become arbitrarily large over time (Chin and Suter 2007; Günter et al. 2007; Kim et al. 2005). For example, whenever the slow features in $V_t$ are to be applied to a new sample $z$, the kernel matrix $\tilde{X}_t^* \phi(z)$ is computed, which requires all data points in the sample matrix $Z_t$. This, however, violates the online learning requirements of bound running speed, as $Z_t$ grows at every update. In the following, the reduced set expansion by Chin and Suter (2007) is adopted, to ensure constant running time. While I describe the main idea of this approach, I refer to the work of Chin and Suter (2007) for details. The reduced set is applied to the PCA step of the SFA calculation.

The proposed algorithm uses the support set $Z_t$ in combination with the eigenvectors $\Omega_t$ in the form of $\tilde{X}_t \Omega_t$. Therefore, as properties of orthogonality are exploited, the reduced set expansion is based on the subspace of the implicit samples in $\tilde{X}_t$, i.e. $\tilde{U}_t = \tilde{X}_t \Omega_t |\Lambda_t|^{-\frac{1}{2}}$. Each principal component in $\{u_r\}_{r=1}^R$ depends on the complete set of previously encountered data in $Z_t$. In particular, each component is realized by computing $u_r = \tilde{X}_t \omega_r |\lambda_r|^{-\frac{1}{2}}$, where $\omega_r$ is the component of $\Omega_r$, related to the $r^{th}$ eigenvalue, denoted...
\[ \hat{z}_{r(q+1)} = \arg\max_{\tilde{z}} \frac{\left( u^*_r - w_{rq}^H \hat{X}^*_r \right) \phi(\tilde{z})^2}{\phi(\tilde{z})^* \phi(\tilde{z})} \quad (6.41) \]

and the optimal weighting as (Chin and Suter 2007; Schölkopf et al. 1999)

\[ w_{r(q+1)} = \left( \hat{X}^*_r \hat{X}^*_{r(q+1)} \right)^{-1} \hat{X}^*_{r(q+1)} u_r. \quad (6.42) \]

Analogously, an additional set of at most \( Q \) preimages is found to approximate the sample

\[ \lambda_r, \text{ at time } t. \] The basic idea behind the reduced set expansion is depicted in figure 6.2 and taken from Chin and Suter (2007). For each component \( u_r \), a set of at most \( Q \) preimages \( \{ \hat{z}^{(r)}_q \}_{q=1}^Q \) is found, whose implicit mappings, denoted \( \hat{X}_r = [\phi(\hat{z}^{(r)}_1) \ldots \phi(\hat{z}^{(r)}_Q)] \), best approximate the optimized eigenvector, \( i.e. u_r \approx \hat{X}_r w_r \). Here \( \{ w_r \}_{r=1}^R \) is an optimized multiplier which controls the weighting of the samples for each eigenvector.

Chin and Suter (2007) propose a two stage greedy search to find the components. Intuitively, the best preimage to describe the remaining data is that which carries most information to reproduce \( u_r \). For convenience, let me denote the mapped set of the first \( q \) samples as \( \hat{X}_{rq} = [\phi(\hat{z}_1) \ldots \phi(\hat{z}_q)] \) and the weighting of the \( q \) elements as \( w_{rq} \in \mathbb{R}^q \). The \((q+1)\)th element is found as

\[ \hat{z}_{r(q+1)} = \arg\max_{\tilde{z}} \frac{\left( u^*_r - w_{rq}^H \hat{X}^*_r \right) \phi(\tilde{z})^2}{\phi(\tilde{z})^* \phi(\tilde{z})} \quad (6.41) \]

\[ w_{r(q+1)} = \left( \hat{X}^*_r \hat{X}^*_{r(q+1)} \right)^{-1} \hat{X}^*_{r(q+1)} u_r. \quad (6.42) \]
mean $\mu_t \approx \hat{X}_\mu w_\mu$. Here $\hat{z}_\mu$ denotes the preimages with mappings $\hat{X}_\mu$ and weights $w_\mu$. One approach to solve this optimization is the gradient descent method, but other algorithms exist (Burges 1996).

Once all preimages are computed, one could set

\[
\begin{align*}
Z'_t & \triangleq [\hat{z}_1 \cdots \hat{z}_R \hat{\mu}] \\
X'_t & \triangleq [\hat{X}_1 \cdots \hat{X}_R \hat{\mu}] \\
\tilde{U}'_t & \triangleq X'_t \begin{bmatrix} w_1 & \cdots & 0_{Q \times 1} \\
\vdots & \ddots & \vdots \\
0_{Q \times 1} & \cdots & w_R \\
0_{Q \times 1} & \cdots & 0_{Q \times 1} \end{bmatrix} \\
M'_t & \triangleq \begin{bmatrix} 0_{(RQ \times 1)} \\
w_\mu \end{bmatrix}.
\end{align*}
\]

(6.43) (6.44) (6.45) (6.46)

More sophisticated, however, is the method shown in figure 6.2(c) and detailed by Chin and Suter (2007), which extracts a better approximation, by using all elements in $Z'_t$ for each component in $\tilde{U}'_t$ and $M'_t$. Notice, this approach effectively makes the weight vectors less sparse, and thus the preimages less redundant. I refer to the work by Chin and Suter (2007) for details.

Once the data is approximated, orthogonality is enforced in the new found subspace $\tilde{U}'_t$. Note, $\tilde{U}'_t$ is only an approximation of the original subspace $U_t$, and thus the unit norm is no longer guaranteed. I now extend the approach by Chin and Suter (2007) to work with complex kernels in Krein space. In particular, using the decomposition of $(\tilde{U}'_t^* \tilde{U}'_t)^2 = \Pi Y^2 \Pi^H$, the estimated projection is whitened to find orthogonal components, i.e.

\[
U'_t \triangleq \tilde{U}'_t \Pi |Y|^{-\frac{1}{2}} \Pi^H.
\]

(6.47)

The new eigenvalues’ magnitudes are given by

\[
|\Lambda'_t| = \text{diag}(U'^*_t U'_t |\Lambda_t|),
\]

(6.48)

where diag(.) discards the values which are not located on a matrix’s diagonal.

After substituting the new representation with the original set, the update is continued through the usual process, as outlined above. Note however, the variable for the number of previous data points $N_t$ remains unchanged. In total, the reduced set contains at most $Q(R + 1)$ preimages, i.e. $Q$ for each component and an additional set of $Q$ data values to represent the mean $\mu_t$. Therefore, it is computed whenever the number of samples in $X_t$ exceeds $Q(R + 1)$. 

Chapter 6. Incremental Slow Feature Analysis 167
6.2.2 Direct Online SFA with Special Kernels

In chapter 5, a special kernel family which allows for a direct representation of the kernel function is introduced. Inspired by DIKPCA, I wish to formulate a direct version of online KSFA, which I call direct incremental KSFA (DIKSFA). In contrast to the general framework in the previous section, DIKSFA does not require a reduced set expansion, making the update computationally faster and more accurate.

Motivation

A regular Krein space’s kernel representation is defined by a “conjugate”-product between the implicit mappings \( \phi(z_i) \) and \( \phi(z_j) \) of samples \( z_i, z_j \in \mathbb{C}^P \) in the original feature space. In traditional systems, the implicity of the mapping, however, makes the reduced set expansion necessary.

The special kernel family in chapter 5 does not require such a set. Recall, kernels of this family are expressed by two explicit functions \( a : \mathbb{C}^P \rightarrow \mathbb{C}^{P'} \) and \( b : \mathbb{C}^P \rightarrow \mathbb{C}^{P'} \) such that the kernel can be computed as

\[
k(z_i, z_j) = a(z_i)^H b(z_j) = b(z_i)^H a(z_j).
\] (6.49)

A member of this kernel family is introduced in chapter 5, where a system is proposed, which achieves very competitive performance when applied to tracking and face recognition. The two explicit mappings here are given by

\[
a(z) = \begin{bmatrix} \frac{r \odot e^{i\theta_1}}{2\sqrt{r^T r P}} \\ e^{i\theta_2} \end{bmatrix} \quad \text{and} \quad b(z) = \begin{bmatrix} e^{i\theta_1} \\ \frac{r \odot e^{i\theta}}{2\sqrt{r^T r P}} \end{bmatrix},
\] (6.50)

where \( z \) corresponds to an image with vectorized gradient magnitudes \( r \in \mathbb{R}^P \) and gradient angles \( \theta \in [-\pi, \pi]^P \). Here, the operator \( \odot \) is shorthand for a componentwise multiplication, and \( e^{i\theta} \triangleq \begin{bmatrix} e^{i\theta(1)} & \cdots & e^{i\theta(P)} \end{bmatrix}^T \). In the current chapter, I desire to employ this kernel to online KSFA due to its previous success on tracking.

Direct Incremental Kernel SFA

A DIKSFA algorithm is developed for any kernel that satisfies eq. (6.49). Recall from chapter 5, the main benefits of the two explicit mappings is the capability of computing an equivalence of the implicit projection explicitly. Given the sample matrices of mappings \( a \) and \( b \), denoted \( X^{(a)} = \begin{bmatrix} a(z_1) & \cdots & a(z_N) \end{bmatrix} \) and \( X^{(b)} \) respectively, the implicit linear combination \( \tilde{X} \Omega \) is replaced by two explicit matrices \( A = X^{(a)} C\Omega \) and \( B = X^{(b)} C\Omega \).
Then, the projection of a new sample \( z \) is exactly computed by utilizing either of its mappings, such that \( \Omega \tilde{X}^\phi(z) = A^Hb(z) = B^H a(z) \). Now I reformulate the online learning process using this setup.

The explicit mappings are exploited to find the eigenvalue decomposition of the updated squared kernel matrix \( \tilde{K}^2_t \) as

\[
\tilde{K}^2_t = \tilde{\Omega}_t A^2 \tilde{\Omega}_t = \begin{bmatrix} |\Lambda_t-1| & A^H_{t-1} B \delta \\ B^H_\delta A_{t-1} & |\delta| \end{bmatrix}^2 ,
\]

where \( A_\delta \) and \( B_\delta \) is related to the new data in \( X^{(a)}_\delta \) and \( X^{(b)}_\delta \) respectively. The update of \( A_t \) and \( B_t \) is analogous to eq. (6.25), and realized through

\[
A_t = \begin{bmatrix} A_{t-1} & A_\delta \end{bmatrix} \tilde{\Omega}_t \quad \text{(6.52)}
\]
\[
B_t = \begin{bmatrix} B_{t-1} & B_\delta \end{bmatrix} \tilde{\Omega}_t \quad \text{(6.53)}
\]

Additionally, the data mean \( \mu \) is substituted by the explicit versions, \( \mu^{(a)} = X^{(a)} M \) and \( \mu^{(b)} \), making the correction in eq. (6.26) and the mean update in eq. (6.27) trivial.

All that is left to do is the formulation of the slow feature extraction using the explicit mappings. The projection in eq. (6.28) is rewritten as follows

\[
W_t^H \tilde{X}_t \tilde{S}_t \tilde{X}_t W_t = |\Lambda_t|^{-1} A^H_\delta \tilde{X}_t \tilde{C}_t \tilde{X}_t^{(a)} H B_t |\Lambda_t|^{-1}
\]
\[
= |\Lambda_t|^{-1} A^H_\delta \tilde{X}_t \tilde{C}_t \tilde{X}_t^{(a)} H B_t |\Lambda_t|^{-1} + |\Lambda_t|^{-1} A^H_\delta \tilde{X}_t^{(b)} \tilde{C}_t \tilde{X}_t^{(a)} H B_t |\Lambda_t|^{-1}
\]
\[
\approx P_t |\Lambda_{t-1}|^{-1} A^H_\delta \tilde{X}_t \tilde{C}_t \tilde{X}_t^{(a)} H B_t |\Lambda_{t-1}|^{-1} P_t
\]
\[
+ |\Lambda_t|^{-1} A^H_\delta \tilde{X}_t^{(b)} \tilde{C}_t \tilde{X}_t^{(a)} H B_t |\Lambda_t|^{-1} .
\]

where \( \tilde{X}^{(a)} \) and \( \tilde{X}^{(b)} \) are the explicit versions of the derived samples in \( \tilde{X} \), expressed via \( \tilde{X}^{(a)} = X^{(a)} D \) and \( \tilde{X}^{(b)} = X^{(b)} D \) respectively, when the backwards derivatives are used. Notice that \( \tilde{X} W \) is expressed through \( A |\Lambda|^{-1} \) and \( B |\Lambda|^{-1} \). Analogously, incorporating explicit mean representations \( \tilde{\mu}^{(a)} = \tilde{X}^{(a)} C \) and \( \tilde{\mu}^{(b)} \), the mean correction factor in eq. (6.33) (or more precisely in eq. (6.39)) is applied. Finally, the projected mean in eq. (6.40) is updated using \( |\Lambda|^{-1} A^H \tilde{\mu}^{(b)} \) or \( |\Lambda|^{-1} B^H \tilde{\mu}^{(a)} \).

After eigenvalue decomposition of the whitened scatter matrix of the derived samples (i.e. \( W_t^H \tilde{X}_t \tilde{S}_t \tilde{X}_t W_t = \Theta_t \Sigma_t \Theta_t^H \)) the slow features expressed by \( V_t \) are represented by the explicit matrices \( V_t^{(a)} = A_t |\Lambda_t|^{-1} \Theta_t \) and \( V_t^{(b)} = B_t |\Lambda_t|^{-1} \Theta_t \), which can be applied on
either mapping of a novel incoming data point \( z \in \mathbb{C}^P \), as

\[
    V_t^*(\phi(z) - \mu_t - \dot{\mu}_t) = V_t^{(a)H}(b(z) - \mu_t^{(b)}) - V_t^*\dot{\mu}_t
    = V_t^{(b)H}(a(z) - \mu_t^{(a)}) - V_t^*\dot{\mu}_t,
\]

(6.55)

where \( V_t^*\dot{\mu}_t = \Theta_t^H W_t^H \tilde{X}_t^* \dot{\mu}_t \) is explicit and taken from eq. (6.40).

### 6.3 Applications of Online SFA

In this section, two applications of the proposed online SFA are introduced. In particular, SFA’s first online change detection algorithm is formulated and applied to temporal video segmentation and tracking with multiple appearance models.

#### 6.3.1 Temporal Video Segmentation

One application of incremental SFA is temporal video segmentation through change detection. As Nater et al. (2011) argue in their work, the segmentation of a video sequence in time is closely related to finding consecutive frames which have large differences in their slow features.

Remember, SFA natively minimizes the slowness of a signal, eq. (6.2). In video, the slowness uses the squared magnitude of the derivative signal, extracted from a sample frame. Analogously, let us define the change of a signal \( x_n \in \mathcal{K} \), at time \( t \), as the magnitude of its projected derivative, \( i.e. \)

\[
    c_t(x_n) = \dot{x}_n^*V_tV_t^*\dot{x}_n = D_t^TX_t^*V_t^*V_t^*X_nD_n,
\]

(6.56)

where \( \dot{x}_n = X_nD_n \) is a product of matrices, which describes the derivative of \( x_n \), using the general notation from section 6.1.2. Again, the derivative may be the product of any two matrices that satisfy \( \dot{x}_n = \hat{X}_nD_n \), where \( \hat{X}_n \) is derived from a set of mapped preimages.

When a new activity has started, the change is expected to be “unusually” large. In this work, the importance of a change is measured as a change ratio between the new data point \( x_{N_t+1} \) and the average change of previous data, given by

\[
    r_t(x_{N_t+1}) = \frac{N_t c_t(x_{N_t+1})}{\sum_{n}^{N_t} c_t(x_n)}.
\]

(6.57)

Notice however, a trivial update of the mean is not possible, as \( c_t \) changes at each time interval. Furthermore, if the whole signal were stored in memory to compute the average at each time step, the requirements of an online system would be violated.
I now present an alternative approach. Considering eq. (6.18) and the update in eq. (6.38), the eigenvalue decomposition of the projected scatter $V_t^*X_tD_tC_tX_t^T\hat{X}_t;V_t = \Theta_t\Sigma_t\Theta_t^H$ is already known. Notice, the eigenvalues are much related to the sum of previous changes, and eq. (6.57) can be computed as follows

$$r_t(x_{N_t+1}) = \frac{N_tC_t(x_{N_t+1})}{\text{tr}(\Sigma_t) + \hat{\mu}_t^*V_t^*\hat{\mu}_t}, \quad (6.58)$$

where $\hat{\mu}_t^*V_t^*\hat{\mu}_t$ handles the mean subtraction by $\hat{\mu}_t$.

With the change ratio $r_t$ in eq. (6.58), it is now possible to identify significant changes in data streams. At each time-step, the significance of variation is first analyzed, and then the SFA is updated with the new data point. A threshold is applied to find the frames with large amount of change. These frames provide the split position in the temporal segments of the video stream. An optional median filter may be applied to smoothen the change detection. However, if immediate output is required this process may be skipped.

### 6.3.2 Multi Appearance Model Tracking

Building on change detection, a tracking framework that detects likely areas of drift is now proposed. Although many tracking applications benefit from online learning (Babenko et al. 2011; Ross et al. 2008), susceptibility to drift is a challenging problem (Babenko et al. 2011). One of the reasons drift can occur is the prolonged exposure to corrupted data, e.g. caused by occlusions, appearance changes or pose variations. Typically these instances eventually harm the tracking system over time, as previously learned appearances are forgotten about in the online model of appearances. In chapter 5, the tracking framework called DIKT is presented, which builds on incremental PCA learning. In the current chapter, I incorporate change detection to this framework.

Figure 6.3 summarizes the new setup of the appearance model. At initialization, a single online model is initiated and used for the tracking mechanism. Once a change is detected, the current version of the knowledge base is copied to create an active online model, and a dormant offline model. The online model now performs the tracking and receives updates, while the offline model is unchanged. In further instances of change, an offline and online model is created for each version of the appearance description, and tracking is done via a combination of all online methods. Finally, the most similar models are merged, as a budget is to be fulfilled to satisfy the online requirements.

In the following, I detail the change detection, the merging of PCA models, and the tracking procedures with holistic PCA appearance models.
Figure 6.3: Tracking with multiple appearance models, using change detection. The models are copied into an offline and online model each time a change is detected. The online models are used to generate the tracking results, while the offline models remain dormant until further changes are detected. Most similar models are merged if necessary to maintain a budget.

Change Detection for Tracking

The change ratio $r_t$ from eq. (6.58) is utilized to detect likely areas of sudden variations, which are assumed to indicate appearance changes.

In particular, one instance of incremental SFA is initiated at the beginning of the video. Its input data consists of the best tracked warped image particles as provided by the tracking system (the same features as for tracking are employed). The update of SFA is computed after every fifth frame, in conjunction with the update of the PCA model in DIKT. The delay not only enables more stable learning, it also allows for a median filter to be used, which further improves the change detection. Finally, a delay is enforced before further change is registered.

Merging Appearance Models

At each change detection, all previously generated models are copied to create one offline and one online version each. Temporary changes therefore only alter half of the appearance models, allowing the other half to stay uncorrupted. In such framework, however, the number of models doubles at each detected change. Thus appearance models need to be merged for such setup to satisfy a memory budget. The procedure for this is detailed in the following.

Let there be two datasets $\{\phi(z^{(i)}_n)\}_{n=1}^{N_i}$ and $\{\phi(z^{(j)}_n)\}_{n=1}^{N_j}$ that make up two different PCAs. Furthermore, let us assume their eigenvalue decompositions of the kernel matrices
are given by

$$
\tilde{K}_i^2 = \left( \frac{1}{N_i} \tilde{X}_i^* \tilde{X}_i \right)^2 = \Omega_i^2 \Lambda_i^2 \Omega_i^H 
$$

(6.59)

$$
\tilde{K}_j^2 = \left( \frac{1}{N_j} \tilde{X}_j^* \tilde{X}_j \right)^2 = \Omega_j^2 \Lambda_j^2 \Omega_j^H 
$$

(6.60)

where $\tilde{X}_i$ and $\tilde{X}_j$ represents the centralized sample matrix of the datasets. The combination of both PCAs is in fact quite similar to the update of the whitening projection for the PCA part of SFA in eq. (6.24). Thus, I find the new, combined eigenvalue spectrum using the decomposition of

$$
\tilde{K}^2 = \tilde{\Omega} \left[ \begin{array}{cc}
\frac{1}{2} | \Lambda_i | & \Omega_i^H \tilde{X}_i \tilde{X}_j \Omega_j \\
\Omega_j^H \tilde{X}_j \tilde{X}_i \Omega_i & \frac{1}{2} | \Lambda_j |
\end{array} \right] \tilde{\Omega}.
$$

(6.61)

Notice, the eigenvalues are weighted, to give both PCAs similar importance, especially in conjunction with further future updates. The number of samples in the combined PCA is set to $N = \frac{N_i + N_j}{2}$ and the mean is updated accordingly.

With the algorithm above, the problem of merging appearance models is simplified to finding the most similar appearance models in the set of subspaces. In chapter 3, I introduce the notion of the angular error between subspaces, based on the work by Gunawan et al. (2005). This measure is now also applied to estimate the difference between PCAs, i.e. the error is computed as

$$
e(U^{(i)}, U^{(j)}) = \max(R_i, R_j) - \sum_{r_i=1}^{R_i} \sum_{r_j=1}^{R_j} \|u^{(i)}_{r_i} u^{(j)}_{r_j}\| \tag{6.62}
$$

where $U^{(i)} = \left[ u^{(i)}_1 \ldots u^{(i)}_{R_i} \right]$ and $U^{(j)} = \left[ u^{(j)}_1 \ldots u^{(j)}_{R_j} \right]$ are the tested PCAs’ subspaces.

**Tracking with Multiple Models**

Once multiple active tracking models are produced, their statistic is to be combined. Let us derive the tracking setup for this probabilistically.

First, consider the framework under linear features that do not require a kernel representation. As shown in previous chapters, the tracking problem is expressed as a HMM. In particular, let $\{A^{(t)}_m\}_{m=1}^M$ be the set of $M$ affine transformations extracted by a particle filter at time $t$, and $\{x^{(t)}_m\}_{m=1}^M$ be the corresponding observation vectors. Then, the probability of the transformation $A^{(t)}_m$ given the particles $\{x^{(t)}_m\}_{m=1}^M$ is to be estimated, which
is approximated by

$$p(A_m^{(t)}|\{x_m^{(t)}\}_{m=1}^M) = p(A_m^{(t)}|x_m^{(t)}) \propto p(x_m^{(t)}|A_m^{(t)})p(A_m^{(t)}).$$

(6.63)

Hence, the system is composed out of an observation model $p(x_m^{(t)}|A_m^{(t)})$ and a transformation model $p(A_m^{(t)}|\{A_{m-1}^{(t-1)}\}_{m=1}^M)$.

The transformation model is provided by a GMM:

$$p(A^{(t)}|\{A_{m-1}^{(t-1)}\}_{m=1}^M) \approx \sum_{m=1}^M p(A_{m-1}^{(t-1)}|z_m^{(t-1)})N(A_m^{(t-1)}, \Xi)|_{A^{(t)}},$$

(6.64)

where the likelihood of the previous time-step $p(A_{m-1}^{(t-1)}|z_m^{(t-1)})$, acts as weight, and $\Xi$ is an independent covariance matrix, which represents the variance in horizontal and vertical displacement, rotation, scale, ratio and shear as in previous chapters and proposed by Ross et al. (2008).

Let us consider the observation model. Probabilistic PCA, as presented by Tipping and Bishop (1999), allows us to formulate the likelihood of a sample, following Ross et al. (2008), as

$$p(x_m) = \left(2\pi\right)^{d/2} \left| U(\Lambda^{1/2} - \sigma^2 I)U^T + \sigma^2 I \right|^{-1/2}$$

$$\times e^{-\frac{1}{2}(x_m - \mu)^T (I - UU^T)(x_m - \mu)}$$

$$\times e^{-\frac{1}{2}(x_m - \mu)^T \Lambda^{1/2} U^T (x_m - \mu)}$$

(6.65)

(6.66)

where $x_m$ corresponds to the observation for $A_m^{(t)}$, $U$ is the PCA’s subspace, $\Lambda^{1/2}$ contains the PCA’s eigenvalues, $\mu$ represents the mean of the training data, $P'$ is the dimensionality of $x$, $||.||$ computes the determinant of a matrix, and $\sigma^2$ controls the spread. Hence, the probability of a sample being produced by a single PCA is explained by the reconstruction error (eq. (6.65)) and the inner subspace distance (eq. (6.66)).

Now consider a model with $G$ PCAs. The average of multiple Gaussians is assumed to provide a sample’s probability. Therefore, the explicit probability of a sample $p(x_m)$ is expressed by

$$\frac{1}{G(2\pi)^{d/2}} \sum_g \sigma_g^{R_g - P'} \prod_{r=1}^{R_g} (\lambda_r(g))^{-\frac{1}{2}} e^{-\frac{1}{2}(x_m - \mu_g)^T \left(\frac{1}{\sigma_g^2}(I - U_gU_g^T) + U_g\Lambda_g^{1/2} U_g^T\right)(x_m - \mu_g)},$$

(6.67)

where $U_g$, $\Lambda_g$, $\mu_g$ and $R_g$ correspond to the $g^{th}$ PCA, $\lambda_r(g)$ is the $r^{th}$ eigenvalue of the $g^{th}$ PCA, and the determinant is expressed as the product of the matrix’s eigenvalues.
Again, this formulation combines the external reconstruction error with the internal inner subspace error.

Now assume kernelized data. In this case the proposed model is commonly not a well defined probability distribution. Nonetheless, as illustrated in figure 6.4 and shown by Moghaddam and Pentland (1997), the cost function may still be based on the internal and external distance of the PCAs. Hence, I assume the internal distance to be given, as before, by

\[(x_m - \mu_g)^* U_g | \Lambda_g |^{-\frac{1}{2}} U_g^* (x_m - \mu_g) \]  

(6.68)

and the external distance to be equivalent to the kernel distance in eq. (6.1), i.e.

\[(x_m - U_g U_g^* x)^* (x_m - U_g U_g^* x). \]  

(6.69)

In previous chapters, and in the following experiments, I employ the reconstruction error only. This is equivalent to assuming a variation of infinity in the subspace, i.e. \(\sigma \rightarrow \infty\). Hence, I assume the likelihood of an observation to be proportional to

\[ \sum_g \left( \prod_{r=1}^{R_g} (\lambda_r^{(g)})^{-\frac{1}{2}} e^{-\gamma_g (x_m - U_g U_g^* x)^* (x_m - U_g U_g^* x)} \right), \]  

(6.70)

where \(\gamma_g\) controls the spread of the individual PCAs as in previous chapters.

### 6.4 Evaluation

I evaluate the proposed incremental framework for SFA in a proof of concept, temporal video segmentation and tracking.
6.4.1 Proof of Concept

The general properties of incremental SFA in the setting of change detection with synthetic data is evaluated. In particular, the batch KSFA, which incorporates all data points at once, is compared to incremental KSFA which learns at each time-step. As Wiskott and Sejnowski (2002), an input signal \( \{ z_n = \begin{bmatrix} \sin(y_n) + \cos(11y_n)^2 & \cos(11y_n) \end{bmatrix}^T \}_{n=1}^N \) is assumed, where \( \{ y_n \}_{n=1}^N \) is taken from \( N = 2000 \) equally distributed values in the range \([0, 4\pi]\). The corresponding slow features are to be found. With a quadratic kernel, the true solution is given by \( o_n = \begin{bmatrix} \sin(y_n) & \cos(11y_n) \end{bmatrix}^T \) for sample \( z_n \).

Three versions of incremental KSFA are tested (figure 6.5). IKSFA uses the full data set as support – it grows larger over time. R-IKSFA employs a reduced set expansion for learning with constant memory. D-IKSFA exploits the direct equivalent mappings of the quadratic kernel (i.e. \( a(z_n) = b(z_n) = \begin{bmatrix} z_n(1) & z_n(2) & z_n(1)z_n(2) & z_n(1)^2 & z_n(2)^2 \end{bmatrix}^T \)). Finally, batch KSFA is considered as ground truth, as in this setup, the complete sequence is known a priori.

All methods converge towards the same slow features. However, they differ in their

\[^3\text{As in standard PCA, the sign of the components is irrelevant.}\]
execution. IKSFA is most stable for change ratio estimation. With a reduced set, noisy results are encountered at the beginning of the sequence, when learned by R-IKSFA. D-IKSFA performs similarly to IKSFA. In terms of running time, IKSFA performs worst as the complete set of samples is required for the kernel trick at each time-step.\textsuperscript{4} R-IKSFA’s reduced support set improves execution by a factor of 9, while D-IKSFA squeezes the learning time to 3.9 seconds – more than 11 times faster than IKSFA, and 3 times faster than the batch version of KSFA. Notice, as preimages are easily computed for the quadratic kernel, R-IKSFA is also fast.

6.4.2 Change Detection with Real Data

The quality of incremental SFA and change detection is evaluated on real data. The data set consists of the expressions in the MMI Facial Expression Database (MMI) produced by Valstar and Pantic (2010). Initially all videos are concatenated, and DIKT from chapter 5 is applied to extract aligned images (40 $\times$ 40 pixel) of several activities. The resulting sequence consists of 4182 frames, cropped to the subject’s face.

I analyze the proposed framework with different kernels. The direct input features (Linear), the quadratic kernel (Quadratic), the standard Gaussian kernel (Gaussian) and the Krein space kernel in chapter 5 (Krein) are compared. The deviation of the Gaussian kernel is fixed to $\frac{1}{N(N-1)} \sum_{n=1}^{N} \sum_{n'=1}^{N} ||x_n - x_{n'}||^2$, where $N$ is the number of samples (Kwok and Tsang 2004), and the parameters of Krein follow those in the previous chapter.

Incremental Learning Behavior

To quantitatively measure the learning behavior of online SFA, the incremental version is compared to its offline equivalent. In particular, the angular difference between the offline and online subspaces is used for evaluation (Gunawan et al. 2005)

$$e(M) = M - \sum_{m=1}^{M} \sum_{m'=1}^{M} \|v_m^{(i)} v_{m'}^{(j)}\|$$

(6.71)

where $M$ is the number of slow features evaluated, $v_m^{(i)}$ and $v_{m'}^{(j)}$ are the individual slow feature projections of the compared SFAs.

Figure 6.6 shows the angular error with different numbers of components in the PCA and varying number of slow features. Here, the kernel versions with complete support set

\textsuperscript{4}Tests were conducted in MATLAB on a desktop computer with an Intel core i7 870 processor at 2.93 GHz and 8 GB RAM.
Figure 6.6: Angular error between batch and online learned subspaces. Different number of components in the PCA ($R$) are shown with varying number of slow features. No reduced set, or forgetting factor is applied.

Figure 6.7: Angular error with and without reduced set. Incremental learning is compared to batch learning. The reduced set representation contains 260 preimages (prefix R) and is compared to direct mappings (prefix D). Results are shown for different sequence sizes and a PCA of 25 components.

are employed, and the forgetting factor is set to 1 (no forgetting) to facilitate comparison. The Linear, Quadratic and Krein kernel perform equally well. The Gaussian kernel has a slightly increased error rate. In general, the proposed setups perform with low errors and they are independent of the number of components in the PCA stage of SFA.

In the second test, I evaluate the incremental SFA with reduced sets (prefix R) or direct mapping (prefix D). Here, the number of components in the PCA is fixed to $R = 25$ and a budget of 260 preimages is requested in the reduced set expansion, i.e. $Q = 10$. Figure 6.7 shows the angular error when learning is performed on different sequence lengths. The algorithms work well for short videos (about 500 frames). Nonetheless, approximation errors in the reduced set representation seem unavoidable and introduces lower performance in longer sequences. Here the algorithms with direct mapping are more suitable, as low errors are achieved for any sequence length.
Figure 6.8: False positive over false negative rate of different kernel setups. Batch Krein learns from the whole sequence a priori.

Change Detection

Now let us apply the change detection algorithm of section 6.3.1 to the dataset. MMI consists of facial expressions, labeled by onset, apex and offset. The onset and offset indicate the start and end of an expression respectively. These labels are utilized as ground truth, as they mark the frames in which the activity in the video changes. The equal error rate of false negatives over false positives then reveals the performance of individual setups.

Linear, Quadratic, Gaussian and Krein are compared. First, I optimize with respect to the number of PCA components $R$, the forgetting factor $w$ and the number of slow features $M$. A median filter of eight frames is applied to the output and direct mappings are used where possible. The Gaussian kernel function receives a fixed budget of $Q = 10$.

All methods perform best with a forgetting factor of $w = 0.996$ ($\approx 250$ frames), and slow feature number $M = 3$. However, the ideal number of PCA components varies for each method ($R = 15$ for Linear, $R = 30$ for Quadratic, $R = 50$ for Gaussian, $R = 10$ for Krein). Additionally, I include the batch version of Krein (Batch Krein), for which the change ratio is computed with all samples known a priori – i.e. I learn the slow features from the complete sequence.

The false negative over false positive rates are shown in figure 6.8. The quadratic expansion has a slightly better equal error rate than linear features, while the Gaussian kernel is the most performant positive definite setup. The best results are achieved by the direct version of incremental SFA with Krein kernel (D-Krein). It is important to
emphasize: The advantage of the proposed Krein kernel stems not only from its domain-specific design; its direct mappings are fast to compute (in linear time complexity) and fewer PCA components are required to outperform other systems. As comparison, the direct mapping of the quadratic expansion is polynomial, and 30 PCA components are employed. Figure 6.9 shows an excerpt from the results for Krein.

Notice, the batch version of Krein (Batch Krein) performs with reduced results to its online equivalent, despite its knowledge of the complete data stream. In the online version, the forgetting factor allows the system to adapt more quickly to new parts of the video. For example, consider figure 6.10 which shows the top three projections after frame 900 for the Linear kernel (i) as batch setup, (ii) with $w = 1$ (no forgetting), and (iii) with $w = 0.996$. Here Linear is chosen to aid visualization, as the projections remain in the original space. Let us compare the batch setup to no forgetting: The resulting projections are virtually equivalent, which validates the proposed update procedure. Notice how the slowest projection relates to the smile around frame 530. With forgetting, the effect of $w$ becomes apparent, as the projections are most relevant to later expressions – here, the smile is no longer visible.

I conclude this section with a selection of video sequences from different scenarios. Figure 6.11 shows the resulting temporal segmentation of two yoga sequences, extracted from YouTube (http://www.youtube.com/watch?v=ziVctQnyvwE) and two examples from the ballet dataset by Fathi and Mori (2008). All in all, it can be seen that the frames with largest change ratio provide plausible segmentations.

Comparison to Optical Flow and PCA

In the final part of the experiments for change detection, I analyze my framework in comparison to alternative methods. A typical approach to finding drift in many tracking
Batch
Linear

Linear (w = 1)

Linear (w = 0.996)

PCA Eigenvalues

Figure 6.10: The three slowest features after 900 frames using the linear kernel. Batch learning is compared to online learning with and without forgetting. The eigenvalues after PCA learning are visualized in the bottom row.

frameworks is the distance between the current frame and the learned subspace. Similarly, this can be applied to change detection via the reconstruction error given by

\[ e_r(R) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{z} - \sum_{r=1}^{R} \mathbf{u}_r \mathbf{u}_r^\text{H} \mathbf{z} \right\|_F^2, \] (6.72)

where \( \mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_r \end{bmatrix} \) is the reduced eigenspace of previous samples, and \( \mathbf{z} \) denotes the tested input. The eigenvectors can be incrementally learned by the methods presented in previous chapters.

Additionally, let us consider another feature input, known as optical flow. Optical flow has proven advantageous for describing the dynamics of a video sequence (Sun et al. 2010). While the computation of such features is expensive (sub-realtime with the source code by Sun et al. (2010)), I will compare it to SFA with Krein space kernel in this section.

I use the following setups in my experiment: PCA learning with intensity values (PCA-Linear), optical flow (PCA-Flow) and my kernel (PCA-Krein), which are compared to SFA with intensity values (SFA-Linear), optical flow (SFA-Flow) and my kernel (SFA-Kernel). Figure 6.12 illustrates the results. Notice, change detection solely based on PCA without the added temporal information (i.e. PCA-Linear and PCA-Krein) is
Figure 6.11: Segmentation for yoga and ballet scenes. SFA’s change detection finds the visually distinct parts of the videos. The same parameters as in the quantitative evaluation are used.
Figure 6.12: False positive over false negative rate comparison for optical flow (Flow), intensity values (Linear) and our kernel (Krein) as PCA and SFA setup.

inferior to other methods. Less efficient PCA-Flow on the other hand is competitive in terms of equal error rate. SFA inherently incorporates temporal information as the derivative scatter matrix is considered in its optimization. Indicative of this is the good performance of SFA-Linear and SFA-Krein. SFA-Flow is less performant, as optical flow is less valuable to SFA.

Figure 6.10 visualizes the learned subspaces of SFA and PCA. Notice, SFA finds the important features, which change slowest over time. In contrast, PCA computes the features that best describe the samples, without the temporal information. For instances, the eyebrows and contours around the nose of the subject are more prominently modeled with PCA.

6.4.3 Tracking with Change Detection

In the final experiment, the tracking framework with change detection (CD-DIKT), as proposed in section 6.3.2, is evaluated. Here, I focus on the analysis of the gain that change detection using SFA brings to adaptive tracking. Hence, DIKT from chapter 5 is used as a base line comparison.

The performance is tested on the nine videos also used in previous chapters. Recall, these videos contain drastic changes of the targets appearance, including pose variation, occlusions, and nonuniform illumination. All videos have three to seven fiducial points.
<table>
<thead>
<tr>
<th></th>
<th>Vid1</th>
<th>Vid2</th>
<th>Vid3</th>
<th>Vid4</th>
<th>Vid5</th>
<th>Vid6</th>
<th>Vid7</th>
<th>Vid8</th>
<th>Vid9</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIKT</td>
<td>4.44</td>
<td>2.77</td>
<td>2.58</td>
<td>(lost)</td>
<td>(lost)</td>
<td>3.79</td>
<td>2.19</td>
<td>2.75</td>
<td>(lost)</td>
</tr>
<tr>
<td>CD-DIKT</td>
<td>4.15</td>
<td>2.60</td>
<td>2.34</td>
<td>5.95</td>
<td>(lost)</td>
<td>3.68</td>
<td>2.18</td>
<td>2.68</td>
<td>6.70</td>
</tr>
</tbody>
</table>

Table 6.1: Mean RMS errors for DIKT and CD-DIKT with change detection. I write “lost” to indicate the sequences in which the tracker does not follow the target throughout, after visual inspection.

Figure 6.13: Tracking results and models of appearances. I encode offline and active models with a sequence of Os and As to highlight their active learning periods (e.g. A-O-A learned with data before and after the occlusion, but not during). The model that provides the highest score is highlighted by a dark bar.

which allow for quantitative performance evaluation. The RMS errors between the true and the estimated locations of these points are used for evaluating the performance. For all videos, I fix the translation model of the particle filter tracking framework to the values found best in chapter 5. The parameters of the change detection in CD-DIKT are set for each video specifically, as the thresholds of the change ratio vary video-dependently. All other parts of DIKT and CD-DIKT are identical.

Table 6.1 lists the mean RMS errors for each video. Consistently, CD-DIKT improves upon DIKT. Furthermore, CD-DIKT is capable of tracking Vid4 and Vid9 successfully as it utilizes previously seen appearances, within the proposed drift reducing setup.

Figure 6.13 shows an example of Vid4. Not visible in the figure, the first change splits the initial model into an active model A, and a dormant offline model O. In frame 415 a change is detected, which reactivates the dormant model O to form the active model O-A and the offline model O-O. The already active model A is also split into offline and online versions, A-O and A-A respectively. The tracker consults both active models, i.e. O-A and A-A, during the occlusion. Notice however, the more naïve O-A is favored during the occlusion as it knows less about the face, and thus favors most recent data. Around frame 435 the face is fully visible again, and model A-A computes the largest likelihood for a short period of time. Finally, after the second change detection in frame 505, all
models are split once more, to form $O-O-O$, $O-O-A$, $O-A-O$, $O-A-A$, $A-O-O$, $A-O-A$, $A-A-O$ and $A-A-A$. Initially, model $A-O-A$ provides most confidence. This model is the one that learns most before the occlusion, yet is idle during the input of corrupted data. Intuitively, it knows most about the face, as it does not drift during the occlusion. Finally $O-O-A$ is favored thereafter, as this model is significantly trained from most recent data after the corruptions in frame 415 to 505.

6.5 Conclusions

In this chapter, I propose an exact kernel SFA (KSFA) framework for arbitrary Krein space kernels. Then, general online KSFA is formulated, which employs a reduced set expansion to fulfill budget requirements. Finally, by utilizing a special kind of kernel family, an exact online KSFA is proposed, for which no reduced set is required.

I apply online SFA and develop SFA’s first change detection algorithm for stream data. This framework is employed to temporal video segmentation and tracking. When applied to real data streams, the proposed method successfully segments the input using change detection. Combined with an online learning tracking system, change detection improves upon systems without such detection, in the presented evaluation.

In the scope of my thesis, I have shown how to formulate the general eigenvalue problem incrementally. Again, the proposed method can be expressed efficiently by use of two explicit representations, which encode a robust kernel framework in Hilbert or Krein space. Additionally, a forgetting factor is introduced to the online learning framework to improve the change detection algorithm.
Conclusions and Outlook

In this thesis, the theories of robust online subspace learning is advanced in a number of approaches. First, robust dissimilarities which allow for domain-specific kernel methods are developed. In contrast to standard systems, where of-the-shelf kernels are used, these proposed algorithms successfully suppress outliers in a number of applications. Then, as some proposed kernels lie in indefinite Krein space, the theories of KPCA are extended to a reproducing kernel Krein space. Here, any arbitrary kernel in Krein space is supported. Moreover, efficiency for online learning is enabled, which allows for exact incremental KPCA. This incremental setup exploits the special properties of the proposed kernel functions. Finally, I extend my work to the generalized eigenvalue problem, and introduce KSFA with indefinite kernels, incremental KSFA for arbitrary kernel functions, and fast online KSFA for the special kernel family. The proposed methods are tested as adaptive visual tracking systems and many other computer vision applications throughout this thesis, where very competitive tracking results are achieved.

The following section, section 7.1, summarizes the individual findings in more detail. Section 7.2 lists the conclusions of my work. In section 7.3 an outlook into future directions is presented. Final remarks are given in section 7.4.

7.1 Summary

Chapter 3 formulates Euler PCA (e-PCA) as a fast, direct and robust alternative to standard ℓ2-norm PCA. Its advantage is that an explicit mapping can be extracted to form a kernel function that is outlier suppressant. Due to the explicit feature space of the non-linear mapping, incremental e-PCA is easily obtained. In experiments, e-PCA achieves promising results for the applications of face reconstruction, object tracking and background modeling.
Inspired by e-PCA, chapter 4 introduces a kernel function that is robust and employs a domain-specific representation. In particular, in contrast to e-PCA, the specific problem of matching vectors of 3D rotations is targeted. Notice, rotations occur in a number of computer vision tasks. First, a novel distance is introduced and expressed using a kernel function with an explicit mapping. The mapping leads to a new representation of 3D rotations, the full-angle quaternion (FAQ) representation, which allows for efficient online subspace learning. The proposed method is applied to 3D object recognition from point clouds, and object tracking from color video.

Another domain-specific kernel function is proposed in chapter 5 for efficient and adaptive visual tracking. Here, a robust gradient-based comparison function is employed as a kernel. This kernel, however, requires the extension of KPCA to allow for Krein space. I formulate KPCA with arbitrary indefinite kernels, and propose an exact incremental learning framework for the kernel’s special properties. By using the proposed online subspace system as appearance model, and including a particle filter, the direct incremental KPCA tracker (DIKT) is created. The evaluation on nine popular and challenging videos reveal superior tracking performance for the introduced system.

Finally in chapter 6, I target the generalized eigenvalue problem, and present incremental KSFA for temporal video segmentation and tracking with change detection. First, KSFA with indefinite kernels is introduced. Then incremental KSFA for arbitrary kernels using preimages is formulated, and an exact incremental KSFA is realized via the special kernel family. Online SFA is applied to develop SFA’s first change detection algorithm for stream data. This framework is employed to temporal video segmentation and tracking where promising results are achieved.

7.2 Conclusions

The main conclusions are listed in the following. These concern the combination of robustness and efficiency, advantages of domain-specific kernels, and the valuable properties of adaptive online subspace learning.

7.2.1 Robustness with Efficiency

Commonly in the literature, systems are designed either for robustness or efficiency. Ross et al. (2008) develop a fast tracking framework which is ideal only for independent and identically distributed Gaussian noise. Approaches like the one by Candés et al. (2009) are designed for robust offline learning. Chin and Suter (2007) kernelize the framework
of Ross et al. (2008), but at a cost of efficiency and accuracy. In contrast, in my work, I carefully design kernel functions which enable the combination of the two.

The theories of kernel methods are a useful tool to non-linearize standardly linear systems. Consequently, more flexible feature spaces can be expressed, which introduce robustness when comparing data points. This thesis shows, that through the development of kernels with explicit kernel equivalent mappings, fast systems can be produced which suppress outliers successfully.

7.2.2 Domain-Specific Properties

Often, methods employ kernel learning, but with standard off-the-shelf kernels, like GRBFs. Typically, these kernels are not robust for independent outliers (de la Torre and Black 2003). The proposed 3D representation in chapter 4, and the tracking framework in chapter 5 are different. Here, kernel functions are presented, which incorporate the inherent robustness from the data distribution. In particular, for example for DIKT in chapter 5, gradient-based features are employed, which are known to be less susceptible to illumination changes and varying light. Moreover, however, the dissimilarity also incorporates a cosine-based measure, which has been shown to be robust to outliers in image data (Tzimiropoulos et al. 2012b).

The choice of kernels is an important one for the design of kernel learning systems. This thesis shows that, by exploiting inherently robust properties of the data, performance and accuracy can be improved. In the experimental results, the proposed learning methods are shown to outperform common state-of-the-art approaches in their class.

7.2.3 Adapting and Learning Incrementally

Especially in tracking systems, adaptive appearance models through incremental learning have proven advantageous in several setups (Babenko et al. 2011; Mei and Ling 2009; Ross et al. 2008). In this thesis, these results are confirmed through the employment of robust online subspace learning for tracking. The presented algorithms incorporate the statistics of the test data points during execution, without the requirement of prior knowledge about this data set – only an initialization is needed.

Generally, incremental learning systems are ideal for setups in which unpredicted situations occur or several object classes are to be supported. In particular, they support events which were not captured in training. Such systems ideally learn what needs to be known, and thus adapt to the presented circumstances. Unfortunately, however, drifts in the online hypothesis are sometimes possible. Here, change detection, as proposed in chapter 6, can somewhat overcome this issue.
7.3 Future Work

In this section, potential further developments of the research presented in this thesis are listed. Most prominently, these target the knowledge base of the system, and further applications of online subspace learning with robust kernel functions.

7.3.1 Machine Lifelong Learning

Online learning enables the extraction of knowledge not contained in the training dataset. In fact, in my research, no offline training is required, as the appearance model of the tracked target is completely learned during the system’s execution. However, introducing \textit{a priori} learned knowledge could enrich the learning framework, as the system, in a sense, then knows what to expect. One route for improvement could be directed towards the combination of online and offline technology. The incremental approach proposed in this thesis could be enhanced with deeply learned offline or prior knowledge.

Another path is machine lifelong learning, which has received an increasing interest in the research community (Freytag et al. 2012; Orabona and Cesa-Bianchi 2011; Silver and Poirier 2007). Typically in this approach, the learning system is equipped with knowledge, that is to be adapted to a novel domain. Similarly, the algorithms of this thesis could be employed for an initial learning phase, which then can be embedded into the deeper knowledge base of the system, analogously to the short-term and long-term memory in the human brain.

7.3.2 Analysis of Further Applications

Incremental learning and the proposed robust kernel methods are applied to a number of application throughout this thesis. Most applications are computer vision problems, and based on PCA. In chapter 6 an incremental generalized eigenvalue problem is developed, which shows promising results for online learned with SFA. However, there are many other problems, which can be solved through eigenvalue decomposition. It would be enriching to the literature to analyze the proposed techniques in other subspace-based settings.

Also, several kernels which exploit robust domain-specific properties are presented. In the experiments, the additional properties of robustness improve the overall results. Additionally, through the proposed theory of learning with indefinite kernels, the space of possible kernel functions is increased. Now, many kernel systems should be revisited to allow for Krein space, and the possibility of exploiting the inherent robustness of the data’s domain is to be explored.
Another interesting avenue of research is to employ my kernel methods to discriminative algorithms such as support vector machines and its kernel version in particular.

7.4 Final Remarks

The value of sophisticated interfaces between human and machine is undoubtedly large in the development of coming technology. Every user is different, environments vary; my vision of the future is the possibility of systems which, given time, will adapt and improve their understanding continuously throughout their life. Systems which adapt their knowledge to suit the current user and situation, will improve the communication between user and machine. The combination of efficient online subspace learning and robust feature representations could help realize this dream of adaptive user interaction in unconstrained environments.

While the goal of such system is still far from being reached, I hope to have improved the understanding of robust online subspace learning through the theories proposed, and I encourage further research into the future work outlined above.
Bibliography


Soriano, M., B. Martinkauppi, S. Houvinen, and M. Laaksonen. Skin Detection in Video


Zhang, F. Quaternions and Matrices of Quaternions. Linear Algebra Appl., 251:2157, Jan


# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>2 dimension</td>
</tr>
<tr>
<td>3D</td>
<td>3 dimension</td>
</tr>
<tr>
<td>e-PCA</td>
<td>Euler PCA</td>
</tr>
<tr>
<td>eT</td>
<td>Euler tracker</td>
</tr>
<tr>
<td>AAM</td>
<td>active appearance model</td>
</tr>
<tr>
<td>CCA</td>
<td>canonical correlation analysis</td>
</tr>
<tr>
<td>CCIPCA</td>
<td>candid covariance-free incremental PCA</td>
</tr>
<tr>
<td>DIKPCA</td>
<td>direct incremental KPCA</td>
</tr>
<tr>
<td>DIKSFA</td>
<td>direct incremental KSFA</td>
</tr>
<tr>
<td>DIKT</td>
<td>direct incremental KPCA tracker</td>
</tr>
<tr>
<td>FAQ</td>
<td>full-angle quaternion</td>
</tr>
<tr>
<td>FRC</td>
<td>fast robust correlation</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian mixture model</td>
</tr>
<tr>
<td>GRBF</td>
<td>Gaussian radial basis function</td>
</tr>
<tr>
<td>HAQ</td>
<td>half-angle quaternion</td>
</tr>
<tr>
<td>HMM</td>
<td>hidden Markov model</td>
</tr>
<tr>
<td>KPCA</td>
<td>kernel PCA</td>
</tr>
<tr>
<td>KSFA</td>
<td>kernel SFA</td>
</tr>
<tr>
<td>LDA</td>
<td>linear discriminant analysis</td>
</tr>
<tr>
<td>MCA</td>
<td>minor component analysis</td>
</tr>
<tr>
<td>MLR</td>
<td>multiple linear regression</td>
</tr>
<tr>
<td>MMC</td>
<td>maximum correntropy criterion</td>
</tr>
<tr>
<td>MMI</td>
<td>MMI Facial Expression Database</td>
</tr>
<tr>
<td>MMS</td>
<td>metric multidimensional scaling</td>
</tr>
<tr>
<td>PCA</td>
<td>principal component analysis</td>
</tr>
<tr>
<td>PLS</td>
<td>partial least squares</td>
</tr>
<tr>
<td>RGB</td>
<td>red-green-blue</td>
</tr>
<tr>
<td>RKHS</td>
<td>reproducing kernel Hilbert space</td>
</tr>
<tr>
<td>RM</td>
<td>rotation matrix</td>
</tr>
<tr>
<td>RMS</td>
<td>root mean square</td>
</tr>
<tr>
<td>RV</td>
<td>rotation vectors</td>
</tr>
<tr>
<td>SFA</td>
<td>slow feature analysis</td>
</tr>
<tr>
<td>SVD</td>
<td>singular value decomposition</td>
</tr>
<tr>
<td>Toshiba</td>
<td>Toshiba Research Europe Limited</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>(\angle)</td>
<td>operator which returns the angle of a complex number</td>
</tr>
<tr>
<td>(\star)</td>
<td>applies a filter to an image</td>
</tr>
<tr>
<td>(</td>
<td>\mathcal{K}</td>
</tr>
<tr>
<td>(c)</td>
<td>computes the change of consecutive data in SFA</td>
</tr>
<tr>
<td>(r)</td>
<td>computes the change ratio of a data point in SFA</td>
</tr>
<tr>
<td>(\mathbb{C})</td>
<td>space of complex numbers</td>
</tr>
<tr>
<td>((\cdot)^*)</td>
<td>Hermitian transpose combined with fundamental matrix</td>
</tr>
<tr>
<td>\text{diag}</td>
<td>discards the elements of a matrix, which are not on the diagonal</td>
</tr>
<tr>
<td>\text{dim}</td>
<td>returns the dimensionality</td>
</tr>
<tr>
<td>(\oplus)</td>
<td>direct sum operator</td>
</tr>
<tr>
<td>(\mathcal{D}\mathcal{T}(3))</td>
<td>space of 3D dilatations</td>
</tr>
<tr>
<td>(\odot)</td>
<td>element-wise product between two vectors</td>
</tr>
<tr>
<td>([\cdot])</td>
<td>finds the integer value of a number</td>
</tr>
<tr>
<td>(|\cdot|_F)</td>
<td>finds the Frobenius norm</td>
</tr>
<tr>
<td>(\mathbf{F}^-)</td>
<td>negative fundamental projection</td>
</tr>
<tr>
<td>(\mathbf{F}^+)</td>
<td>positive fundamental projection</td>
</tr>
<tr>
<td>(\mathcal{J})</td>
<td>fundamental symmetry of Krein spaces</td>
</tr>
<tr>
<td>((\cdot)^\mathcal{H})</td>
<td>Hermitian transpose of a matrix</td>
</tr>
<tr>
<td>(\mathcal{H})</td>
<td>reproducing kernel Hilbert space</td>
</tr>
<tr>
<td>(\mathbb{H})</td>
<td>space of quaternion numbers</td>
</tr>
<tr>
<td>(\mathbf{I})</td>
<td>identity matrix</td>
</tr>
<tr>
<td>(i)</td>
<td>imaginary part of a matrix</td>
</tr>
<tr>
<td>(\text{Im})</td>
<td>extracts the imaginary part of a complex number as real number</td>
</tr>
<tr>
<td>(\langle\cdot,\cdot\rangle)</td>
<td>computes the inner product of two vectors</td>
</tr>
<tr>
<td>(\mathcal{K})</td>
<td>reproducing kernel Krein space</td>
</tr>
<tr>
<td>(\mathcal{K}_-)</td>
<td>negative definite part of a Krein space</td>
</tr>
<tr>
<td>(\mathcal{K}_+)</td>
<td>positive definite part of a Krein space</td>
</tr>
<tr>
<td>(\mathbb{R})</td>
<td>finds the element-wise magnitudes of a matrix</td>
</tr>
<tr>
<td>(\mathbb{N})</td>
<td>space of natural numbers</td>
</tr>
<tr>
<td>(\mathcal{N})</td>
<td>normal Gaussian distribution</td>
</tr>
<tr>
<td>(\text{orth})</td>
<td>computes the orthogonal components</td>
</tr>
<tr>
<td>(\mathbb{R})</td>
<td>space of real numbers</td>
</tr>
<tr>
<td>(\mathbb{R}_+)</td>
<td>real numbers, greater than 0</td>
</tr>
<tr>
<td>(\text{Re})</td>
<td>extracts the real part of a complex number</td>
</tr>
<tr>
<td>(\Delta)</td>
<td>computes slowness of a slow feature function</td>
</tr>
<tr>
<td>(\mathcal{SO}(2))</td>
<td>space of 2D rotations</td>
</tr>
<tr>
<td>(\mathcal{SO}(3))</td>
<td>space of 3D rotations</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>spread of a distribution</td>
</tr>
<tr>
<td>(\mathcal{O})</td>
<td>computational time complexity</td>
</tr>
<tr>
<td>(\text{tr})</td>
<td>Computes the trace of a matrix</td>
</tr>
<tr>
<td>((\cdot)^\top)</td>
<td>Transpose of a matrix</td>
</tr>
<tr>
<td>(\mathbb{Z})</td>
<td>space of integer numbers</td>
</tr>
</tbody>
</table>
Properties of the Indefinite Kernel

Let kernel $k : \mathbb{C}^P \times \mathbb{C}^P \to \mathbb{C}$ be the kernel from eq. (5.14) in chapter 5 and $z_i, z_j \in \mathbb{C}^P$ are two samples in original space.

**Lemma 1.** $\sum_{p=1}^{P} r_i(p) \leq \sqrt{\sum_{c=1}^{d} r_i^2(p)} P$, where $r_i = R(z_i)$.

**Proof.** For any $P$-dimensional vector $r_i \in \mathbb{R}_+^P$ the following is true.

$$
\sum_{p=1}^{P} r_i(p) \leq \sqrt{\sum_{p=1}^{P} r_i^2(p)} P
$$

$$
\sum_{p=1}^{P} r_i(p) \sum_{p=1}^{P} r_i(p) \leq \sum_{p=1}^{P} r_i^2(p) P
$$

$$
\sum_{p=1}^{P} r_i^2(p)) + 2 \sum_{p=1}^{P} \sum_{p' = p + 1}^{P} r_i(p') r_i(p') \leq \sum_{p=1}^{P} r_i^2(p) + \sum_{p'=1}^{P} r_i^2(p)(P - 1)
$$

$$
0 \leq \sum_{p=1}^{P} r_i^2(p)(P - 1) - 2 \sum_{p=1}^{P-1} \sum_{p'=p+1}^{P} r_i(p) r_i(p')
$$

$$
0 \leq \sum_{p=1}^{P-1} \sum_{p' = p + 1}^{P} (r_i(p) - r_i(p'))^2
$$

(A.1)

**Theorem 3.** The kernel is normalized, i.e. $|\text{Re}(k(z_i, z_j))| \leq 1$.

**Proof.** With lemma 1 the following holds.

$$
|\text{Re}(k(z_i, z_j))| = \frac{\sum_{p=1}^{P} r_i(p) \cos(\theta_i(p) - \theta_j(p))}{2\sqrt{\sum_{p=1}^{P} r_i^2(p)} P} + \frac{\sum_{p=1}^{P} r_j(p) \cos(\theta_i(p) - \theta_j(p))}{2\sqrt{\sum_{p=1}^{P} r_j^2(p)} P}
$$

$$
\leq \frac{\sum_{p=1}^{P} r_i(p) \cos(\theta_i(p) - \theta_j(p))}{2\sqrt{\sum_{p=1}^{P} r_i^2(p)} P} + \frac{\sum_{p=1}^{P} r_j(p) \cos(\theta_i(p) - \theta_j(p))}{2\sqrt{\sum_{p=1}^{P} r_j^2(p)} P}
$$

$$
\leq \frac{\sum_{p=1}^{P} r_i(p)}{2\sqrt{\sum_{p=1}^{P} r_i^2(p)} P} + \frac{\sum_{p=1}^{P} r_j(p)}{2\sqrt{\sum_{p=1}^{P} r_j^2(p)} P}
$$

$$
\leq \frac{\sqrt{P}}{2\sqrt{P}} + \frac{\sqrt{P}}{2\sqrt{P}} = 1
$$

(A.2)
Correction Term with Forgetting

Ross et al. (2008) provide the formulation of the correction term for the mean subtraction (in eq. (6.38)) only for $w = 1$. I derive a general term for any $w \in [0, 1]$. The notation follows chapter 6. The scatter matrix $S_t$ at time $t$ with applied $w$ is given by

$$S_t = \sum_{n=1}^{N_t-1} (x_n - \mu_t) w^2 (x_n - \mu_t)^H J + \sum_{n=1}^{N_\delta} (x_n - \mu_t) (x_n - \mu_{t-1})^H J$$

$$= w^2 \sum_{n=1}^{N_t-1} \left( (x_n - \mu_{t-1}) - \frac{N_\delta (\mu_\delta - \mu_{t-1})}{wN_{t-1} + N_\delta} \right) \left( (x_n - \mu_{t-1}) - \frac{N_\delta (\mu_\delta - \mu_{t-1})}{wN_{t-1} + N_\delta} \right)^H J$$

$$+ \sum_{n=1}^{N_\delta} \left( (x_n - \mu_\delta) - \frac{wN_{t-1} (\mu_{t-1} - \mu_\delta)}{wN_{t-1} + N_\delta} \right) \left( (x_n - \mu_\delta) - \frac{wN_{t-1} (\mu_{t-1} - \mu_\delta)}{wN_{t-1} + N_\delta} \right)^H J$$

$$= w^2 S_{t-1} + S_\delta + \frac{w^2 N_{t-1} N_\delta (N_{t-1} + N_\delta)}{(wN_{t-1} + N_\delta)^2} (\mu_{t-1} - \mu_\delta)^* (\mu_{t-1} - \mu_\delta)^* \cdot (B.1)$$

Thus, the correction term for the mean subtraction is given by the additional data point

$$\sqrt{\frac{w^2 N_{t-1} N_\delta (N_{t-1} + N_\delta)}{wN_{t-1} + N_\delta}} (\mu_{t-1} - \mu_\delta) \cdot (B.2)$$