Technische Universität München
Lehrstuhl für Datenverarbeitung
(Fachgebiet Geometrische Optimierung und Maschinelles Lernen)

# Robust Structured and Unstructured Low-Rank Approximation on the Grassmannian

**Clemens Otto Benjamin Hage**

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

**Doktor-Ingenieurs (Dr.-Ing.)**

genehmigten Dissertation.

**Vorsitzende(r):** Prof. Dr. Jörg Conradt

**Prüfer der Dissertation:**

1. Prof. Dr.-Ing. Klaus Diepold
2. Prof. Pierre-Antoine Absil, Ph. D. (nur schriftliche Beurteilung)
3. Priv.-Doz. Dr. Martin Kleinsteuber (mündliche Prüfung)

Die Dissertation wurde am 20.12.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 01.06.2017 angenommen.

Clemens Otto Benjamin Hage. *Robust Structured and Unstructured Low-Rank Approximation on the Grassmannian.* Dissertation, Technische Universität München, Munich, Germany, 2017.

# Acknowledgments

Writing a doctoral thesis requires a good amount of perseverance. But much more so it requires a healthy working environment that provides scientific, personal and financial support. I am very thankful to my advisers Dr. Martin Kleinsteuber and Prof. Dr. Klaus Diepold for providing me with such a constructive working environment at our institute and especially within our all-too-soon deceased research group. Working at the university can sometimes be a sobering experience regarding the discrepancy between how much time some people invest in teaching and how little it is often valued. Considering how much our society relies on the quality of education, valuing grants and citations over teaching and personality can only be a step in the wrong direction. The tougher the times, the more important it is to have good companions at your side. Thanks to Marko for bringing me to the institute and for helping me out so many times. Thanks to Tim for joining me on my side path in Audio Processing. Thanks to Simon, Uli and Julian for all the fun times in the past and for your inspiration and advice. Thanks to Hao, Spüli, Röhrich, my roomie Dominik, Hauser, Vince, Alex and Peter for all your support in recent years and for always having an open ear. Thanks to Pierre-Antoine Absil for reviewing my thesis. Thanks to my family and to my friends for your great encouragement. And more than anything, to Marijke - how pointless all this would have been without you at my side.

# Abstract

Low-rank approximation is a linear dimensionality reduction technique with countless applications in data analysis and signal processing. A standard algorithm for finding the best low-rank approximation according to the Euclidean distance measure is the well-known Principal Component Analysis, which offers a closed-form solution to the problem, based on the assumption that additive disturbances are Gaussian-distributed. However, this makes the approach vulnerable against large outliers in the data. To leverage the issue, Robust Low-Rank Approximation methods have been developed that are tailored to the low-rank and sparse data model instead, which assumes that disturbances may have arbitrary magnitude but occur only at few positions. The low-rank constraint is commonly addressed either through nuclear norm minimization or by factorizing the low-rank matrix. In this work a factorization model with orthogonality constraints is considered through which the orthogonal factor represents a basis of the subspace to be found while the other one represents the coordinates of the low-rank approximation in this subspace. This resolves scaling ambiguities and allows to identify every subspace estimate one-to-one with an element of the Grassmannian, the manifold of low-dimensional subspaces. The sparsity of the residual error is commonly enforced with the $\ell_1$ norm because it is the closest convex relaxation to the ideal $\ell_0$ measure. As it is known that non-convex sparsity measures are a closer $\ell_0$-approximation and thus offer enhanced sparsifying behavior, this thesis proposes to replace the $\ell_1$ norm by a non-convex sparsifying function based on the extension of the $\ell_p$ norm to $0 < p < 1$. In contrast to the $\ell_1$ norm, the proposed smoothed $\ell_p$-norm loss function is differentiable and allows for gradient-based optimization. Its practical use is investigated in three related Robust Low-Rank Approximation problems.

Firstly, an algorithm for Robust PCA from incomplete observations is proposed, which achieves increased robustness against outliers in the data. Due to the employed factorization model the proposed loss function is minimized in alternating manner on the Grassmannian and in Euclidean space, respectively, using efficient Conjugate Gradient methods with subsampled backtracking line-search to solve the individual optimization problems. Compared to other state-of-the-art algorithms on simulated data, the proposed method is able to recover the underlying subspace even in cases where competing methods fail as the data does not fulfill the ideal low-rank-and-sparse-assumption. In a real-world video segmentation experiment the method is used to separate foreground elements from a dynamic background in an unsupervised manner. The proposed method reduces ghosting artifacts and achieves fast processing times compared to competing approaches.

Secondly, a Robust Subspace Tracking algorithm is developed that also uses the proposed smoothed $\ell_p$-norm loss function to enforce sparsity on the residual error of the approximation. The method exploits the manifold setting to update the subspace model incrementally with every new data sample, while avoiding the introduction of additional slack variables or Lagrangian multipliers. This allows to deal with dynamic subspaces that vary over time. Based on this idea, the *pROST* method has been developed, which is a specialized algorithm for online video segmentation in realtime using GPU acceleration. The evaluation of the method on a public benchmark reveals superior behavior for certain types of dynamic backgrounds.

As a third contribution, a Robust Low-Rank Approximation method with additional structural constraints is presented, which is more robust to outliers and scales better to large dimensions than existing factorization-based approaches. For the special application of Time Series Analysis, an efficient online method is proposed that is based on structured low-rank approximation with Hankel structures. By reusing previous estimates, the method achieves improved efficiency compared to randomly initialized structured low-rank approximation.

# Contents

**List of Figures** **ix**

**List of Tables** **xiii**

**List of Symbols** **xv**

**1. Introduction** **1**

1.1. Low-Rank Approximation problems . . . . . 2

1.2. Formulation of the research problem . . . . . 7

1.3. Contributions . . . . . 9

1.4. Thesis Outline . . . . . 11

**2. Related Work** **13**

2.1. Matrix Completion . . . . . 13

2.2. Robust PCA . . . . . 17

2.3. Subspace Tracking . . . . . 20

2.4. Structured Low-Rank Approximation . . . . . 24

**3. Optimization on the Grassmannian** **27**

3.1. Stiefel and Grassmann manifolds . . . . . 28

3.2. Quotient geometry of the Grassmannian . . . . . 31

3.3. Gradient and Conjugate Gradient Descent on the Grassmannian . . . . . 33

3.4. Conjugate Gradient Descent with subsampled line search . . . . . 38

**4. Grassmannian Robust PCA** **43**

4.1. Smoothed $\ell_p$-norm loss function . . . . . 44

4.2. Algorithmic description . . . . . 47

4.2.1. Alternating minimization scheme . . . . . 48

4.2.2. Input normalization . . . . . 51
4.2.3. Memory efficient processing for Big Data analysis . . . . . 52
4.3. Evaluation on simulated data . . . . . 52
4.3.1. Phase transitions in rank and sparsity . . . . . 56
4.3.2. Runtime and optimum choice of the shrinkage parameter $\mu$ . . . . . 60
4.3.3. Evaluation of cost function parameter $p$ . . . . . 60
4.3.4. Robust Matrix Completion . . . . . 63
4.3.5. Computational complexity and memory requirements . . . . . 66
4.4. Application on Video Segmentation . . . . . 68
4.4.1. Principles of video segmentation using Robust PCA . . . . . 69
4.4.2. Real-world example . . . . . 73

**5. Robust Subspace Tracking on the Grassmannian 79**
5.1. Algorithmic Description . . . . . 80
5.2. Evaluation on Simulated Data . . . . . 83
5.3. The pROST algorithm . . . . . 86
5.3.1. Motivation . . . . . 86
5.3.2. Description of the algorithm . . . . . 87
5.3.3. Evaluation on the *changedetection.net* benchmark . . . . . 90
5.3.4. Results and discussion . . . . . 93

**6. Robust Structured Low-Rank Approximation on the Grassmannian 101**
6.1. Linear matrix structures . . . . . 101
6.2. Alternating minimization framework . . . . . 104
6.3. An efficient algorithm for Online Time Series Analysis . . . . . 108
6.4. Experiments on System Identification and Time Series Forecasting . . . . . 111

**7. Conclusion 119**

**Bibliography 123**

**Appendix 131**
A.1. Subspaces and Norms . . . . . 131
A.1.1. Vector spaces and subspaces . . . . . 131

A.1.2. Vector and matrix norms . . . . . . . . . . . . . . . . . . . . . . . . . . 131
A.2. Gradient derivation for the Robust SLRA problem . . . . . . . . . . . . . . . . 133

# List of Figures

1.1. Three examples of linear matrix structures . . . . . . . . . . . . . . . . . . . . . 6

3.1. Illustration of the Conjugate Gradient Descent method on the Grassmannian 36

4.1. Visualization of the normalized $\ell_p$ norm for scalar input for different values of the parameter $p$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.2. Comparison between $\ell_2$ norm, $\ell_1$ norm and normalized smoothed $\ell_p$-norm loss function with $p = 0.1$ and varying parameter $\mu$ . . . . . . . . . . . . . . . 46
4.3. Phase transitions in rank and sparsity for *GRPCA* in comparison to state-of-the-art Robust PCA algorithms. White: Relative Reconstruction Error below $10^{-8}$, Black: Relative Reconstruction Error above $10^{-1}$ . . . . . . . . . 57
4.4. Progress in minimizing the Relative Reconstruction Error over the number of iterations for varying minimum values $\mu_{\text{end}}$ of the smoothing parameter $\mu$ . 61
4.5. Phase transitions in rank and sparsity for the proposed *GRPCA* method for varying cost function parameters $p$. White: Relative Reconstruction Error below $10^{-4}$, Black: Relative Reconstruction Error above $10^{-1}$ . . . . . . . . . 62
4.6. Phase transitions in rank and sparsity for the proposed *GRPCA* method for varying cost function parameters $p$ on a data set with outliers of large magnitude. White: Relative Reconstruction Error below $10^{-3}$, Black: Relative Reconstruction Error above $10^{-1}$ . . . . . . . . . . . . . . . . . . . . . . . . . 64
4.7. Phase transitions in rank and sparsity for the proposed *GRPCA* method for varying fractions of observed entries. White: Relative Reconstruction Error below $10^{-4}$, Black: Relative Reconstruction Error above $10^{-1}$ . . . . . . . . . 65
4.8. Phase transitions in rank and sparsity for *RMC* for varying fraction of observed entries. White: Relative Reconstruction Error below $10^{-4}$, Black: Relative Reconstruction Error above $10^{-1}$ . . . . . . . . . . . . . . . . . . . . . 66

4.9. Run time comparison for *GRPCA* with full cost function evaluation and subsampled conjugate gradient with $|\Psi| = 10^4$, respectively, for squared input of varying dimension $m$ . . . 67

4.10. Run time comparison for *GRPCA* with full cost function evaluation, subsampled conjugate gradient with $|\Psi| = 10^4$ and additional selective matrix-matrix product, respectively, for squared input of varying dimension $m$, sampled at a rate of $\frac{|\Omega|}{m^2} = 0.1$ . . . 68

4.11. Vectorized video sequence containing $n$ frames with $m$ pixels each ($m = 9, n = 3$) . . . 70

4.12. Rank-one approximation of a simplified video sequence with common PCA . . 71

4.13. Frame-wise reconstruction of the video segmentation results obtain from common PCA. left/above: background, right/below: foreground . . . 71

4.14. Rank-one approximation of a simplified video sequence with Robust PCA . . 72

4.15. Frames # 1806, 1814 and 1820 of the *escalator* sequence. . . . 73

4.16. Background estimates for frames # 1806, 1814 and 1820 of the *escalator* sequence, obtained using *GRPCA* with varying cost function parameter $p$. . . 75

4.17. Comparison of background estimates for frames # 1806, 1814 and 1820 of the *escalator* sequence. From top to bottom: *GRPCA* with $p = 0.1$, *IALM*, *LMaFit*, *GoDec* . . . 76

4.18. Comparison of foreground estimates for frame # 1813 of the *escalator* sequence. From top left to bottom right: *GRPCA* with $p = 0.1$, *IALM*, *LMaFit*, *GoDec* . . . 77

5.1. Progression of relative error during the online identification of a static subspace of dimension $k = 5$ within surrounding dimension $m = 500$ from 10% of samples with 10% sparse outliers, evaluated for different values of cost parameter $p$ . . . 84

5.2. Progression of relative error while tracking subspaces of dimension $k = 5$ within surrounding dimension $m = 50$ with 60% dense outliers, with rapid subspace change at $j = 15000$, evaluated for different values of cost parameter $p$ 85

5.3. Input, ground truth and segmentation result of *pROST* for frame # 1150 of the *badminton* sequence from the *changedetection.net* benchmark . . . 96

6.1. Results for the identification of a SISO LTI system. Gray: Noise-free impulse response, Black: Sample points of the impulse response with additive Gaussian noise and sparse outliers at a sampling rate of 0.5 for $t \in [0, 50]$. Red: Low-Rank approximation with *SLRAbyF* ($\ell_2$ data fit). Green: Robust Low-Rank approximation with the proposed *GRSLRA* method. . . . . . . . . 112
6.2. Online identification of a Single-Input-Single-Output Linear Time Invariant system . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
6.3. Online identification of a Single-Input-Single-Output Linear Time Varying system . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
6.4. Six-month forecast of monthly airline passenger data from the years 1952-1960 based on a 3-year observation period. Actual data and predictions using *GRSLRA*, *SLRAbyF* and a seasonal *ARIMA* model, respectively . . . . 117
6.5. Six-month forecast of monthly airline passenger data from the years 1999-2014 based on a 3-year observation period. Actual data and predictions using *GRSLRA*, *SLRAbyF* and a seasonal *ARIMA* model, respectively . . . . 118

# List of Tables

4.1. Run time comparison for state-of-the-art Robust PCA algorithms. Noise-free test data generated according the model of Shen et al. [75] with dimensions $m = n = 200$, rank $k = 20$ and outlier density $\rho = 0.1$ . . . . . . . . . . . . . 59

4.2. Run time comparison of state-of-the-art Robust PCA algorithms for foreground-background segmentation of the *escalator* video sequence (dimensions $20,800 \times 3417$) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

5.1. Results per category for *pROST* on the *changedetection.net* benchmark. . . . 94

5.2. Results per category for *GRASTA* on the *changedetection.net* benchmark. . . 95

# List of Symbols

| | |
|---|---|
| $\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{S}$ | Matrices, written as capital boldface letters. |
| $\boldsymbol{x}, \boldsymbol{l}, \boldsymbol{s}$ | Column vectors, written as lowercase boldface letters. |
| $a, \beta, N$ | Scalars, written as lowercase arabic/greek or capital letters. |
| $\boldsymbol{x}_j$ | $j$-th column of matrix $\boldsymbol{X}$. |
| $\boldsymbol{x}_{i,:}$ | $i$-th row of matrix $\boldsymbol{X}$. |
| $X_{ij}$ | $i$-th element in the $j$-th column of matrix $\boldsymbol{X}$. |
| $x_i$ | $i$-th entry of vector $\boldsymbol{x}$. |
| $\boldsymbol{X}^{(i)}, \boldsymbol{x}^{(i)}, \alpha^{(i)}$ | Elements at the $i$-th iteration. |
| $\boldsymbol{X}_{(j)}, \boldsymbol{x}_{(j)}, \alpha_{(j)}$ | Elements at time instance $j$. |
| $\boldsymbol{X}^{\dagger}$ | Moore-Penrose pseudoinverse of $\boldsymbol{X}$. |
| $\boldsymbol{X}^{\top}, \boldsymbol{x}^{\top}$ | Matrix / Vector transposed. |
| $\boldsymbol{I}_k$ | Identity matrix of dimension $(k \times k)$. |
| $\mathbf{0}_m$ | All zero vector of dimension $m$. |
| $\boldsymbol{e}_i$ | Canonical basis vector with the $i$-th entry equal to one and zeros elsewhere. |
| $\boldsymbol{E}_{ij}$ | Matrix with the $(i, j)$-th entry equal to one and zeros elsewhere. |
| $\\|\cdot\\|_2$ | $\ell_2$-norm. |
| $\\|\cdot\\|_F$ | Frobenius norm. |
| $\\|\cdot\\|_1$ | $\ell_1$-norm. |
| $\\| \cdot \\|_p$ | $\ell_p$-norm. |
| $\\|\cdot\\|_0$ | $\ell_0$-pseudo-norm. |
| $\\|\cdot\\|_*$ | Nuclear norm. |
| $\operatorname{rank}(\cdot)$ | Matrix rank. |
| $\operatorname{tr}(\cdot)$ | Matrix trace. |
| $\mathit{\Omega}, \mathcal{P}_{\mathit{\Omega}}(\boldsymbol{X})$ | Sampling set, subsampled matrix. |
| $\operatorname{vec}(\boldsymbol{X})$ | Column-wise vectorization of matrix $\boldsymbol{X}$. |
| $\operatorname{vec}^{-1}(\boldsymbol{x})$ | Inverse vectorization operation. |

| | |
|---|---|
| $m$ | Signal dimension. |
| $n$ | Number of measurements. |
| $k$ | Upper bound of the rank. |
| $\mathrm{Gr}_{k,m}$ | Grassmann manifold. |
| $\mathrm{St}_{k,m}$ | Stiefel manifold. |
| $\boldsymbol{U} \in \mathrm{St}_{k,m}$ | Stiefel matrix with $k$ orthogonal columns of length $m$. |
| $[\boldsymbol{U}] \in \mathrm{Gr}_{k,m}$ | Element of the Grassmannian (equivalence class representation). |

# Chapter 1.

# Introduction

Nowadays, we live in a world of data. For a long time, data acquisition was limited by the cost and quality of sensor hardware such as digital cameras, whereas today even cheap hardware like the sensors of smartphones allow to capture high-quality data. While computational power and storage was expensive and required immobile acquisition platforms such as a desktop computer, a virtually unlimited amount of data can be recorded and stored today by mobile devices like smartphones or wearables, be it for private, public or commercial use. With the advent of social networks and the rise of electronic commerce, a new kind of data besides digital media such as pictures or videos is of growing importance, which is user data. Practically every use of an online device leaves behind a trail of user data, containing information about people's habits, their preferences and their interests. As skeptical as one should be about the constant recording of electronic actions and interactions and their possible use or abuse, it would be naive and also impossible to try to reverse this trend of evermore increasing data acquisition. While it is important to discuss about what to do and what not to do with recorded data, a more technical question from an engineering point of view is how to manage the massive amounts of data. Not only is data recorded more and more often, letting the amount of *data samples* increase, but also the *dimensionality* of the data, i.e. the size of the samples or the number of *features*, grows larger and larger. This makes it difficult to process data, a phenomenon often labeled as the *Big Data* problem. Unlike ever before, this development asks for effective and efficient methods of extracting relevant information out of seemingly unfathomable sets of data. But there is hope, because even if the dimension of data often appears to be very high from the outside, the underlying factors are commonly much simpler and less dimensional. *Dimensionality reduction* techniques aim at identifying these *latent variables* and at reducing the complexity of data while maintaining the relevant information.

## 1.1. Low-Rank Approximation problems

Considering that every data sample of an input matrix $\boldsymbol{X}^{m\times n}$ is an element of an $m$-dimensional vector space, a linear dimensionality reduction method searches for a low-dimensional subspace of dimension $k < m$ (often $k \ll m$) in which the data can be represented. As the full representation $\boldsymbol{L}$ of the approximation in the surrounding space is a low-rank matrix with $\operatorname{rank}(\boldsymbol{L}) \leq k$, the problem of determining this subspace and the coordinates of the approximated data points is a *Low-Rank Approximation* problem. Depending on the application and on the assumptions on the data, different norms may be chosen to measure the residual error $\boldsymbol{X} - \boldsymbol{L}$ and thus the distance between the original data and the approximation. The following overview lists the low-rank approximation problems discussed in this thesis. An explanation of subspaces and norms can be found in Appendix A.1.

### Low-Rank Approximation with Euclidean distances

An intuitive and often-made assumption is that the underlying low-rank representation of the data points is contaminated by i.i.d. Gaussian noise, i.e. the energy of the disturbance is evenly spread across all dimensions. The optimum distance measure for this assumption is the Euclidean or $\ell_2$ distance, motivating the minimization problem

$$\min_{\operatorname{rank}(\boldsymbol{L})\leq k} \|\boldsymbol{X} - \boldsymbol{L}\|_F^2 \ . \tag{1.1}$$

This problem is commonly identified with **Principal Component Analysis (PCA)** [69], which searches for an orthogonal basis that spans the subspace of the closest approximation in Euclidean sense and which also computes the coordinates of the data in this subspace. A closed-form solution exists, based on the Singular Value Decomposition (SVD)

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{1.2}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $m \times \min(m, n)$ and, respectively, $n \times \min(m, n)$-dimensional matrices with orthogonal columns and $\Sigma$ is a diagonal matrix containing the singular values in descending order of magnitude. It is well-known that the best approximation of rank $k$ in Euclidean ($\ell_2$) sense is provided by the truncated SVD of $\boldsymbol{X}$

$$\boldsymbol{L} = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^\top \tag{1.3}$$

with $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ denoting the truncation of the respective matrices to the first $k$ columns and $\boldsymbol{\Sigma}_k$ being a matrix that contains only the first $k$ singular values on its diagonal [36]. This convenient solution makes PCA a simple and easily applicable tool for a wide range of applications and explains its omnipresent appearance.

### Matrix Completion

So far it has been assumed that the input data is fully observed, i.e. all entries of $\boldsymbol{X}$ are known. However, in some applications this is not the case, as e.g. partial entries of a database may be missing. The problem of reconstructing the missing entries of a matrix while assuming that the matrix has low rank is known as *Matrix Completion* and can be formally stated as

$$\min_{\operatorname{rank}(\boldsymbol{L}) \leq k} \left\| \mathcal{P}_{\varOmega} \left( \boldsymbol{X} - \boldsymbol{L} \right) \right\|_F^2 , \tag{1.4}$$

where $\mathcal{P}_{\varOmega}$ is a linear measurement operator

$$\mathcal{P}_{\varOmega}\left(\boldsymbol{X}\right) = \begin{cases} X_{ij} & \text{if} \quad (i,j) \in \varOmega \\ 0 & \text{otherwise} \end{cases} \tag{1.5}$$

and $\varOmega$ is the index set containing tuples $(i,j)$ corresponding to the row and column position of the revealed matrix entries.

The problem has received considerable attention, e.g. for its use in collaborative filtering tasks where a data set is partly observed by several entities and the goal is to reconstruct it from the incomplete measurements. As one of the most prominent examples this appears in recommender systems for music or video streaming services. Given the ratings that users have assigned to certain products, the goal is to anticipate the users' attitude towards items they have not rated yet, as to improve the quality of recommendations for other products and to increase the user activity on the platform. In 2006 the company *Netflix* initiated a competition to improve the results of their existing recommendation system, which is now known as the *Netflix Challenge*. Although the targeted improvement was not met within the competition [5], the initiative inspired many new thoughts on the problem.

### Robust Low-Rank Approximation

The results on how well missing data can be reconstructed from noise-corrupted measurements using the low-rank assumption have inspired further investigations in this field. Specifically, the Low-Rank Approximation problem has been extended to non-Gaussian disturbances that have an arbitrarily large magnitude but occur only at a few coordinates. As these entries are not in line with the measurements at other coordinates they are dubbed *intra-sample outliers*. Such disturbances may cause the affected data points to lie significantly far away from the dominant subspace in Euclidean sense and thus from non-affected samples, making the affected samples *inter-sample outliers*. As the two phenomena commonly go hand in hand, the term *outliers* will be used throughout the thesis to denote the occurrence of defective measurements, while inter-sample outliers are referred to as *outlier samples* whenever the distinction is important. Similarly, the term *inlier* (sample) will denote a measurement that is in line with other measurements. A data model with additive outliers is

$$\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S} \quad \text{with} \quad \operatorname{rank}(\boldsymbol{L}) \leq k \quad \text{and} \quad \|\boldsymbol{S}\|_0 \ll mn, \tag{1.6}$$

where the number of non-zero entries in the outlier matrix $\boldsymbol{S}$ is measured using the $\ell_0$ norm[1]. Matrices with few non-zero entries are known as *sparse* matrices and the data model in (1.6) has become popular as the *low-rank-and-sparse* data model. Accordingly, the *low-rank-and-sparse decomposition* aims at decomposing a given matrix blindly into the sum of a low-rank and a sparse component. This approach is a robustified version of the PCA problem (1.1) in terms of robustness against spurious outliers of large magnitude in the data set. While common PCA assumes that errors in the data set are Gaussian distributed (i.e. the disturbance is homogeneously spread across all dimensions), the errors in (1.6) affect only few entries of the data but concentrate all energy on these entries. It is quite intuitive that

[1]Technically, the $\ell_0$ measure is only a pseudo norm, see Appendix A.1.2. It will nevertheless be referred to as the $\ell_0$ norm throughout this work, while keeping in mind the limitations of the concept.

common PCA will lead to suboptimal results in the presence of large outliers, as the $\ell_2$-loss aims at distributing the residual error across all directions.

The ideal ***Robust PCA*** problem can be stated as

$$\min_{\operatorname{rank}(\boldsymbol{L})\leq k} \|\boldsymbol{X}-\boldsymbol{L}\|_0 . \tag{1.7}$$

Minimizing the $\ell_0$ norm and thereby solving (1.7) is an NP-hard problem for which no closed-form solution exists [30].

If the data set is only incompletely observed on an index set $\Omega$, (1.7) can be restated as

$$\min_{\operatorname{rank}(\boldsymbol{L})\leq k} \|\mathcal{P}_\Omega(\boldsymbol{X}-\boldsymbol{L})\|_0 , \tag{1.8}$$

which implies that the sparsity of the residual error is only enforced over the set of observable coordinates. This problem is referred to as ***Robust Matrix Completion***, although the term Robust PCA often encompasses also the case of incomplete observations.

## Subspace Tracking

In a *static* environment all data samples are available at the time of processing, even if their entries may only be partially observed. As a consequence, the low-rank approximation problems (1.1) and (1.7) are *batch* algorithms that process all data samples at once. However, cases may occur where the size of the data set is prohibitively large, such that batch algorithms cannot process them due to memory limitations. When data is captured one sample at a time, the delay introduced by accumulating a batch of data samples before processing them may become unacceptable. Or the underlying distribution of the samples is not stationary but the statistics change over time, so that older and newer samples cannot be jointly processed. To resolve all these cases, *Subspace Tracking* methods have been developed. As the name indicates, they learn a low-dimensional subspace incrementally over time and can track temporal changes in the subspace. Formally, given a sequence of data samples $\mathcal{X} := \{\boldsymbol{x}_{(t)} \in \mathbb{R}^m \mid t = 1, \dots, T\}$ the minimization problem

$$\min_{\mathcal{L}} \sum_{t=1}^{T} \left\|\boldsymbol{x}_{(t)} - \boldsymbol{l}_{(t)}\right\|_2^2 \tag{1.9}$$

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ x_2 & x_3 & x_4 & x_5 & x_6 \\ x_3 & x_4 & x_5 & x_6 & x_7 \\ x_4 & x_5 & x_6 & x_7 & x_8 \\ x_5 & x_6 & x_7 & x_8 & x_9 \end{bmatrix}$$

**(a)** Hankel matrix

$$\begin{bmatrix} x_5 & x_4 & x_3 & x_2 & x_1 \\ x_6 & x_5 & x_4 & x_3 & x_2 \\ x_7 & x_6 & x_5 & x_4 & x_3 \\ x_8 & x_7 & x_6 & x_5 & x_4 \\ x_9 & x_8 & x_7 & x_6 & x_5 \end{bmatrix}$$

**(b)** Toeplitz matrix

$$\begin{bmatrix} x_5 & x_4 & x_3 & x_2 & x_1 \\ x_1 & x_5 & x_4 & x_3 & x_2 \\ x_2 & x_1 & x_5 & x_4 & x_3 \\ x_3 & x_2 & x_1 & x_5 & x_4 \\ x_4 & x_3 & x_2 & x_1 & x_5 \end{bmatrix}$$

**(c)** Circulant matrix

**Figure 1.1.:** Three examples of linear matrix structures

searches for a sequence of vectors $\mathcal{L} := \{\boldsymbol{l}_{(t)} \in \mathbb{R}^m \mid t = 1, \ldots, T\}$ that minimizes the sample-wise distance to the input sequence. Replacing the metric with the $\ell_0$ norm leads to the ***Robust Subspace Tracking*** problem. The difference to previously discussed problems is that the low-rank approximation samples $\boldsymbol{l}_{(t)}$ do not necessarily all lie in the same subspace as the columns of $\boldsymbol{L}$ do in (1.1) or (1.7), respectively. Instead, the subspace can vary over time, so that two subsequent approximation vectors $\boldsymbol{l}_{(t)}$ and $\boldsymbol{l}_{(t+1)}$ lie in different yet related subspaces. Therefore, with each new data sample the subspace estimate is updated. This makes Subspace Tracking an efficient and flexible tool for online low-rank approximation under varying conditions.

## Structured Low-Rank Approximation (SLRA)

Structured matrices play an important role in various Signal Processing applications such as System Identification and Time Series Analysis. Prominent linear matrix structures are e.g. Hankel, Toeplitz or Circulant matrices, which are visualized in Figure 1.1. While the Hankel structure enforces identical elements on its antidiagonals, (i.e. subsequent columns are shifted upwards by one element), the other two have identical elements on the diagonals (i.e. the columns are shifted downwards), with the Circulant matrix having the additional constraint that elements vanishing from the last row reappear in the first one after shifting.

In some applications it is of interest to find a low-rank approximation of these structured matrices, e.g. to determine the inner dynamics of a Linear Time Invariant (LTI) system by approximating its Hankel-structured impulse response with a low-rank Hankel matrix. Considering a set $\mathcal{S}$ of matrices fulfilling certain linear structural constraints, then the

*Structured Low-Rank Approximation* (SLRA) problem writes as

$$\min_{\operatorname{rank}(\boldsymbol{L}) \leq k} \|\boldsymbol{X} - \boldsymbol{L}\|_F^2 \quad \text{subject to} \quad \boldsymbol{L} \in \mathcal{S} \ . \tag{1.10}$$

Again, the norm of the residual between $\boldsymbol{X}$ and $\boldsymbol{L}$ may be measured with respect to the $\ell_2$ or the $\ell_0$ norm, with the latter describing a ***Robust SLRA*** problem. Generally, there is no efficient closed-form solution to (1.10) regardless of the distance measure. Given an intermediate solution, it is possible to find the closest structured matrix and, if the residual is measured with the $\ell_2$ norm, to solve the low-rank approximation task in closed form via PCA. However, it is also quite obvious that low-rank approximation in general does not maintain a given matrix structure and, on the other hand, the projection onto the space of structured matrices does not preserve the low-rankness of a matrix. Therefore, a feasible low-rank approximation algorithm should minimize the residual while respecting both the low-rank and the structural constraint at the same time.

## 1.2. Formulation of the research problem

The development of efficient methods for Robust Low-Rank Approximation problems has been intensively studied in the past years, driven on the one hand by the general focus on sparse data models and, on the other hand, by the evermore growing demand for understanding large amounts of data through extraction of latent factors in an unsupervised way. There is a general tendency in the community to prefer convex approaches before non-convex approaches, as convexity allows to prove algorithmic properties like convergence or recovery guarantees. For Robust Low-Rank Approximation problems based on the low-rank-and-sparse data model, this has been the major incentive to address the low-rankness with nuclear norm minimization and to employ the $\ell_1$ norm to enforce sparsity on the residual error of the approximation. At the same time, however, it has been observed that non-convex approaches are often feasible and effective in practice and surpass the results of convex methods in real-world applications. This is the main incentive to investigate such approaches and to further tweak their performance. For example, factorizing a low-rank matrix instead of minimizing the nuclear norm guarantees an upper bound on the approximation and is much more memory-efficient, even if the overall approach of optimizing the factors becomes non-convex. While the estimation of the best rank is a separate topic and outside the scope of this work I will cover the following aspects in my thesis:

### Robust PCA

The vast majority of existing approaches employ the $\ell_1$ norm as a sparsifying function, as it is the closest convex approximation of the practically infeasible $\ell_0$ norm. However, previous results have shown that non-convex sparsifying functions allow for an even better sparsifying behavior than the $\ell_1$ norm. In this thesis I investigate whether Robust Low-Rank Approximation with non-convex $\ell_0$-surrogates is feasible in practice, as to push the limits of robust subspace recovery. For the static setting I develop a Robust PCA algorithm that is efficient in terms of both computation and memory. The aim is to exploit the benefits of a non-convex sparsity measure while observing good convergence behavior despite the general risks of non-convex methods such as premature termination in suboptimal solutions.

### Robust Subspace Tracking

Imposing orthogonality constraints on one of the factors in the factorization model comes at the cost of some computational overhead, as the optimization of this variable is no longer in Euclidean space but on the Grassmann manifold. In practice, however, this is not a significant drawback, since efficient solvers are at hand. Moreover, the manifold setting allows to update an existing approximation gradually by moving between elements on the Grassmannian, which makes the algorithm much more flexible in an online application and which furthermore requires less memory compared to a batch algorithm. I investigate if a Robust Subspace Tracking approach using a two-factor data model and a differentiable non-convex $\ell_0$-surrogate loss function is feasible without introducing further slack variables, keeping the algorithm as simple as possible. The method should be able to learn a subspace sample by sample from highly outlier-contaminated observations. As Robust PCA has been shown to perform well in the task of foreground-background segmentation of video data, I develop a real-time video segmentation method based on Robust Subspace Tracking using a non-convex $\ell_0$-surrogate cost function.

### Robust Structured Low-Rank Approximation

Low-Rank approximation using a factorization approach has successfully been extended to the structured case, in which the admissible set of solutions is limited to those matrices that follow certain structural constraints. Yet, the structured problem is much less studied than the unstructured case, especially considering the robustification against outliers. Therefore I investigate how to employ a non-convex $\ell_0$ surrogate cost function as a data fitting term in order to find meaningful structured low-rank approximations even if the observed data contains large outliers. At the example of Time Series Analysis using low-rank Hankel matrices I furthermore examine how to benefit from the setting on the Grassmannian in the structured case. Precisely, how a previously found low-rank approximation can reduce the complexity of finding a structured low-rank approximation for related data in a similar way as for the subspace tracking problem in the unstructured case.

## 1.3. Contributions

This thesis investigates the Robust Low-Rank Approximation problem for static and temporally evolving subspaces and and its extension to structural constraints. All presented methods share the common approach of restricting the rank of the approximation with a factorization approach, while using a smoothed $\ell_p$-norm cost function to enforce sparsity on the residual error. The experimental results of this thesis show that extending the $\ell_p$ norm to the case $0 < p < 1$ leads to a closer approximation of the ideal $\ell_0$ norm and makes the approach even more robust against large outliers in the data. The smoothing parameter makes the cost function differentiable and allows for an efficient alternating minimization approach without additive slack variables. Imposing orthogonality on one of the factors lets this factor represent an orthogonal basis of the subspace to be found, whereas the other factor then contains the coordinates within this subspace. Optimization on the Grassmann manifold resolves ambiguities of the factorization as it establishes a one-to-one identification between a point on the manifold and a specific subspace.

For the static low-rank approximation problem, the Grassmannian Robust PCA (*GRPCA*) method is proposed, which reconstructs an underlying low-dimensional structure in the data from incomplete measurements, even when the low-rank and sparse assumptions on the data are not exactly met. An iterative shrinkage of the smoothing parameter allows for quick progress during the optimization. Furthermore, it is observed to reduce the risk of premature termination in local optima, which is an alleged weak point of non-convex approaches. Due to the factorization approach and the absence of additional slack variables the method requires a minimum amount of memory, as only the actually observed data and the commonly thin matrix factors need to be stored. A subsampled conjugate gradient algorithm together with a selective matrix-matrix product reduce the computational effort of evaluating the cost function and computing the gradient, which lets the algorithm scale well to large dimensions.

Due to the manifold setting on the Grassmannian, a Robust Subspace Tracking method can be laid out that can deal with temporally varying subspaces. Using a two-factor data model allows for efficient subspace and coordinate updates. The beneficial properties of the proposed sparsifying function transfer to the online setting and allow to track subspaces in conditions where $\ell_1$-based methods fail. A method for real-time video segmentation of real-world video scenes is presented, which achieves promising results for certain kinds of dynamic backgrounds on a video segmentation benchmark.

As a third contribution, the robustification of the structured low-rank approximation problem is studied, which extends the previously studied problems to the case of additive structural constraints. The Grassmannian Robust Structured Low-Rank-Approximation (*GRSLRA*) algorithm is proposed, which employs an efficient Augmented Lagrangian Multiplier scheme with alternating conjugate gradient optimization that scales well to large dimensions. Minimizing the residual error according to the smoothed $\ell_p$-norm cost function allows to find a meaningful structured low-rank approximation in challenging scenarios, such as system identification from outlier-contaminated impulse responses. Finally, a fast and robust method for online time series forecasting using Hankel-structured low-rank matrices is developed, which again exploits the manifold setting.

All algorithms presented in this work have been implemented in *Python* using the *NumPy* framework. Following the good practice of reproducible research, the source code of the algorithms and the scripts and parameter settings for all conducted experiments are publicly available for download at `https://github.com/clemenshage`. The experiments have been conducted inside a virtual environment on a desktop computer with a 3.3 GHz quad core CPU and 16 GB of RAM running Ubuntu 16.04.

## 1.4. Thesis Outline

After a first motivational overview on low-rank approximation problems has been laid out in this chapter, Chapter 2 takes a closer look at the prior art of solving the respective problems and gives an overview about relevant approaches in literature that have motivated the development of the methods presented in this work. Chapter 3 reviews the geometry of the optimization problems in this work and outlines the methods employed for solving them in an efficient way. The three main chapters 4 through 6 investigate three different Robust Low-Rank Approximation problems and how to solve them using a smoothed $\ell_p$-norm cost function while exploiting the manifold geometry of the Grassmannian. Chapter 4 presents the Grassmannian Robust PCA (*GRPCA*) method and compares its subspace reconstruction performance and its computational efficiency to several state-of-the-art methods, mainly on simulated data but also in a real-world video segmentation experiment. Chapter 5 considers the case of time-varying subspaces and presents a framework for addressing the Robust Subspace Tracking problem. Besides some experimental results on simulated data, a great part of the chapter is dedicated to the practical application of real-time video segmentation using the *pROST* ($\ell_p$-norm Robust Online Subspace Tracking) algorithm. Finally, Chapter 6 investigates the scenario of low-rank approximation with additional structural constraints and presents the Grassmannian Structured Low-Rank Approximation (*GRSLRA*) algorithm, together with a robust and efficient scheme for Online Time Series Analysis using low-rank Hankel matrices, which are evaluated on system identification and time-series forecasting tasks.

# Chapter 2.

# Related Work

## 2.1. Matrix Completion

The matrix completion problem (1.4) is an NP-hard problem and cannot be solved efficiently in this form. But it has been shown that solving a relaxed version of the problem allows to recover the missing data under mild conditions. Candès and Recht [21] restate the problem as

$$\min \|\boldsymbol{L}\|_* \quad \text{subject to} \quad \mathcal{P}_{\Omega}(\boldsymbol{L}) = \mathcal{P}_{\Omega}(\boldsymbol{X}) \tag{2.1}$$

with the nuclear norm (A.9) as the convex relaxation of the non-convex rank constraint. They also derive a lower bound for the required number of samples to guarantee exact data recovery. One important assumption on the data is made, namely that the singular vectors (i.e. the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ if $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$) must be sufficiently spread. This means that the energy must not be concentrated on one single entry as it would be the case for a canonical basis vector $\boldsymbol{e}_i$ that contains a 1 at position $i$ and zeros elsewhere. A coherence measure in this sense is proposed and a bound is derived on how incoherent both row and column space must be to maintain recovery guarantees. To solve the problem, a semidefinite program (SDP) is proposed. Unfortunately, standard SDP solvers such as *SeDuMi* do not scale well to higher dimensional problems, which limits the feasibility of this approach to small dimensions.

The singular value thresholding technique ($SVT$) proposed by Cai et al. [17] is based on the same convex relaxation, but it allows to tackle large-scale problems. As the name of the method indicates, the first step in the approach consists of computing the SVD and soft-thresholding the singular values of the optimization variable, i.e. setting all singular values to zero if they are smaller than a certain threshold. In a second correction step, the residual error of the intermediate solution is scaled with a suitable step size and added to the optimization variable. These two steps are iteratively performed until convergence, while monotonously shrinking the step size parameter after each iteration. It needs to be noticed that the soft thresholding in $SVT$ differs from the hard thresholding in the truncated SVD (1.3), where a fixed number of singular values are annihilated regardless of their value. Soft thresholding controls the magnitude of the residual error at the cost of losing control over the actual rank of the approximation. In contrast, hard thresholding sets an upper limit for the rank while minimizing the residual error in $\ell_2$ sense. The difference between both methods becomes even more relevant when additive Gaussian noise is considered [20]. In this scenario, either the rank is minimized for a given bound $\epsilon$ on the noise via

$$\min \operatorname{rank}(\boldsymbol{L}) \quad \text{subject to} \quad \|\mathcal{P}_{\Omega}(\boldsymbol{X}-\boldsymbol{L})\|_F^2 < \epsilon \tag{2.2}$$

or, if the rank should be bounded instead of the residual error, the residual error of the approximation is minimized as in the original Matrix Completion statement (1.4).

A simple way of guaranteeing an upper bound on the rank of the approximation is to use the bilinear factorization model $\boldsymbol{L} = \boldsymbol{A}\boldsymbol{Y}$, in which a low-rank matrix $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(L) \leq k$ is modeled as the product of two rectangular matrices $\boldsymbol{A} \in \mathbb{R}^{m\times k}$ and $\boldsymbol{Y} \in \mathbb{R}^{k\times n}$, respectively. Recht et al. [71] state the matrix completion problem using bilinear factorization as

$$\min_{\boldsymbol{A},\boldsymbol{Y}} \tfrac{1}{2}\left(\|\boldsymbol{A}\|_F^2 + \|\boldsymbol{Y}\|_F^2\right) \quad \text{subject to} \quad \mathcal{P}_{\Omega}(\boldsymbol{A}\boldsymbol{Y}) = \mathcal{P}_{\Omega}(\boldsymbol{X}) \ . \tag{2.3}$$

They propose to solve the problem with an alternating minimization approach in which one variable is fixed while optimizing the other, and vice versa. Although the two subproblems may be convex, the overall optimization problem is non-convex [71]. Thus it is hard to prove the success of the method, which might depend on the initialization of the factors. Nevertheless, the factorization approach has gained a lot of interest, especially in Big Data applications. On the one hand, alternating minimization can be faster than nuclear norm minimization approaches, and on the other hand the memory footprint can be reduced tremendously as only the two factors need to be stored instead of the whole low-rank matrix. This makes the approach applicable to large scale problems such as the data set of the *Netflix challenge* [54]. In the context of recommender systems, one of the factors can be interpreted as the latent factors or features of the rated items while the other matrix contains the user-specific weightings or coefficients.

Cabral et al. [15] analyze the relation between the factorization model (2.3) and the convex relaxation (2.1) and find that the factorization method achieves equivalent results compared to the nuclear norm relaxation if the rank is chosen large enough. Also, in the case where the actual rank of the data is known the reconstruction is generally better using the factorization approach.

Jain et al. [49] have presented a detailed investigation on the convergence behavior of alternating minimization methods for the Matrix Completion problem. They conclude that if the data is well-conditioned, global optimality of this type of methods can be guaranteed. Specifically, not only must the underlying subspace be sufficiently incoherent as already discussed for the convex relaxation approaches [23, 21] but the incoherence property must hold for each intermediate iterate in the alternating minimization process. Furthermore, they propose to subsample the observation set randomly in each step as the support of the original observation set $\Omega$ might be far from being uniformly distributed. Although the two factors in Equation (2.3) are unconstrained, in practice it is often useful to impose additional constraints, such as sparsity for sparse PCA or nonnegativity for Nonnegative Matrix Factorization (NMF). For the matrix completion problem Jain et al. [49] propose to orthogonalize one of the factors after each alternating minimization step as means of regularization.

This regularization issue has been discussed by Dai and Milenkovic [28], stating that the factorization $\boldsymbol{AY}$ is not unique as a different factorization $(\boldsymbol{AM})(\boldsymbol{M}^{-1}\boldsymbol{Y})$ with any invertible $k \times k$-Matrix $\boldsymbol{M}$ could be found. Therefore, they propose the factorization $\boldsymbol{UY}$ instead, with $\boldsymbol{U} \in \mathrm{St}_{m,k}$ being an element of the set of $k$-dimensional orthogonal frames in $m$-dimensional space, i.e. $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$. Moreover, $\boldsymbol{U}$ in this case is not just an orthogonal matrix but needs to be understood as a representation of the unique subspace that is spanned by its columns. Such a unique subspace again is an element of the Grassmannian, the manifold of $k$-dimensional subspaces in $m$-dimensional space. They minimize the residual error $\|\mathcal{P}_\Omega(\boldsymbol{X} - \boldsymbol{UY})\|_F^2$ via alternating minimization as proposed by Recht et al. [71], only that the minimization over $\boldsymbol{U}$ is performed via gradient descent on the Grassmannian.

The spectral matrix completion method *OptSpace* by Keshavan et al. [52] starts from an initial guess of the low-rank component obtained via hard singular value thresholding, i.e. truncated SVD. After this projection the found row and column spaces (i.e. the matrices containing the left and right singular values) are refined with a gradient descent algorithm on the Grassmannian as well. In a subsequent report [51] a rank estimation technique is proposed that searches for a significant drop in the distribution of singular values as to estimate the dimension of the subspace.

Boumal and Absil [9] build upon the approaches by Dai and Milenkovic [28] and Keshavan and Montanari [51], respectively, and propose the cost function

$$\min_{\boldsymbol{U},\boldsymbol{Y}\in\mathbb{R}^{k\times n}} \tfrac{1}{2}\left\|\mathcal{P}_\Omega\left(\boldsymbol{W}\odot(\boldsymbol{X}-\boldsymbol{U}\boldsymbol{Y})\right)\right\|_F^2 + \tfrac{\lambda_1}{2}\left\|\boldsymbol{Y}\right\|_F^2 - \tfrac{\lambda_2}{2}\left\|\mathcal{P}_\Omega\left(\boldsymbol{U}\boldsymbol{Y}\right)\right\|_F^2, \tag{2.4}$$

where $\boldsymbol{W}$ is a weighting matrix that weighs the element-wise residual error of the reconstruction through Hadamard multiplication, and the other terms regularize the magnitude of the entries in the low-rank approximation. Note that the magnitudes of the entries of $\boldsymbol{U}$ are inherently bound by the orthogonality constraint. In this specific formulation the full computation of $\boldsymbol{U}\boldsymbol{Y}$ is avoided, as only the entries within the observation set need to be evaluated. This reduces the theoretical complexity of the algorithm significantly if only a few entries are observed. It needs to be noted, however, that selecting the specific entries in a practical implementation creates a large overhead compared to highly-optimized matrix-matrix computation routines. The authors propose a Riemannian trust-region method [1] for matrix completion on the Grassmannian, which outperforms competing methods like *SVT* [17] or *OptSpace* [53].

Lastly, the matrix completion problem has also been tackled as an optimization problem on the Riemannian manifold of fixed-rank matrices, such as the *LORETA* method by Shalit et al. [74] or the similar approach of Meyer et al. [65], which has been further improved by Vandereycken [79] by replacing the gradient descent with a conjugate gradient descent method. The fixed-rank approach relates to the special case of the factorization approach where both factors are bound to have full rank, e.g. arising from a symmetrically partitioned SVD like in the balanced factorization model [65]. As a downside, this restricts the space of possible solution to matrices whose rank is exactly $k$, so that approximations with a rank strictly lower than $k$ are outside the feasible set.

## 2.2. Robust PCA

Over the course of the past years, the term *Robust PCA* has more and more been associated with the low-rank-and-sparse data model (1.6), largely sparked by the work of Candès et al. [19]. Yet, it needs to be mentioned that earlier approaches towards robustifying classical PCA exist, cf. [29, 46, 50, 55, 77].

Using the $\ell_1$ norm as a convex relaxation of the $\ell_0$ norm and relaxing the low-rank constraint with the nuclear norm, Candès et al. [19] state the convex program *Principal Component Pursuit* as

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{s.t.} \quad \boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S}, \tag{2.5}$$

which aims at blindly decomposing $\boldsymbol{X}$ into the sum of a low-rank matrix $\boldsymbol{L}$ and a sparse matrix $\boldsymbol{S}$. At first sight it seems that the problem of recovering $\boldsymbol{L}$ and $\boldsymbol{S}$ from the mixture is highly ill-posed, as more values need to be recovered than the number of measurements. Also, in contrast to the missing entries in the matrix completion problem, the number and position of the sparse outliers in the Principal Component Pursuit are unknown a priori. Nevertheless, using a similar incoherence principle as for matrix completion, successful recovery can be guaranteed under mild conditions, cf. also the work of Chandrasekaran et al. [23]. Intuitively, the reconstruction of a low-rank matrix will only be successful if the left and right basis vectors are sufficiently distant from the canonical basis vectors, resulting in a low-rank component that is dense. Vice versa, the support of the sparse components is assumed to be random in this data model and thus the sparse component cannot be low-rank.

A detailed investigation of the problem and its relations to matrix completion is also provided by Wright et al. [81], who propose an iterative thresholding algorithm derived from the *SVT* method [17]. However, as pointed out by Candès et al. [19], iterative thresholding techniques are quite slow. Instead, they propose to solve problem (2.5) with the Augmented Lagrangian Multiplier (ALM) [6] type method by Lin et al. [60] due to its faster convergence. In this approach, the side condition in Equation (2.5) is integrated into the Lagrangian function

$$\|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{X} - \boldsymbol{L} - \boldsymbol{S} \rangle + \tfrac{\mu}{2} \|\boldsymbol{X} - \boldsymbol{L} - \boldsymbol{S}\|_F^2 , \tag{2.6}$$

and the problem becomes an alternating minimization over $\boldsymbol{L}$, $\boldsymbol{S}$ and the Lagrangian parameter $\boldsymbol{\Lambda}$, which has been proposed in a similar way by Yuan and Yang [84]. All three separate optimization problems can be solved in closed form. The optimization over $\boldsymbol{L}$ is done by soft-thresholding the singular values of the term $\boldsymbol{X} - \boldsymbol{S} + \frac{1}{\mu}\boldsymbol{\Lambda}$, which can be understood as removing the estimate of the sparse outliers from the data and reducing the rank of this term. The update for $\boldsymbol{S}$ on the other hand is performed by thresholding the entries of $\boldsymbol{X} - \boldsymbol{L} + \frac{1}{\mu}\boldsymbol{\Lambda}$, which extracts large entries in the residual error of the low-rank approximation. The overall problem can either be solved by alternating between the optimization of $\boldsymbol{L}$ and $\boldsymbol{S}$ until convergence before updating $\boldsymbol{\Lambda}$ and $\mu$, which is referred to as the exact ALM method (*EALM*). Or just one optimization step is performed for optimizing $\boldsymbol{L}$ and $\boldsymbol{S}$, respectively, before updating $\boldsymbol{\Lambda}$. This variant is known as the inexact ALM (*IALM*), leading to similar solutions in practice at lower computational cost. Remarkably, the value for the parameter $\lambda$ weighing between low-rank and sparsity does not need to be tuned, but it can be fixed as $\lambda = \frac{1}{\sqrt{n}}$, where $n$ is the dimension of the square input matrix $\boldsymbol{X}$.

After Robust PCA with incomplete observations

$$\min_{\operatorname{rank}(\boldsymbol{L}) \le k} \left\| \mathcal{P}_{\Omega} \left( \boldsymbol{X} - \boldsymbol{L} \right) \right\|_0 \tag{2.7}$$

had already been discussed in [19], the authors of *SpaRCS* [80] extend the problem even further by considering more general linear measurement operators than the canonical one used for matrix completion. The algorithm iteratively updates the estimation of $\boldsymbol{L}$ and $\boldsymbol{S}$, borrowing ideas from from compressive sampling [66] and matrix completion [57] for the particular estimation steps.

As for matrix completion, there also exist Robust PCA methods that bound the dimension of the subspace approximation instead of performing convex relaxation on the rank. The greedy algorithm *GECO* [73] reconstructs a data set iteratively based on SVD while increasing the rank of the approximation in each step. Zhou and Tao [85] consider the data model $\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S} + \boldsymbol{N}$ with an additive Gaussian noise term $\boldsymbol{N}$. Their *GoDec* method alternates between estimating a low-rank component of bounded rank via hard-tresholding of the singular values and detecting sparse outliers of known cardinality with a hard thresholding step on the entries of the residual. A significant speedup is achieved by replacing the singular value truncation step with bilinear random projections. Mateos and Giannakis [64] describe Robust PCA from a viewpoint that is very closely related to traditional PCA.

While the low-rank component in the low-rank-and-sparse model (1.6) is assumed to be centered around zero, the optimization problem in [64]

$$\min_{\boldsymbol{m},\boldsymbol{U},\boldsymbol{Y},\boldsymbol{S}} \left\| \boldsymbol{X} - \boldsymbol{m}\boldsymbol{1}_n^\top - \boldsymbol{U}\boldsymbol{Y} - \boldsymbol{S} \right\|_F^2 + \lambda \left\| \boldsymbol{S} \right\|_0 \tag{2.8}$$

incorporates an explicit estimation of the mean vector $\boldsymbol{m}$. As the data is assumed to contain outliers albeit at unknown positions ($\boldsymbol{S}$ initialized with all zeros), the initial estimation of $\boldsymbol{m}$ is performed by computing the coordinate-wise median over the outlier-contaminated samples. The subsequent alternating minimization procedure starts by removing the mean and the detected outlier entries from the input data, computing the principal components $\boldsymbol{Y}$, updating the subspace estimate via SVD and thresholding the residual to find outliers in the data. Finally, the mean is computed over the data after removing the newly estimated outliers and the process is repeated. The outlier estimation step via soft thresholding as described in Algorithm 1 in [64] corresponds to relaxing the $\ell_0$ norm in Equation (2.8) with the $\ell_{1,2}$ norm, thereby regarding the columns of $\boldsymbol{X}$ either as inlier samples or as outlier samples that must not be considered for the subspace estimation. The authors stress that other surrogates for the $\ell_0$ norm could be considered in order to improve the sparsity of $\boldsymbol{S}$. As this makes the overall problem non-convex, they propose to run their convex approach first and then to refine their solution using a concave $\ell_0$-surrogate term with an iteratively reweighted least squares scheme as proposed in [22]. Shen et al. [75] state the problem as

$$\min_{\boldsymbol{A},\boldsymbol{Y},\boldsymbol{Z}} \left\| \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{Z} \right) \right\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{Y} - \boldsymbol{Z} = 0 \; . \tag{2.9}$$

Their *LMaFit* method solves the problem with a combination of the method of Alternating Directions and the Augmented Lagrangian Multiplier (ALM) method. The Lagrangian function

$$\mathcal{L}\left( \boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{\Lambda} \right) = \left\| \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{Z} \right) \right\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{A}\boldsymbol{Y} - \boldsymbol{Z} \rangle + \tfrac{\beta}{2} \left\| \boldsymbol{A}\boldsymbol{Y} - \boldsymbol{Z} \right\|_F^2 \tag{2.10}$$

is optimized via alternating minimization over $\boldsymbol{A}, \boldsymbol{Y}, \boldsymbol{Z}$, followed by updating the Lagrangian multiplier $\boldsymbol{\Lambda}$. Although the factors are not restricted a priori, $\boldsymbol{A}$ is orthogonalized via QR decomposition after each iteration to stabilize the approach. The method starts with an initial (over-estimated) value for the rank and subsequently tries to estimate the actual rank from the R factor of the (sorted) QR decomposition. The non-convex nature of the problem and especially the varying estimate of the rank make a convergence analysis difficult. However, experimental results illustrate that *LMaFit* surpasses *IALM* [60] and the very similar but computationally more expensive approach by Yuan and Yang [84]. Recently, Cambier and Absil [18] have presented a robust matrix completion method on a fixed-rank manifold, which is a robustified version of the $\ell_2$-based matrix completion algorithm by Vandereycken [79]. The modified cost function combines the restriction of the $\ell_1$-penalized residual to the set of observed entries with an $\ell_2$ regularization over the entries where the data is not observed, as proposed also in [9]. As the $\ell_1$ norm is not differentiable, it is replaced by the smoothed function $\sqrt{\mu^2 + x^2}$. In a similar manner as the Huber loss function [45], it resembles the $\ell_1$ norm whenever $x^2$ is large while being smooth around zero. The smoothing parameter is initialized with a large value for fast initial convergence and shrunk after each iteration.

## 2.3. Subspace Tracking

Subspace Tracking has been an active field of research for many years with applications predominantly in the field of signal processing [27]. An important task here is to determine a signal's covariance matrix, which is commonly estimated by averaging over the outer product of $T$ samples, such that $\widehat{\boldsymbol{C}}_{(t)} = \sum_{\tau=0}^{T} \boldsymbol{x}_{(t-\tau)} \boldsymbol{x}_{(t-\tau)}^\top$. Thus, the estimate of the covariance matrix and with it the underlying subspace of the signal are updated with every new sample. A computationally efficient algorithm is the *PAST* algorithm [83], which tracks a subspace by minimizing the error between an accumulated number of samples and their projection onto this subspace. In order to account for recent samples more than for samples in the past, the sum of the individual errors is weighted with a forgetting factor. Although the projection onto the signal subspace does not include any orthogonality constraints, the authors report that the resulting basis is observed to be close to orthogonal. In their *GROUSE* method, Balzano et al. [4] solve the minimization problem

$$\min_{\boldsymbol{U}_{(t)} \in \mathcal{U}, \boldsymbol{y}_{(t)} \in \mathbb{R}^k} \left\| \mathcal{P}_{\Omega_{(j)}} \left( \boldsymbol{x}_{(t)} - \boldsymbol{U}_{(t)} \boldsymbol{y}_{(t)} \right) \right\|_2^2 \tag{2.11}$$

at every time instance $t$. They consider a partial observation of the data vectors on an index set $Omega_{(t)}$ (here, $\mathcal{P}_\Omega$ represents the one-dimensional case of Equation (1.5)). In the same way as for the Matrix Completion algorithm by Dai and Milenkovic [28], $\boldsymbol{U}_{(t)}$ is an orthogonal frame that spans the subspace in which the data sample lies at time instance $t$, while $\boldsymbol{y}_{(t)}$ are the coordinates in this subspace. The search space for $\boldsymbol{U}_{(t)}$ is limited to a set $\mathcal{U}$, which encompasses the neighborhood of the previously estimated subspace estimate on the Grassmannian. In practice, the search space is restricted by limiting the admissible step size for the update of the subspace. As Balzano et al. [4] point out, the problem of estimating this subspace while sampling the data incompletely column by column is equivalent to the Matrix Completion problem if the actual underlying subspace is time invariant. However, other than in [28], the optimization of $\boldsymbol{U}$ in *GROUSE* is a column-wise stochastic gradient descent on the Grassmannian, which can be even more randomized in the batch case by randomly permuting the columns of the input data. While the accuracy of *GROUSE* in the matrix completion task lies in the range of competing state-of-the-art methods, the method is computationally much more efficient as it avoids the computation of matrix decompositions such as QR or SVD. In the case where the subspace is changing over time the algorithm is able to follow these changes, which is demonstrated by tracking suddenly changing or rotating subspaces.

Similar to *GROUSE*, the *PETRELS* algorithm [25] tracks a subspace over time from incomplete observations with an $\ell_2$ measure for penalizing the residual error $\boldsymbol{x}_{(t)} - \boldsymbol{A}_{(t)}\boldsymbol{y}_{(t)}$. The algorithm also alternates between estimating the coefficient vector $\boldsymbol{y}$ of the current sample using the previously estimated subspace and adjusting the subspace estimate $\boldsymbol{A}$ based on the past observations, but no orthogonality constraints are imposed on $\boldsymbol{A}$. Thus, the minimization of the projection error during the coordinate update requires the computation of the pseudo-inverse of $\boldsymbol{A}$, whereas an orthogonal frame such as $\boldsymbol{U}$ in *GROUSE* generally allows the much simpler solution $\boldsymbol{y} = \boldsymbol{U}^\top \boldsymbol{x}$. The subspace update of *PETRELS*

$$\min_{\boldsymbol{A}\in\mathbb{R}^{m\times k}} \quad \sum_{\tau=1}^{T} \beta^{T-\tau} \left\| \mathcal{P}_{\Omega} \left( \boldsymbol{x}_{(t-\tau)} - \boldsymbol{A}\boldsymbol{y}_{(t-\tau)} \right) \right\|_2^2 , \tag{2.12}$$

seems more complex than the update in *GROUSE* as well, as it considers not only the current sample but the past $T$ observations, which are weighted with a discounting factor $\beta < 1$. Yet, the authors show that due to the quadratic loss function the estimation of the subspace $A_{(t)}$ can be written as a recursive update of the previous estimation $A_{(t-1)}$. Furthermore, the estimation can be performed independently on the rows of $\boldsymbol{A}$, which allows to parallelize the operation. As a result, *PETRELS* achieves similar complexity as *GROUSE* and performs similarly well on matrix completion tasks, while the authors claim superior performance in terms of adaptation speed for subspace tracking. But it is disputable whether this is actually due to ”barriers in the search path on the Grassmannian” as Chi et al. [25] state (cf. also the discussion in [28]) or whether a suboptimal step size has been chosen for the gradient descent of *GROUSE* in these experiments.

### Robust Subspace Tracking

One of the first discussions on robustifying Subspace Tracking against outliers in the data is presented by Li et al. [58]. In this approach, the subspace is learned incrementally by updating an estimate of the covariance matrix with each new input vector and determining the dominant subspace via Eigenvalue Decomposition. While this inherently minimizes the squared projection error, the authors investigate the possibility of robustifying their incremental PCA by introducing a weighting step. Before updating the subspace with the new data sample the residual error of the projection on the past estimate of the subspace is computed and the input sample is weighted according to this residual. Experimental results show that their robust incremental PCA is a viable approach for video segmentation. Due to the incremental approach, the low-rank background model can adapt to persistent changes in the scene while the robust weighting helps to reduce ghosting artifacts from sparse foreground objects.

Qiu and Vaswani [70] take on Robust Subspace Tracking from a compressed sensing perspective with their *ReProCS* method. Given an initial estimate of the current subspace, they try to eliminate the low-rank component from the mixture by projecting the data onto the orthogonal complement of the subspace. When the estimate is close to the actual subspace, all that is left is to reconstruct the sparse outlier entries from a noisy observation. Temporal changes in the subspace are accounted for by updating the estimation with a recursive PCA approach. Comparing their approach to Candès et al. [19], the authors of *ReProCS* state that their algorithm is not only able to operate online, but also that it can conceptually deal with sparse outlier matrices that are not full rank and with a low-rank component of higher rank. However, the initial estimate of the subspace can only be estimated offline from outlier-free samples.

Mateos and Giannakis [64] extend their Robust PCA approach to Robust Subspace Tracking by restating their cost function in vectorized form with a weighting factor $\beta$ as

$$\min_{\boldsymbol{m},\boldsymbol{U},\boldsymbol{y},\boldsymbol{s}} \sum_{t=1}^{T} \beta^{T-t} \left\| \boldsymbol{x}_{(t)} - \boldsymbol{m} - \boldsymbol{U}\boldsymbol{y} - \boldsymbol{s}_{(t)} \right\|_2^2 + \lambda \left\| \boldsymbol{s}_{(t)} \right\|_2^2 . \tag{2.13}$$

Whenever a new sample appears, a soft thresholding step classifies the sample either as an inlier sample ($\boldsymbol{s}_{(t)} = \boldsymbol{0}$) or as an outlier sample. In the latter case, a line-search method estimates its entries. Then the coefficient vector $\boldsymbol{y}$ is updated in a similar way as for the *PAST* algorithm [83]. The subspace estimate $\boldsymbol{U}$ is updated with a recursive least-squares and, as the authors report, commonly keeps its orthogonality as reported in [83]. Finally, the mean vector is re-computed as a weighted average over the past observations' means. An experimental evaluation of the method on outlier-contaminated data proves the increased robustness against outliers in comparison to the $\ell_2$-based methods *GROUSE* [4] and *PAST* [83].

He et al. [42] present *GRASTA*, a robustified version of *GROUSE* that uses the $\ell_1$ norm instead of the $\ell_2$ norm in the loss function (2.11). Using a similar regularization as the *LMaFit* [75] approach for Robust PCA, the Augmented Lagrangian function to be minimized for the partial observation $\mathcal{P}_\Omega(\boldsymbol{x})$ of a new sample turns out as

$$\mathcal{L}(\boldsymbol{U},\boldsymbol{y},\boldsymbol{s},\boldsymbol{\lambda}) = \|\boldsymbol{s}\|_1 + \boldsymbol{\lambda}^T \left(\mathcal{P}_\Omega(\boldsymbol{x}) - \mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{y}+\boldsymbol{s})\right) + \frac{\rho}{2} \|\mathcal{P}_\Omega(\boldsymbol{x}) - \mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{y}+\boldsymbol{s})\|_2^2 \tag{2.14}$$

with the temporal indices omitted for clarity. The subspace-representing matrix $\boldsymbol{U}$ is optimized with a single gradient descent step on the Grassmannian, in a similar way as for the *GROUSE* method. The remaining variables, however, cannot be optimized in closed-form. Instead, a separate optimization loop is proposed, in which the respective variables are optimized in an alternating manner before updating the subspace again. In contrast to *GROUSE*, the step size for the subspace update in *GRASTA* cannot be fixed but needs to be selected adaptively, as to guarantee full converge to a feasible stationary solution while still being able to follow changes in the subspace. Experimental results on simulated data show that the method is suitable to track subspaces in the presence of large outliers in the data. *GRASTA* is furthermore evaluated on a real-world video segmentation task to confirm that the method is capable of updating its background model dynamically in an online setting.

## 2.4. Structured Low-Rank Approximation

The term Structured Low-Rank Approximation (SLRA) describes the general problem of finding a low-rank approximation that preserves matrix structures. While this understanding generalizes to any linear structure, the restriction of the problem to certain specific structures has a much longer history. Toeplitz matrices, for example, play an important role in Signal Processing, where SLRA is used as a tool for denoising [26]. Hankel structures on the other hand appear in Linear Time Invariant (LTI) system theory, where denoising and model reduction problems can be written as SLRA with Hankel constraints. The system theory with Hankel matrices has inspired many methods for Hankel-SLRA, such as the Singular Spectrum Analysis (SSA) proposed by Broomhead and King [12]. The method is quite simple, as it computes a low-rank approximation of a Hankel-structured matrix followed by a so-called diagonal averaging step, the orthogonal projection onto the space of Hankel matrices. Generally, this projection will not maintain the low-rank property established before, so the resulting outcome will likely have a rank higher than targeted. Cadzow's Method [16] alternates between these two steps until the algorithm converges to a solution that is indeed low-rank and structured. However, as stated in [26] the solutions produced by this approach can in practice be far away from the actual optimum. In fact, there are no guarantees of finding an actually meaningful approximation to the data. Chu et al. [26] themselves present a method that uses various equality constraints to solve the SLRA problem for the special case of symmetric Toeplitz matrices, but an extension to the general SLRA problem is not explicitly provided. Markovsky [61] considers SLRA with affine matrix structures and gives a comprehensive overview of existing SLRA approaches with real-world use-cases. In the discussed problems the fit between the data and the approximation is predominantly computed with the standard $\ell_2$ norm, but an extension to weighted low-rank approximation is outlined, which eventually leads to an SLRA algorithm that can deal with missing data [63]. A toolbox for SLRA is presented in [62] that solves a wide range of SLRA problems with (mosaic) Hankel structure. The solver aims at finding a matrix $\boldsymbol{R} \in \mathbb{R}^{m-k\times m}$ that minimizes

$$\min_{\boldsymbol{R}\in\mathbb{R}^{m-k\times m}} \left( \min_{\hat{\boldsymbol{x}}\in\mathbb{R}^N} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_w^2 \quad \text{subject to} \quad \boldsymbol{R}\boldsymbol{X}_{\mathcal{S}} = 0 \right), \tag{2.15}$$

where $\boldsymbol{x}$ is a given data vector that produces a structured matrix $\boldsymbol{X}_{\mathcal{S}} \in \mathcal{S}$ and $\|\cdot\|_w^2$ denotes the weighted $\ell_2$ norm. The feasibility of the approach arises from the fact that the inner minimization problem can be solved in closed form for a given $\boldsymbol{R}$. As all matrices $\boldsymbol{R}$ with the same row space produce the same solution for the inner minimization problem, this invariance carries over to the outer minimization problem, which is in fact an optimization problem on the Grassmannian [62]. Ishteva et al. [48] review existing SLRA approaches and find that the factorization model (as discussed for unstructured low-rank approximation problems) has gained rather little attention in the community. They consider structured input $\boldsymbol{X}_{\mathcal{S}}$ and propose a cost function for SLRA that joints the structural and low-rank constraint, resulting in the problem

$$\min_{\boldsymbol{A}\in\mathbb{R}^{m\times k},\boldsymbol{Y}\in\mathbb{R}^{k\times n}} \left\|\boldsymbol{x} - \boldsymbol{S}^{\dagger}\mathrm{vec}\left(\boldsymbol{A}\boldsymbol{Y}\right)\right\|_w^2 + \lambda \left\|\boldsymbol{A}\boldsymbol{Y} - \mathit{\Pi}_{\mathcal{S}}\left(\boldsymbol{A}\boldsymbol{Y}\right)\right\|_F^2 \tag{2.16}$$

where $\boldsymbol{x}$ is the given data vector, $\mathrm{vec}\left(\cdot\right)$ denotes the column-wise vectorization of a matrix, $\boldsymbol{S}^{\dagger}$ denotes the orthogonal projection of a vectorized arbitrary matrix onto the underlying data vector of the closest structured matrix and $\mathit{\Pi}_{\mathcal{S}}\left(\cdot\right)$ denotes the orthogonal projection of an arbitrary matrix onto the set of structured matrices (see Section 6.1 for a more detailed derivation). The dimension of the two matrix factors is an upper bound on the rank of the approximation, so that the resulting low-rank matrix $\boldsymbol{L} = \boldsymbol{A}\boldsymbol{Y}$ has a rank of at most $k$. The first term of Equation (2.16) controls the element-wise data fit between the input data and the approximation with a weighted $\ell_2$ norm, in order to ensure that the found approximation is close to the input. The second term, on the other hand, penalizes the distance between the low-rank approximation and its element-wise quadratic distance to the closest structured matrix. Ergo, it vanishes if the low-rank approximation fulfills the structural constraints of $\mathcal{S}$. The optimization procedure alternates between minimizing the cost function over $\boldsymbol{A}$ and over $\boldsymbol{Y}$, respectively, while the individual optimization problems can be stated as least-squares problems. Whenever an intermediate solution has been found (i.e. the progress in minimizing the cost function falls below a certain threshold) the penalty parameter $\lambda$ is increased. The overall procedure is terminated if $\lambda$ is sufficiently large for the low-rank approximation to fulfill the structural constraint.

As previously discussed for unstructured low-rank approximation, algorithms based on the $\ell_2$ norm bring along an inherent vulnerability against outliers in the data, which has led to the advancement from PCA to Robust PCA in the unstructured case. Ayazoglu et al. [3]

have therefore proposed to extend this concept to structured matrices with their Structured Robust PCA (SRPCA). Precisely, they extend Lin et al. [60]'s Robust PCA approach by the additional constraints that both the low-rank approximation and the sparse outliers have to meet a certain structure $\mathcal{S}_L$ and $\mathcal{S}_S$, respectively. The objective reads as

$$\min_{\boldsymbol{L},\boldsymbol{S}} \sum_i w_i \sigma_i(\boldsymbol{L}) + \|\boldsymbol{W}_S \boldsymbol{S}\|_1 + \tfrac{1}{2}\|\boldsymbol{W}_F \boldsymbol{S}\|_F^2 \quad \text{subject to} \quad \boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S}, \quad \boldsymbol{L} \in \mathcal{S}_L, \quad \boldsymbol{S} \in \mathcal{S}_S \tag{2.17}$$

where $w_i\sigma_i$ denote the weighted singular values and $\boldsymbol{W}_S$ and $\boldsymbol{W}_F$ are weighting matrices. The mixture of $\ell_1$- and $\ell_2$-regularization of $\boldsymbol{S}$ in Equation (2.17) differs from classic Principal Pursuit (2.5) but also from noisy Robust PCA [85] in that the entries are forced to be both sparse and small in magnitude, thus possibly containing both Gaussian noise and large outliers. Furthermore, $\boldsymbol{S}$ should be structured, which makes it even more difficult to predict the result of the interplay of the two norms. As all experiments in [3] are conducted with $\boldsymbol{W}_F = 0$, it seems reasonable to disregard the third term in the cost function, or to replace it by $\|\boldsymbol{X} - \boldsymbol{L} - \boldsymbol{S}\|_F^2$ as to allow for additive Gaussian noise. The SRPCA problem is solved with an augmented Lagrangian multiplier approach that joints the side condition $\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S}$ as in the unstructured case, cf. [60]. The structural constraint for $\boldsymbol{L}$ is taken care of by replacing it with an unstructured variable $\boldsymbol{J}$ and adding the constraint that $\boldsymbol{J} = \boldsymbol{L}_{\mathcal{S}}$ where $\boldsymbol{L}_{\mathcal{S}}$ is always a structured matrix. The same procedure is undertaken with the sparse component $\boldsymbol{S}$, resulting in a Lagrangian function with three Lagrangian multipliers and a sum of nine terms in total. As the authors report, the resulting problem is convex and can be solved with conventional convex optimization tools. However, as typically the case for Semidefinite Programming routines, the memory requirements can be prohibitive even for comparably small problems. The performance of the Robust SLRA approach by Ayazoglu et al. [3] is demonstrated on several experiments using Hankel structures. In the Target Location Prediction task, the goal is to forecast a trajectory based on its previous temporal development, which is a classic time series analysis problem applied to the vision setting. Similarly, the structured matrix completion problem can be used to match tracklets, and outliers can be removed from a trajectory by composing a Hankel matrix and performing Robust SLRA.

# Chapter 3.

# Optimization on the Grassmannian

In this work the factorization model $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{Y}$ with an $m \times k$-dimensional matrix $\boldsymbol{U}$ and $\boldsymbol{Y} \in \mathbb{R}^{k\times n}$ will be employed in order to account for the low-rank constraint $\operatorname{rank}(\boldsymbol{L}) \leq k$. In order to fix the scaling of one of the factors and to allow that the upper limit $k$ on the rank is always attainable, the columns of the factor $\boldsymbol{U}$ are constrained to be orthogonal, i.e. $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$. Thereby, they form an orthogonal basis of the subspace which is to be found, whereas the entries of $\boldsymbol{Y}$ are the coordinates within this subspace. In order to measure the residual error between input data $\boldsymbol{X}$ and a low-rank approximation, an entry-wise loss function

$$f : \mathbb{R}^{m\times n} \to \mathbb{R}, \quad \boldsymbol{X} \mapsto f(\boldsymbol{X}) \tag{3.1}$$

is considered, which is assumed to be differentiable and separable, i.e. the overall residual error is the sum of the residual errors in the respective coordinates. Using this function, the low-rank approximation problem can be written as

$$\min_{\boldsymbol{U},\boldsymbol{Y}} f(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Y}). \tag{3.2}$$

This chapter addresses the questions of how to deal with ambiguities in the factorization model and how to minimize a separable loss function efficiently over the respective variables. Imposing orthogonality constraints on $\boldsymbol{U}$ acts as a regularizer for the scaling and guarantees that the upper bound on the rank can always be achieved during an alternating optimization scheme between $\boldsymbol{U}$ and $\boldsymbol{Y}$. Minimizing the loss function over $\boldsymbol{U}$ with respect to the orthogonality constraints requires certain tools of *Manifold Optimization*, which will be laid out in this chapter. Precisely, a conjugate gradient method on the Grassmannian will be derived, which is the manifold of $k$-dimensional subspaces. The general idea of manifold optimization [34] is to examine the set of possible solutions to a given problem and, given that this set has a manifold structure, to derive a geometric description of the manifold. Starting at some element and taking a reasonably sized step along some direction on this manifold, one reaches an element which yields a lower value of the cost function. Repeating this process, a locally or even globally optimal solution to the problem is eventually found. The main difficulties are that many concepts of the common Euclidean space such as directions and angles are generally not defined on a manifold. A variety of literature has therefore been dedicated to the field of Geometric Optimization. For topics exceeding the derivation of the Grassmannian CG method, the interested reader is referred to the work of Edelman et al. [32] and Absil et al. [2]. Some specific insight on Grassmannian optimization and a discussion of conjugate gradient methods on the Grassmannian are provided in [47].

## 3.1. Stiefel and Grassmann manifolds

### Stiefel manifold

While the *orthogonal group*

$$O(m) := \{\boldsymbol{Q} \in \mathbb{R}^{m\times m} \mid \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}_m\} \tag{3.3}$$

describes all quadratic orthogonal matrices, the set of all rectangular $m \times k$ matrices with orthogonal columns is known as the *Stiefel* manifold

$$\mathrm{St}_{k,m} := \{\boldsymbol{U} \in \mathbb{R}^{m\times k} \mid \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k\}. \tag{3.4}$$

The Stiefel manifold can be derived as a Riemannian submanifold of $\mathbb{R}^{m\times k}$ and leads to the topology described by Absil et al. [2]. Thereby, elements of $\mathrm{St}_{k,m}$ are uniquely identified with an $m \times k$-dimensional matrix $\boldsymbol{U}$ as in Equation (3.4). The Stiefel manifold is endowed with the standard inner product $\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \mathrm{tr}\left(\boldsymbol{B}^\top \boldsymbol{A}\right)$ inherited from the surrounding Euclidean space. For a submanifold $\mathcal{M}$ that is embedded in a surrounding space (e.g. a sphere in Euclidean space) it is straightforward to define a *tangent space* $T_{\boldsymbol{X}}\mathcal{M}$ that contains the tangent vectors to all curves running through $\boldsymbol{X} \in \mathcal{M}$ (e.g. a tangent hyperplane to a sphere). At this particular point, the tangent space is a local vector space approximation of the manifold and allows to define distances and directions according to the chosen metric, since they are not defined on the manifold itself whenever the manifold is not a vector space. One can furthermore derive the projection of vectors from Euclidean space onto the tangent space and, respectively, the *normal space*, its orthogonal complement [2, 32].

Besides the definition as an embedded submanifold, Edelman et al. [32] investigate the derivation of the Stiefel manifold as a quotient manifold

$$\mathrm{St}_{k,m} := O(m)/O(m-k). \tag{3.5}$$

In this definition one element of $\mathrm{St}_{k,m}$ is represented by an equivalence class

$$\left\{ \boldsymbol{Q} \begin{bmatrix} \boldsymbol{I}_k & 0 \\ 0 & \boldsymbol{Q}_{m-k} \end{bmatrix}, \quad \boldsymbol{Q}_{m-k} \in O(m-k) \right\} \tag{3.6}$$

of all matrices out of $O_m$ whose first $k$ columns are identical. As Absil et al. [2] state, the concept of tangents of a quotient manifold is much more abstract in comparison to the intuitive geometry of embedded submanifolds. If $[\boldsymbol{X}]$ is an element of the quotient manifold, the tangent space of $[\boldsymbol{X}]$ can be derived by partitioning the tangent space at $\boldsymbol{X}$ into the directions within the equivalence class and those across equivalence classes. The former, i.e. the directions that are tangent to the equivalence class and that lead to other representatives of the same equivalence class, are contained in the *vertical space*. Its orthogonal complement is described as the *horizontal space* and contains only the directions that lead to other equivalence classes. As a consequence, the inherited metric (being the orthogonal group metric in the definition (3.5)) must be restricted to the horizontal space only, which represents the tangent space of the quotient manifold [32].

For the Stiefel manifold the quotient manifold perspective seems overly complex, as the embedded submanifold definition allows for a more efficient representation with a unique mapping from matrices to Stiefel elements. Moving along to the Grassmannian, however, the benefits of the quotient manifold derivation will become more obvious as the representation of its elements is less straightforward.

### Grassmann manifold

The columns of a Stiefel element $\boldsymbol{U}$ span one specific $k$-dimensional subspace within $m$-dimensional space. But it is quite obvious that the same subspace could be spanned as well by a different (rotated) basis $\tilde{\boldsymbol{U}} := \boldsymbol{U}\boldsymbol{Q}$ with $\boldsymbol{Q} \in O(k)$. Consequently, a low-rank matrix $\boldsymbol{L}$ that can be factorized into $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{Y}$ can equivalently be factorized into $\tilde{\boldsymbol{U}}\tilde{\boldsymbol{Y}} := \boldsymbol{L}\boldsymbol{Q}_m\ \boldsymbol{Q}^T\boldsymbol{Y}$. This motivates the definition of an equivalence class

$$[\boldsymbol{U}] := \{\boldsymbol{U}\boldsymbol{Q} \quad | \quad \boldsymbol{Q} \in O(k)\}\ . \tag{3.7}$$

A specific $k$-dimensional subspace is then represented *uniquely* by one particular equivalence class. The set of all $k$-dimensional subspaces is known as the *Grassmann manifold* or the *Grassmannian*, named after the mathematician Herrmann Günther Graßmann[1] and is denoted by $\mathrm{Gr}_{k,m}$.

Edelman et al. [32] define the Grassmannian as a quotient manifold $St_{k,m}/O(k)$ arising from the division of the Stiefel manifold (in the definition (3.5)) by the $k$-dimensional orthogonal group, so that

$$\mathrm{Gr}_{k,m} := \{[\boldsymbol{U}] \quad | \quad \boldsymbol{U} \in \mathrm{St}_{k,m}\} \tag{3.8}$$

with the equivalence relation (3.7). This representation has the advantage that only $m \times k$-dimensional matrices appear in practical computations involving elements of the Grassmannian. Yet, one needs to keep in mind that the stored representation is just one possible representative out of all the equivalence class members.

Whenever a unique matrix representation is required, the Grassmannian can alternatively be defined as

$$\mathrm{Gr}_{k,m} := \{\boldsymbol{P} \in \mathbb{R}^{m\times m} \mid \boldsymbol{P} = \boldsymbol{U}\boldsymbol{U}^\top, \quad \boldsymbol{U} \in \mathrm{St}_{k,m}\}, \tag{3.9}$$

where $\boldsymbol{P}$ is an $m \times m$-dimensional orthogonal projector that projects a vector $\boldsymbol{x} \in \mathbb{R}^m$ onto a specific $k$-dimensional subspace. If $k \ll m$, however, which is the case in many

[1]Graßmann (1809 - 1877) studied these concepts around 1840 as an autodidact and developed groundbreaking concepts of Linear Algebra in his book *Ausdehnungslehre.* However, his work was not respected at the time due to its abstract notation. Rumor has it that his publication was used by his publisher as waste paper. When similar concepts finally became en vogue in the 1860s and 1870s Graßmann had already resigned all his mathematical work.

practical applications, a submanifold embedding of $\mathrm{Gr}_{k,m}$ in $\mathbb{R}^{m\times m}$ inflates the dimensions of the problem tremendously. A possible way to alleviate the issue besides resorting to the quotient manifold definition is to develop efficient *retractions*, which provide a local transformation of a manifold optimization problem to an equivalent problem on its tangent space, cf. [2].

On a side note, a third definition of the Grassmannian arises from deriving it as a quotient manifold of the orthogonal group via $O(m)/\left(O(m)\times O(m-k)\right)$. However, the resulting representation with $m\times m$ equivalence classes is similarly memory-consuming as working with projectors.

## 3.2. Quotient geometry of the Grassmannian

In the following, the geometry of the Grassmannian as a quotient manifold will be recalled, starting with the metric that is inherited from the Stiefel manifold. If the Stiefel manifold is derived as a quotient manifold from the orthogonal group, it inherits the orthogonal metric, which is not equivalent to the Euclidean metric. However, as Edelman et al. [32] state, the necessary restriction of the orthogonal group metric to the horizontal space at $\boldsymbol{U}\in\mathrm{St}_{k,m}$ leads to a metric that again is equivalent to the Euclidean metric. Thus the standard inner product holds in the tangent space of the Grassmannian, which simplifies its description. The tangent space of the Grassmannian $\mathrm{Gr}_{k,m}$ at a point $[\boldsymbol{U}]$, which is the horizontal space at $\boldsymbol{U}\in\mathrm{St}_{k,m}$, can be described as

$$T_{[\boldsymbol{U}]}\,\mathrm{Gr}_{k,m} := \{\boldsymbol{H}\in\mathbb{R}^{m\times k}\mid \boldsymbol{U}^\top\boldsymbol{H}=0\}, \tag{3.10}$$

where $\boldsymbol{H}$ represents any possible tangent direction. An element $\boldsymbol{Z}\in\mathbb{R}^{m\times n}$ from the surrounding space can be projected onto the tangent space $T_{[\boldsymbol{U}]}\,\mathrm{Gr}_{k,m}$ via

$$\Pi_{T_{[\boldsymbol{U}]}\,\mathrm{Gr}_{k,m}}(\boldsymbol{Z}) = (\boldsymbol{I}_m-\boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{Z}. \tag{3.11}$$

Moving across the manifold requires a description of its *geodesics*, which are the shortest paths from one element on the manifold to another one (such as great circles on a sphere). The generic way of computing the geodesics involves the matrix exponential, which is costly to evaluate in practice. Following the derivation in [32] instead, the geodesics emanating from a point $[\boldsymbol{U}]$ on the Grassmannian can efficiently be computed as

$$\gamma_{[\boldsymbol{U}]}(t, \boldsymbol{H}) = (\boldsymbol{U}\boldsymbol{V}_H \cos(\boldsymbol{\Sigma}_H t) + \boldsymbol{\Theta}_H \sin(\boldsymbol{\Sigma}_H t))\boldsymbol{V}_H^\top, \tag{3.12}$$

with $\boldsymbol{\Theta}_H \boldsymbol{\Sigma}_H \boldsymbol{V}_H^\top$ being the compact Singular Value Decomposition of a tangent direction $\boldsymbol{H} \in T_{[\boldsymbol{U}]} \operatorname{Gr}_{k,m}$.

Knowledge of the geodesics also allows to derive a formula for the *parallel translation* between tangent spaces. Since the directions are not defined on the manifold itself but only within the tangent spaces at distinct points on the manifold, comparing two directions at different points on the manifold requires to transfer one direction along the geodesics to the respective tangent space of the other. For the quotient manifold definition, Edelman et al. [32] derive the translation of an element $\boldsymbol{G}$ from the tangent space at $[\boldsymbol{U}]$ in the direction $\boldsymbol{H} \in T_{[\boldsymbol{U}]} \operatorname{Gr}_{k,m}$ as

$$\tau_{[\boldsymbol{U}]}\left(\boldsymbol{G}, t, \boldsymbol{H}\right) = \boldsymbol{G} - \left(\boldsymbol{\Theta}_H \left(\boldsymbol{I}_k - \cos(\boldsymbol{\Sigma}_H t)\right) - \boldsymbol{U}\boldsymbol{V}_H \sin(\boldsymbol{\Sigma}_H t)\right) \boldsymbol{\Theta}_H^\top \boldsymbol{G} \tag{3.13}$$

with $t \in \mathbb{R}$ defining the length of the path along the geodesic (3.12) [2].

## Cost functions and their domains

After the geometry of the problem has been identified, the individual cost functions for the alternating minimization of the overall objective (3.2) can be defined. The cost function for the optimization with respect to $\boldsymbol{U}$ can be stated as

$$f_U : \operatorname{St}_{k,m} \to \mathbb{R}, \quad \boldsymbol{U} \mapsto f\left(\boldsymbol{X} - \boldsymbol{U}\boldsymbol{Y}_0\right) \tag{3.14}$$

with a fixed set of coefficients $\boldsymbol{Y}_0$. Vice versa, considering a fixed $\boldsymbol{U}_0 \in \operatorname{St}_{k,m}$ the cost function for $\boldsymbol{Y}$ is

$$f_Y : \mathbb{R}^{k \times n} \to \mathbb{R}, \quad \boldsymbol{Y} \mapsto f\left(\boldsymbol{X} - \boldsymbol{U}_0\boldsymbol{Y}\right) \tag{3.15}$$

As discussed above, $\boldsymbol{U}$ is an element of the Stiefel manifold, but it only serves as a matrix representation of the equivalence class $[\boldsymbol{U}] \in \operatorname{Gr}_{k,m}$. In order to resolve the ambiguities of

[2] On a side note, if the transported direction coincides with the direction of transport, (3.13) simplifies to $\tau_{[\boldsymbol{U}]}(\boldsymbol{H}, t, \boldsymbol{H}) = \left(\boldsymbol{\Theta}_H \cos(\boldsymbol{\Sigma}_H t) - \boldsymbol{U}\boldsymbol{V}_H \sin(\boldsymbol{\Sigma}_H t)\right) \boldsymbol{\Sigma}_H \boldsymbol{V}_H^\top$ .

the factorization model, $f_U$ is actually optimized on the Grassmannian instead of the Stiefel manifold. Therefore, the optimization strategy consists of alternating between the separate minimization problems

$$\min_{[\boldsymbol{U}]\in \mathrm{Gr}_{k,m}} f_U\left(\boldsymbol{U}\right) \tag{3.16}$$

and

$$\min_{\boldsymbol{Y}\in \mathbb{R}^{k\times n}} f_Y\left(\boldsymbol{Y}\right). \tag{3.17}$$

## 3.3. Gradient and Conjugate Gradient Descent on the Grassmannian

What follows is an explanation of how to solve the optimization problem (3.16) on the Grassmannian. The derived concepts of tangent space, projection onto the tangent space and geodesics allow for a simple *Gradient Descent* (GD) method. As a first step, the Euclidean gradient $\nabla f_U\left(\boldsymbol{U}\right)$ of the cost function with respect to $\boldsymbol{U}$ is computed. Following Edelman et al. [32], its projection onto the tangent space at $[\boldsymbol{U}]$ is equivalent to the *Riemannian Gradient* and to the projection onto the column space of $\boldsymbol{U}_\perp$, the orthogonal complement of $\boldsymbol{U}$. This again can efficiently be performed by subtracting all parts of the gradient that are in the column space of $\boldsymbol{U}$ via

$$\boldsymbol{G} := (\boldsymbol{I}_k - \boldsymbol{U}\boldsymbol{U}^\top)\nabla f_U\left(\boldsymbol{U}\right). \tag{3.18}$$

Starting at an initial point $\left[\boldsymbol{U}^{(0)}\right]$, one can now reduce the cost function (3.14) by walking along the geodesics (3.12) in the direction $\boldsymbol{H} := -\boldsymbol{G}$, the opposite direction of the Riemannian gradient. The procedure is outlined in Algorithm 3.1.

GD methods are computationally cheap compared to second order methods that also consider the Hessian of the cost function. Thus they are a practical tool for unconstrained non-convex optimization in high dimensions. However, it is also known that GD methods exhibit slow convergence speed under certain circumstances, e.g. when the directions of two subsequent steps are almost opposing each other in badly conditioned problems [76]. This

**Algorithm 3.1** Gradient Descent on the Grassmannian

```
input $\boldsymbol{U}_0$
  Initialize $\boldsymbol{U} := \boldsymbol{U}_0$
  repeat
    Evaluate gradient of the cost function $\nabla f(\boldsymbol{U})$
    Compute Riemannian gradient $\boldsymbol{G}$ via Eq. (3.18)
    Set search direction $\boldsymbol{H} = -\boldsymbol{G}$
    Compute reduced SVD of $\boldsymbol{H}$ for geodesics
    Determine step size $t$ with Alg. 3.2
    Update $\boldsymbol{U} \leftarrow \boldsymbol{U}(t)$ via Eq. (3.12)
  until converged
output $\hat{\boldsymbol{U}}$
```

can be overcome by considering the *Conjugate Gradient* (CG) method [43] instead, which will be employed to solve the majority of problems within this work.

The general idea of CG methods is to improve the effectiveness of the individual optimization steps by choosing in each iteration a search direction that is *conjugate* to the previous search directions. In the original derivation of CG as a method for minimizing linear quadratic forms of the kind $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$, the term conjugacy means $\boldsymbol{A}$-orthogonality with respect to the symmetric and positive-definite matrix $\boldsymbol{A}$. The search direction is then obtained by computing the momentary negative gradient and subtracting all parts that are not conjugate (i.e. $\boldsymbol{A}$-orthogonal) to the previous search directions. For a general and not necessarily convex cost function, conjugacy is commonly understood with respect to the second derivative. Since the explicit computation of the second derivative is costly, various gradient-based update rules have been proposed instead and have been adapted to the manifold setting. In this work the direction for the CG algorithm on the Grassmannian is obtained via the update rule

$$\boldsymbol{H}^{(i+1)} = -\boldsymbol{G}^{(i+1)} + \beta^{(i)}\tau(\boldsymbol{H}^{(i)}) \tag{3.19}$$

with

$$\beta^{(i)} = \frac{\langle \boldsymbol{G}^{(i+1)}, (\boldsymbol{G}^{(i+1)} - \tau(\boldsymbol{G}^{(i)}))\rangle}{\langle \tau(\boldsymbol{H}^{(i)}), (\boldsymbol{G}^{(i+1)} - \tau(\boldsymbol{G}^{(i)}))\rangle}, \tag{3.20}$$

where $\tau(\cdot)$ is an abbreviated notation of the transport $\tau_{[\boldsymbol{U}^{(i)}]}\left(\,\cdot\,, t^{(i)}, \boldsymbol{H}^{(i)}\right)$.

Many different options exist for updating $\beta$ in (3.19). The particular choice (3.20) is an adaptation of the original Hestenes-Stiefel formula [43], and it is used throughout the work due to its favorable convergence behavior. Figure 3.1 serves as an illustration of the problem's geometry and its relevant elements, especially the the progression of projecting the gradient of the cost function onto the current tangent space and transporting it along the geodesics into the tangent space of the subsequent subspace estimate.

## Step size selection

The remaining question is how the best *step size* $t$ can be determined in practice. Finding the optimum step size in an unconstrained optimization method is a whole field of research on its own and countless sophisticated methods have been proposed to solve it, cf. [67]. For the CG methods in this work the step size is determined with the backtracking line search algorithm listed in Algorithm 3.2. Starting with an initial value $t_{\text{start}}$, the step size $t$

**Algorithm 3.2** Backtracking line search on the Grassmannian

```
Choose $t_{\text{start}} > 0,\ c, \rho \in (0,1)$ and set $t \leftarrow t_{\text{start}}$
repeat
    $t \leftarrow \rho t$
until $f_U\left(\gamma_{[\boldsymbol{U}]}(t, \boldsymbol{H})\right) \leq f_U(\boldsymbol{U}) + c\, t \operatorname{tr}\left(\boldsymbol{G}^\top \boldsymbol{H}\right)$ (*Armijo-Goldstein condition*)
Choose step-size $t^{(i)} := t$
```

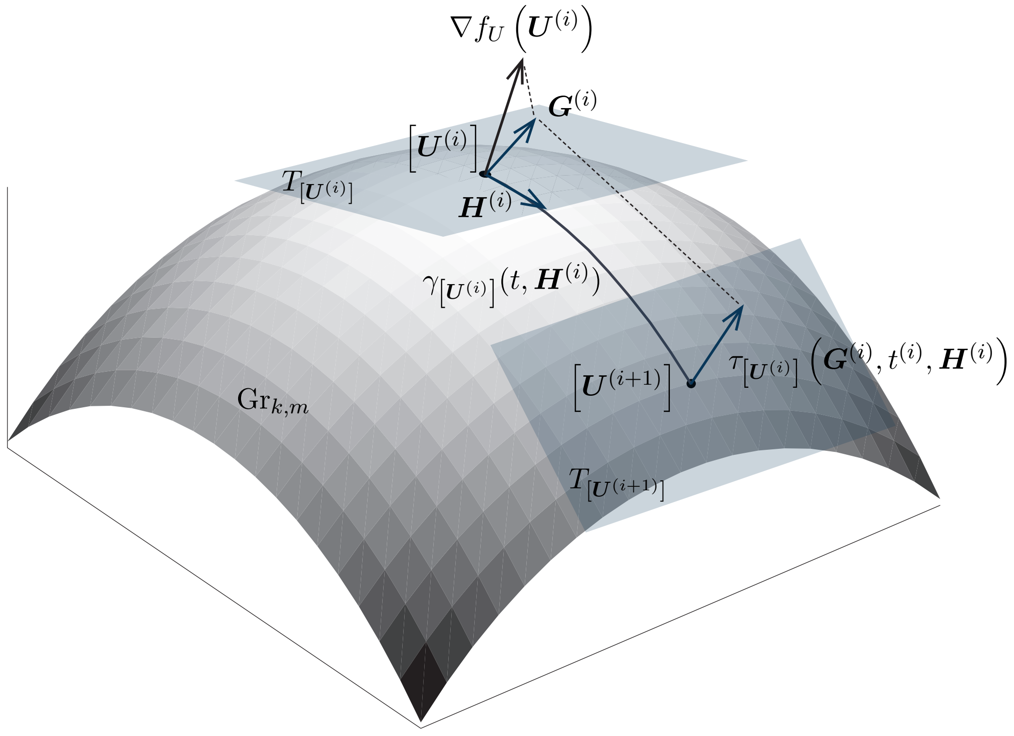

$\nabla f_U\left(\boldsymbol{U}^{(i)}\right)$
$\boldsymbol{G}^{(i)}$
$\left[\boldsymbol{U}^{(i)}\right]$
$T_{[\boldsymbol{U}^{(i)}]}$
$\boldsymbol{H}^{(i)}$
$\gamma_{[\boldsymbol{U}^{(i)}]}(t, \boldsymbol{H}^{(i)})$
$\left[\boldsymbol{U}^{(i+1)}\right]$
$\tau_{[\boldsymbol{U}^{(i)}]}\left(\boldsymbol{G}^{(i)}, t^{(i)}, \boldsymbol{H}^{(i)}\right)$
$\mathrm{Gr}_{k,m}$
$T_{[\boldsymbol{U}^{(i+1)}]}$

**Figure 3.1.:** Illustration of the Conjugate Gradient Descent method on the Grassmannian

is reduced in each iteration by multiplying $t$ with a factor $\rho$ with $0 < \rho < 1$. A preliminary step of length $t$ is undertaken by walking along the geodesics (3.12) in order to compute the cost function at $\gamma_{[U]}(t, \boldsymbol{H})$. The *Armijo-Goldstein condition* is evaluated to check whether the step size $t$ leads to a sufficient decrease in the cost function or whether the investigated step size is too large. Its evaluation requires the inner product between the Riemannian gradient $\boldsymbol{G}$ and the search direction $\boldsymbol{H}$ defined in the tangent space at $[\boldsymbol{U}]$ and a parameter $0 < c < 1$. Unless the step size passes the Armijo test, the step size is further reduced.

An overview of the Conjugate Gradient Descent on the Grassmannian is listed in Algorithm 3.3. The CG consists of several repetitions, each time employing the Backtracking Line

**Algorithm 3.3** Conjugate Gradient Descent on the Grassmannian

```
input $\boldsymbol{U}_0$
  Initialize $\boldsymbol{U} := \boldsymbol{U}_0$
  repeat
    Evaluate gradient of the cost function $\nabla f_U(\boldsymbol{U})$
    Compute Riemannian gradient $\boldsymbol{G}$ via Eq. (3.18)
    Compute $\beta$ via Eq. (3.20) and update search direction $\boldsymbol{H}$ via Eq. (3.19)
    Compute reduced SVD of $\boldsymbol{H}$ for geodesics and transport
    Determine step size $t$ with Alg. 3.2
    Update $\boldsymbol{U} \leftarrow \gamma_{[\boldsymbol{U}]}(t, \boldsymbol{H})$ via Eq. (3.12)
    Compute transported directions $\tau(\boldsymbol{G})$ and $\tau(\boldsymbol{H})$
  until converged
output $\hat{\boldsymbol{U}}$
```

Search (Algorithm 3.2) to find the optimum step size. After each iteration, the search direction is updated using the Hestenes-Stiefel update rule (3.19). As pointed out by Shewchuk [76], the convergence can be sped up with periodic restarts, i.e. through resetting the search direction after a certain number of iterations. Formally, $\beta^{(i)} = 0$ whenever $i \bmod j = 0$ for some $j \in \mathbb{N}_*$, so that $\boldsymbol{H}^{(i)} = -\boldsymbol{G}^{(i)}$ for these iterations. The CG method is considered converged when the step size returned by the line search falls below a certain threshold $t_{\min}$, but it may also be reasonable to terminate the algorithm prematurely after a certain number of iterations have been undertaken.

**Optimization of $\boldsymbol{Y}$ with Euclidean CG**

The minimization of $f_Y$ (3.15) in Euclidean space is a much simpler problem than the minimization of $f_U$ on the Grassmannian, and the procedure follows straightforwardly from Algorithm 3.3. Starting at an iterate $\boldsymbol{Y}$ with cost $f_Y(\boldsymbol{Y})$ and taking a step of length $t$ in the search direction $\boldsymbol{H}$ (obtained from the Euclidean gradient $\boldsymbol{G}$ and the previous search direction), the resulting cost for the new iterate is $f_Y(\boldsymbol{Y} + t\boldsymbol{H})$. In the line search procedure, the step size is slowly shrunk by setting $t \leftarrow \rho t$ until a new iterate is found that reduces the cost function significantly by means of fulfilling the Armijo-Goldstein condition

$$f_Y(\boldsymbol{Y} + t\boldsymbol{H}) \leq f_Y(\boldsymbol{Y}) + c\ t\ \operatorname{tr}\left(\boldsymbol{G}^\top \boldsymbol{H}\right) \tag{3.21}$$

with $c > 0$.

## 3.4. Conjugate Gradient Descent with subsampled line search

The main computational effort of first-order line search methods besides the computation of the gradient lies in the repetitive evaluation of the cost function in order to determine an appropriate step size. As Nocedal and Wright [67] describe, a trade-off has to be found between maximizing the reduction of the cost function per step on the one hand and minimizing the effort of determining a favorable step size that allows such a reduction on the other hand. If the focus is on fast processing time, a fixed step size should be selected. However, this comes at the risk of slowing down the convergence by selecting unnecessarily small or overly large steps and thereby achieving suboptimal progress in minimizing the cost function. On the contrary, high control over reducing the cost function is achieved with a backtracking line-search that guarantees sufficient decrease, e.g. according to the Armijo-Goldstein condition. An even more elaborate measure would be to check for the Wolfe conditions that also involve the magnitude of the gradient [67]. In practice, the computational effort of the backtracking line search is strongly depending on the selection of the initial step size and the contraction parameter $\rho$. While some rules for selecting these parameters exist (see e.g. [67]), the values are often tweaked and selected empirically. The smaller a value is chosen for $\rho$, the faster one step of the line search is performed as fewer step sizes need to be evaluated before eventually reaching a good estimate. But with increasing granularity of the search it may happen that the closest feasible step size misses the optimum step size, so that the actually achieved improvement is suboptimal. As a trade-off between speed and accuracy, $\rho$ can be varied throughout the optimization [67]. For example, it may be chosen quite small in the beginning of the alternating minimization and may be increased towards termination, which allows to make quick but coarse progress in the beginning and an optimum solution can still be reached in the end. Regarding the initialization of the step size, since the step sizes of subsequent CG iterations are often in the same range, it is reasonable to initialize the line search only slightly larger than the previous step size, instead of initializing with the largest possible step size in every step. Yet, after some steps a full search over the whole admissible range should be performed again.

### Separable cost function and subsampling

The loss function (3.1) has been assumed to be separable, i.e. for an $m \times n$-dimensional input $\boldsymbol{X}$ the value of the function is computed by summing over the contribution in every single entry. In order to make the function invariant to the dimensions of the problem, the separable cost function will be considered as

$$f(\boldsymbol{X}) = \frac{1}{mn} \sum_{i}^{m} \sum_{j}^{n} g(X_{ij}), \tag{3.22}$$

where $g(x)$ is a scalar function evaluating the residual error at one particular coordinate. As a result, the objective of the optimization problem is to minimize the reduction of the residual error averaged over all coordinates. This opens up the question whether it may be possible to estimate the reduction in the cost function from a subset of coordinates, as this could considerably reduce the computational effort of evaluating the cost function during the line search. For this purpose, the individual contributions $g(X_{ij})$ will be regarded in the following as $mn$ independent and identically distributed (i.i.d.) random variables following the unknown distribution of the residual error of the approximation. In this case, the above stated averaged sum over the individual observations is an estimate of its expected value, and the goal is to find a low-rank approximation that leads to a residual error distribution with minimal expected value. How well the expected value can be approximated from a subset of coordinates depends on the size of this subset and on the distribution of the residual error.

### Sparsifying cost function as a Bernoulli parameter estimator

As discussed in Chapter 1, the ideal sparsity measure is the $\ell_0$ norm, which takes on the value 0 only if its argument is zero and 1 otherwise. Averaging over the dimensions of the problem, one obtains the cost function

$$f_0 : \mathbb{R}^{m\times n} \to \mathbb{R}, \quad \boldsymbol{X} \mapsto f_0\left(\boldsymbol{X}\right) = \frac{1}{mn}\left\|\boldsymbol{X}\right\|_0 \tag{3.23}$$

which measures the density of the input. Minimizing this function should lead to a solution with a maximum possible number of entries that are zero. Taking on a stochastic point of view, the entry-wise contributions (0 if $X_{ij} = 0$ and 1 otherwise) shall be regarded as the outcome of $mn$ independent Bernoulli experiments with a common but unknown probability $p$ of an entry being 1. As a consequence, the loss function (3.23) estimates the expected value of a Binomial distribution $\mathcal{B}(nm, p)$ divided by the number of experiments, which again is the Bernoulli parameter $p$. Assuming now a set of coordinates $\Psi$ with $|\Psi| < mn$, evaluating the cost function (3.23) only on this subset leads to a subsampled estimation of the Bernoulli parameter, which shall be denoted by $p_\Psi$.

The first question now is how big a subsampling set $\Psi$ needs to be so that $p_\Psi$ is close to $p$ with high probability. Or, vice versa, considering a fixed set $\Psi$ of cardinality $|\Psi|$ on which the Bernoulli parameter $p_\Psi$ is estimated, how close is $p_\Psi$ to the actual parameter $p$. This is equivalent to the question how well the function value $f_0\left(\boldsymbol{X}\right)$ can be approximated by subsampling the function as $f_0\left(\mathcal{P}_\Psi\left(\boldsymbol{X}\right)\right)$. For $|\Psi|$ sufficiently large, $p_\Psi$ is a Gaussian-distributed random variable with mean $p$ and variance $\frac{p(1-p)}{|\Psi|}$. On the one hand, this means that the estimator is unbiased, i.e. the estimation is exact on average. On the other hand, if $p$ was known a priori, one could compute the number of samples required to guarantee that $p_\Psi$ lies within a certain error margin $(p - \epsilon, p + \epsilon)$ with a confidence of $(1 - \alpha)$ where $0 < \alpha < 1$. However, as $p$ is not known a priori, it must be replaced by an estimate $p^*$ in the computation of the cardinality

$$|\Psi| \geq \left(\frac{z_{\alpha/2}}{\epsilon}\right)^2 p^*(1-p^*), \tag{3.24}$$

with $z_{\alpha/2}$ being obtained from the cumulative distribution function of the standard distribution and depending on $\alpha$ (e.g. $z_{\alpha/2} \approx 1.96$ for $\alpha = 0.05$, i.e. 95% confidence that $p - \epsilon < p_\Psi < p + \epsilon$). Quite intuitively, the required sample size grows with the desired confidence and resolution of the estimation, and whenever $p$ cannot be estimated an upper bound is computed by selecting $p^* = 0.5$. It is important to notice that the required cardinality of $\Psi$ is independent of the dimensions of the problem, i.e. the success does not depend on the sampling rate but on the absolute amount of samples.

Letting $p^{(i)}$ and $p^{(i+1)}$ denote the parameters of the Bernoulli distribution corresponding to the distribution of zeros and nonzeros in two matrices $\boldsymbol{X}^{(i)}$ and $\boldsymbol{X}^{(i+1)}$, the actual progress in minimizing the cost function from iteration $i$ to $i+1$ is

$$\delta := f_0\left(\boldsymbol{X}^{(i)}\right) - f_0\left(\boldsymbol{X}^{(i+1)}\right) = p^{(i)} - p^{(i+1)}. \tag{3.25}$$

The estimated progress on the other hand, predicted from observation on the subset $\Psi$ only, is computed as

$$\delta_\Psi := f_0\left(\mathcal{P}_\Psi\left(\boldsymbol{X}^{(i)}\right)\right) - f_0\left(\mathcal{P}_\Psi\left(\boldsymbol{X}^{(i+1)}\right)\right) = p_\Psi^{(i)} - p_\Psi^{(i+1)}, \tag{3.26}$$

which is equivalent to the difference between the estimated parameters of the respective Bernoulli distributions.

Choosing the new iterate $\boldsymbol{X}^{(i+1)}$ based on its sparsity on $\Psi$ saves a lot of computational effort compared to computing the full sparsity pattern, but comes at the cost of reduced control on the actual overall progress. The chances that the new iterate is actually sparser than the previous one (i.e. $P(\delta > 0)$) depend on $|\Psi|$ and $\delta_\Psi$. If the error margin for the estimation of $p_\Psi^{(i)}$ and $p_\Psi^{(i+1)}$ is restricted to $\epsilon < \delta_\Psi/2$ and both parameters lie within this margin with a confidence of $(1-\alpha)$, then $P(\delta > 0 \mid \delta_\Psi > 2\epsilon) = (1-\alpha)^2$. This means that if the cost function is significantly reduced on a sufficiently large subset of coordinates, the probability of erroneously increasing the cost function over all coordinates becomes very small. As a result, the separability of the sparsifying function allows to estimate the overall sparsity on a coordinate subset instead of evaluating all coordinates. This leads to a constant complexity for evaluating the cost function during the line search, which lets the overall approach scale well to large dimensions.

**Practical applicability**

Obviously, the result is based on several assumptions that may be violated in a real-world application. It has been discussed that the ideal $\ell_0$ norm does not allows for a feasible minimization approach, which is why it is replaced in practice with a surrogate loss function. As a consequence, the cost function will be a suboptimal estimator of the sparsity. Also, the residual error of the individual coordinate entries may be statistically dependent and may be differently distributed depending on the nature of the data. Also, if additional Gaussian noise is present the residual error will not follow a Bernoulli distribution. Despite all these issues it has been observed empirically that for the optimization of $f_U$ and $f_Y$ evaluating the cost function only on a subset $\Psi$ is sufficient to find a reasonably good step size. This is likely due to the Armijo rule acting as a safeguard to prevent insufficiently small progress $\delta_\Psi$ on $\Psi$, which would raise the probability of an increase in the cost function evaluated on all observable coordinates. Most importantly, experimental results support the finding that the success of the method is only depending on the absolute cardinality of $\Psi$ and not on the sampling rate, i.e. for sufficiently large dimensions it is independent of the overall dimension of the data.

# Chapter 4.

# Grassmannian Robust PCA

It is well known that common PCA is susceptible to outliers in the data, as the residual error $\|\boldsymbol{X} - \boldsymbol{L}\|_F^2$ between the data and the low-rank approximation is measured in the $\ell_2$ sense. Due to the squared loss function, large entries in the residual are suppressed while small errors are tolerated, thus spreading the residual error across all coordinates. This is the optimal strategy for a low-rank plus Gaussian noise data model $\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{N}$ as all entries are evenly likely to be affected by noise and the noise energy is spread. But for the low-rank-and-sparse data model $\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S}$, the ideal loss function would be $\|\boldsymbol{X} - \boldsymbol{L}\|_0$, as the $\ell_0$ norm enforces as many entries as possible in the residual to become exactly zero, while ignoring the magnitude of the non-zero entries. As discussed in Section 2.2, the $\ell_0$ norm is commonly replaced by the $\ell_1$ norm, as it is the closest convex $\ell_0$ surrogate, with convexity being a crucial prerequisite for convergence analysis and recovery guarantees [19]. But by no means this implies that non-convex low-rank approximation methods are infeasible, with the convergence analysis by Jain et al. [49] being a prominent counterexample that shows that the (non-convex) factorization method is a viable approach for matrix completion. As a matter of fact, all optimization methods on the Grassmannian are inherently non-convex problems as the Grassmannian is a non-convex set. The observation that non-convex penalties like the $\ell_p$ norm with $0 < p < 1$ recover sparser solutions than the convex $\ell_1$ norm ([56, 35]) has been the main incentive to investigate the use of non-convex $\ell_0$-surrogate loss functions for Robust PCA on the Grassmannian [38].

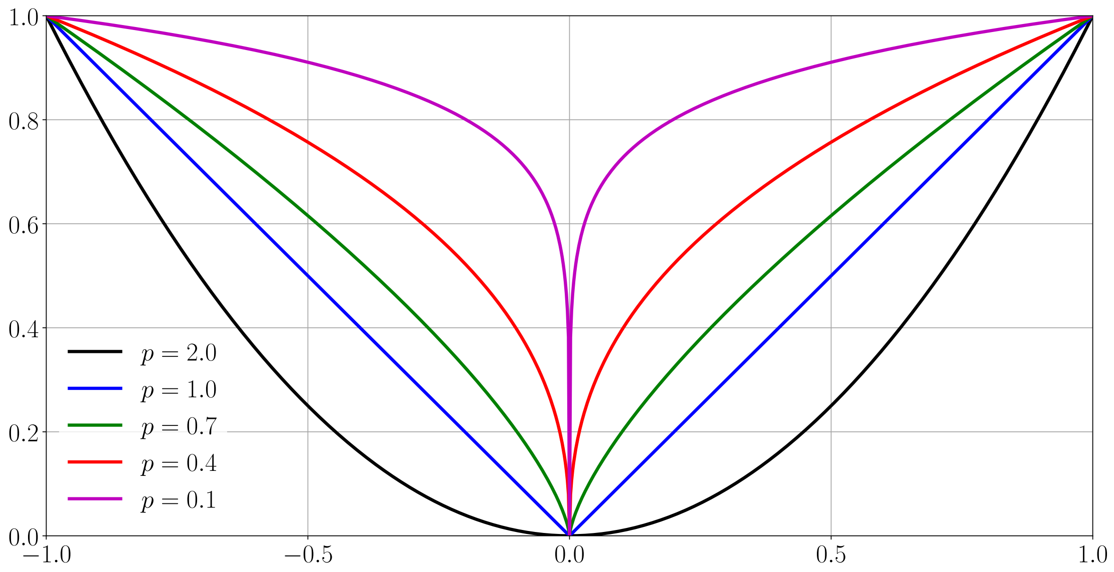


**Figure 4.1.:** Visualization of the normalized $\ell_p$ norm for scalar input for different values of the parameter $p$

## 4.1. Smoothed $\ell_p$-norm loss function

The general $\ell_p$ norm (cf. Section A.1.2) with $p \geq 1$ is defined as an entry-wise norm for vector or matrix-valued input. In the following, an element-wise sparsifying cost function will be derived, which is based on the extension of the $\ell_p$ norm to $0 < p < 1$. To ease the notation, the function will be derived for the scalar case and will later be extended again to vectors and matrices.

In search of an ideal sparsity measure, Fan and Li [33] postulate that its derivative should be infinite at zero (or next to zero in case that the measure is not differentiable at zero) and should vanish for great input values. While the $\ell_2$ norm exhibits quite an opposite behavior, the $\ell_1$ norm comes closer to this ideal definition, which explains its sparsifying behavior. Yet, lowering the value of $p$ below 1 lets the $\ell_p$ norm fulfill these conditions even better as can be seen from Figure 4.1. In contrast to the $\ell_2$ norm, which is steep on the outer end and shallow around zero, the $\ell_1$ norm has constant slope everywhere. For $p < 1$, the slope of the $\ell_p$ norm is shallow for large values and steep around zero. In the same way as the $\ell_1$ norm, the $\ell_p$ norm is not differentiable at zero, which hinders its use as a loss function for a gradient-descent based optimization problem. To leverage this issue, a smoothing parameter $\mu$ can be introduced so that a smoothed version of the $\ell_p$ norm

$$g_{p,\mu}(x) = \left(x^2 + \mu\right)^{\frac{p}{2}}, \quad 0 < p < 1, \quad \mu > 0, \tag{4.1}$$

is obtained, which is differentiable everywhere while approaching the actual $\ell_p$ norm in the limit $\mu \to 0$. Its derivative is

$$g'_{p,\mu}(x) := \frac{d}{dx} g_{p,\mu}(x) = p\ x \left(x^2 + \mu\right)^{\left(\frac{p}{2}-1\right)}. \tag{4.2}$$

In order to put the smoothed $\ell_p$ norm in perspective to the $\ell_1$ and $\ell_2$ norms for varying $\mu$, the offset resulting from introducing $\mu$ must be subtracted. As the input is commonly normalized to a magnitude range of $[0, 1]$ the function must be scaled so that it meets the $\ell_1$ and the $\ell_2$ norm at 1. The normalized smoothed $\ell_p$ norm

$$\bar{g}_{p,\mu}(x) := \frac{g_{p,\mu}(x) - g_{p,\mu}(0)}{g_{p,\mu}(1) - g_{p,\mu}(0)}, \tag{4.3}$$

fulfills $\bar{g}_{p,\mu}(0) = 0$ as well as $\bar{g}_{p,\mu}(1) = 1$ and its derivative is straightforwardly computed as

$$\bar{g}'_{p,\mu}(x) := \frac{1}{g_{p,\mu}(1) - g_{p,\mu}(0)}\ g'_{p,\mu}(x)\ . \tag{4.4}$$

Figure 4.2 compares the normalized smoothed $\ell_p$ norm for $p = 0.1$ and various values for the parameters $\mu$ to the $\ell_2$ and $\ell_1$ norm. The dominance of the smoothing parameter around zero creates a convex region, whose width depends on the particular value of $\mu$. Beyond the point of inflection the function is concave and the function becomes more and more shallow on the outer end. A quick computation reveals that the points of inflection are at

$$\frac{d^2}{dx^2}\ \bar{g}_{p,\mu}(x) = 0 \quad \Leftrightarrow \quad x = \pm\sqrt{\frac{\mu}{1-p}}. \tag{4.5}$$

For the special case $p = 1$, the smoothed $\ell_p$ norm is closely related to the Huber loss function

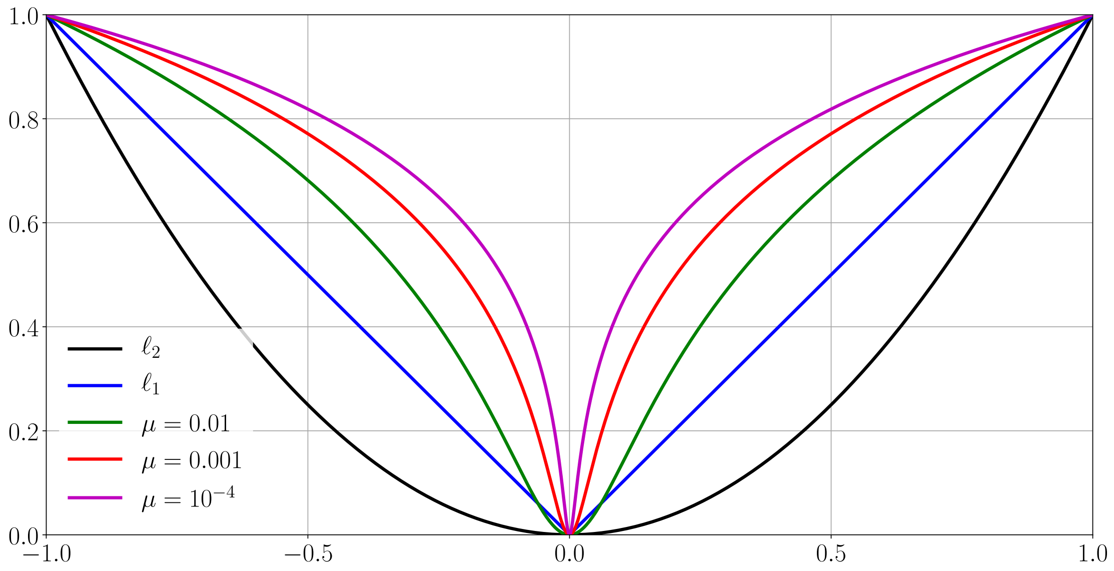


**Figure 4.2.:** Comparison between $\ell_2$ norm, $\ell_1$ norm and normalized smoothed $\ell_p$-norm loss function with $p = 0.1$ and varying parameter $\mu$

[45], which is defined piece-wise as

$$L_\tau(x) = \begin{cases} \frac{1}{2}x^2 & |x| \leq \tau, \\ \tau\left(|x| - \frac{1}{2}\tau\right) & \text{otherwise} \end{cases} \tag{4.6}$$

i.e. it is quadratic around zero and has constant slope for input values beyond a certain threshold $\tau$.

Beyond making the loss function differentiable everywhere, the smoothing parameter serves a second purpose with regard to additive Gaussian noise. Consider a data model $\tilde{\boldsymbol{X}} = \boldsymbol{L} + \boldsymbol{S} + \boldsymbol{N}$ with $\boldsymbol{L}$ and $\boldsymbol{S}$ as before and $\boldsymbol{N}$ being a dense noise matrix whose entries are i.i.d according to $\mathcal{N}(0, \sigma_N^2)$, the normal distribution with zero mean and variance $\sigma_N^2$. It becomes obvious that the strict low-rank-and-sparse decomposition cannot model additive Gaussian noise well. While a low-rank estimate $\hat{\boldsymbol{L}}$ can only contain a portion of the noise that lies in the signal subspace, a matrix $\boldsymbol{S} = \boldsymbol{X} - \hat{\boldsymbol{L}}$ resulting from enforcing sparsity on the residual cannot embody dense noise either, as it would not be sparse anymore. However, if the loss function is convex in a small area around zero and has a shallow slope for large arguments, the residual $\hat{\boldsymbol{S}}$ can contain some entries of large magnitude and many entries that are *almost* zero, which exactly corresponds to $\boldsymbol{S} + \boldsymbol{N}$. In this scenario sparsity is measured by counting the number of entries that are smaller than a given threshold $\tau$, which should be chosen according to the estimated noise level. Assuming Gaussian noise as before, then almost all noise is inside $(-3\sigma_N, 3\sigma_N)$. Thus, the smoothing parameter $\mu$ should be chosen such that the point of inflection (4.5) for the smoothed $\ell_p$ norm is at $\tau = 3\sigma_N$. For $0 < p < 1$, this is achieved with

$$\mu_{\text{opt}} = (1-p)\tau^2 = 9(1-p)\sigma_N^2 \ . \tag{4.7}$$

Empirical results show that it is beneficial to initialize $\mu$ with a larger value (such as $\mu_{\text{start}} = 0.1$) as the $\ell_2$-behavior of the cost function can speed up the convergence of the algorithm in the beginning. As long as sufficient progress is made in reducing the cost function the parameter $\mu$ is held constant. And whenever the algorithm is slowly converging to a local optimum (i.e. the progress in reducing the cost function falls below a certain threshold) the parameter is shrunk by updating $\mu \leftarrow c_\mu \mu$ for a fixed $0 < c_\mu < 1$. The process of shrinking $\mu$ is repeated until $\mu$ reaches a pre-defined minimum value $\mu_{\text{end}}$, so that the desired trade-off between noise tolerance and sparsity is achieved.

Apart from the $\ell_p$ norm, many other non-convex $\ell_0$ surrogates could be thought of, discussed among others by Gasso et al. [35]. Experimental results in [38] indicate that their performance as sparsifying functions for Robust PCA is not substantially different from the discussed smoothed $\ell_p$ norm (4.3) [38], which is why all experiments in this work are conducted with this particular choice.

## 4.2. Algorithmic description

In Chapter 3 the Low-Rank Approximation problem has been discussed in a generic way in order to simplify the deduction of the CG method on the Grassmannian. It is now time to extend the problem to the case of incomplete observations, to concretize the cost function for the Robust PCA problem and to describe how to solve it with an alternating minimization scheme.

As discussed for the $\ell_2$ case in Section 2.1, low-rank approximation can be used to fill in missing entries in a given data set. Assume that the data $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ is observed only on an index set $\Omega$, then $\mathcal{P}_{\Omega}(\boldsymbol{X})$ contains only the observed entries of $\boldsymbol{X}$. Therefore, the loss function $f(\boldsymbol{X}-\boldsymbol{U}\boldsymbol{Y})$ and its gradient need to be restricted to the observed coordinates $\Omega$, leading to a Robust Matrix Completion problem with the cost function $f(\mathcal{P}_{\Omega}(\boldsymbol{X}-\boldsymbol{U}\boldsymbol{Y}))$. Apart from this restriction on the index set, the algorithms for Robust PCA and Robust Matrix Completion are identical. Therefore, the term Robust PCA in this chapter encompasses both cases.

### 4.2.1. Alternating minimization scheme

The proposed scalar smoothed $\ell_p$-norm loss function (4.3) shall be employed as an entry-wise sparsifying function for Robust PCA. The overall loss is thus obtained by evaluating the entry-wise contributions and summing over all dimensions

$$h_{\mu}:\mathbb{R}^{m\times n}\to\mathbb{R},\quad \boldsymbol{X}\mapsto h_{\mu}(\boldsymbol{X})=\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\bar{g}_{p,\mu}(X_{ij})\,. \tag{4.8}$$

To improve the legibility, the cost function parameter $p$ is omitted in the final cost function (4.8). This also relates to the fact that the smoothing parameter $\mu$ is varied throughout the optimization, while $p$ is held constant. Letting $\boldsymbol{E}_{ij}$ denote the matrix with entry 1 at position $(i,j)$ and zeros elsewhere, the gradient of the separable loss function can be written as

$$\nabla h_{\mu}(\boldsymbol{X})=\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\bar{g}'_{p,\mu}(X_{ij})\,\boldsymbol{E}_{ij}. \tag{4.9}$$

For the case of incomplete observations, whenever $(i,j)\notin\Omega$ then the respective entry of $\nabla h_{\mu}(\boldsymbol{X})$ is zero. Using the smoothed $\ell_p$-norm loss function as a sparsity measure, the Robust PCA objective writes as

$$\min_{\boldsymbol{U}\in\mathrm{St}_{k,m},\boldsymbol{Y}\in\mathbb{R}^{k\times n}} h_{\mu}(\mathcal{P}_{\Omega}(\boldsymbol{X}-\boldsymbol{U}\boldsymbol{Y}))\,. \tag{4.10}$$

It will be solved by alternatingly minimizing the two separate loss functions

$$f_U:\mathrm{Gr}_{k,m}\to\mathbb{R},\quad \boldsymbol{U}\mapsto h_{\mu}(\mathcal{P}_{\Omega}(\boldsymbol{X}-\boldsymbol{U}\boldsymbol{Y}_0)) \tag{4.11}$$

with fixed coordinates $\boldsymbol{Y}_0$ and $\boldsymbol{U}$ being the representative of the equivalence class $[\boldsymbol{U}]$, and

$$f_Y : \mathbb{R}^{k \times n} \to \mathbb{R}, \quad \boldsymbol{Y} \mapsto h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{U}_0 \boldsymbol{Y} \right) \right) \tag{4.12}$$

with a fixed basis $\boldsymbol{U}_0$. The respective Euclidean gradients are

$$\nabla f_U \left( \boldsymbol{U} \right) = \nabla h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{U} \boldsymbol{Y}_0 \right) \right) \boldsymbol{Y}_0^\top \tag{4.13}$$

and

$$\nabla f_Y \left( \boldsymbol{Y} \right) = \boldsymbol{U}_0^\top \nabla h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{U}_0 \boldsymbol{Y} \right) \right). \tag{4.14}$$

Algorithm 4.1 lists the alternating minimization scheme for the proposed Grassmannian Robust PCA (*GRPCA*) method. The initialization of $\boldsymbol{U}$ and $\boldsymbol{Y}$ depends on the size of the data set. If it is feasible to compute the reduced SVD of an $(m \times n)$-dimensional matrix $\boldsymbol{X}_0$ with $\mathcal{P}_\Omega \left( \boldsymbol{X}_0 \right) = \mathcal{P}_\Omega \left( \boldsymbol{X} \right)$ then $\boldsymbol{U}_0$ can be selected as the first $k$ left singular vectors of $\boldsymbol{X}_0$ and the coefficient matrix is naturally initialized as $\boldsymbol{Y}_0 = \boldsymbol{U}_0^\top \boldsymbol{X}_0$ to complete the $\ell_2$-low-rank approximation. Alternatively, a random orthogonal $m \times k$ matrix can be chosen as $\boldsymbol{U}_0$, then $\boldsymbol{Y}_0 = 0$. The range for the parameter $\mu$ is defined a priori, as well as the shrinkage rate $c_\mu$. At the beginning of each iteration, a subset $\Psi$ is randomly drawn from the observation set $\Omega$, on which the subsampled line search (cf. Section 3.4) will be performed. Firstly, the loss function for $\boldsymbol{U}$ is minimized with a subsampled CG method on the Grassmannian. Then, based on the updated subspace estimate the estimate of $\boldsymbol{Y}$ is refined with a subsampled CG method in Euclidean space. After both factors have been updated the relative progress in decreasing the cost function is measured as

$$\delta^{(i+1)} := \frac{h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{U}^{(i)} \boldsymbol{Y}^{(i)} \right) \right) - h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{U}^{(i+1)} \boldsymbol{Y}^{(i+1)} \right) \right)}{h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{X} - \boldsymbol{U}^{(i)} \boldsymbol{Y}^{(i)} \right) \right)} \tag{4.15}$$

with $\left( \boldsymbol{U}^{(i)}, \boldsymbol{Y}^{(i)} \right)$ and $\left( \boldsymbol{U}^{(i)}, \boldsymbol{Y}^{(i+1)} \right)$ denoting the pair of variables before and, respectively, after the optimization. This time the cost function is evaluated not just on $\Psi$ as in the line search but over $\Omega$. If the relative progress is insufficient, the smoothing parameter $\mu$ is shrunk to overcome the risk of premature termination in a local optimum. Consequently, a new iteration is performed unless $\mu$ is smaller than the minimum value $\mu_{\text{end}}$ or a maximum

**Algorithm 4.1** Alternating minimization scheme for *GRPCA*

```
Input: $\mathcal{P}_{\Omega}(\boldsymbol{X})$
Initialize $\boldsymbol{U} = \boldsymbol{U}_0$, $\boldsymbol{Y} = \boldsymbol{Y}_0$
Choose $0 < c_\mu < 1, \mu_{\text{start}}$ and $\mu_{\text{end}}$, initialize $\mu = \mu_{\text{start}}$
while $\mu \geq \mu_{\text{end}}$ do
  Randomly draw $\Psi \subseteq \Omega$
  $\boldsymbol{U} \leftarrow \arg\min_{[\boldsymbol{U}] \in \mathrm{Gr}_{k,m}} f_U(\boldsymbol{U})$     (4.11)
  $\boldsymbol{Y} \leftarrow \arg\min_{\boldsymbol{Y} \in \mathbb{R}^{k \times n}} f_Y(\boldsymbol{Y})$     (4.12)
  if $\delta < \delta_{\text{min}}$ then
    $\mu \leftarrow c_\mu \mu$
  end if
end while
Outputs:
$\hat{\boldsymbol{L}} = \boldsymbol{U}\boldsymbol{Y}$
$\hat{\boldsymbol{S}} = \boldsymbol{X} - \hat{\boldsymbol{L}}$
```

number of iterations has been performed. In this case the algorithm terminates and the low-rank approximation may be computed as well as the sparse residual, which (depending on the application) might be thresholded in a post-processing step to separate between large outliers and Gaussian noise.

### 4.2.2. Input normalization

As discussed in Section 4.1, the proposed loss function is convex in a small region around zero and concave otherwise, with the point of inflection depending on the selection of parameters $p$ and $\mu$. In contrast to the $\ell_1$ norm, which has a constant slope everywhere, the sparsifying behavior of the smoothed $\ell_p$ norm depends on the magnitude of the entries in the residual $\boldsymbol{X} - \boldsymbol{L}$ and thereby on the scale of the input data. Furthermore, linear subspaces always pass through the origin (cf. Section A.1.1), so that only centered subspaces can be modeled. Thus, if a non-centered subspace of dimension $k$ should be found, the dimension needs to be increased to $k+1$ to account for the offset. Especially in the case of concentrated data with a large offset the first basis vector of the low-rank approximation will represent the offset itself. If one wishes to avoid this issue, the mean vector can be computed over all samples a priori and the data can be centered by subtracting the mean from every column. For the low-rank-and-sparse data model, however, centering the data is not that trivial. As the data model explicitly assumes outliers in the input, it is likely that these outliers will distort the mean estimation. An alternative way of centering the data in this scenario is to compute the median in each direction/ feature and to subtract the resulting median vector from every sample [64]. Subsequently, the data can be scaled to a range that meets the parameterization of the cost function, which intuitively is the range $[-1, 1]$. Naively, the scaling factor would be chosen so that the largest entry in the input data is scaled to a magnitude of 1. But since outliers are assumed to be present, a more robust choice is to scale the data according to the magnitude at a certain percentile. Assuming that the low-rank-and-sparse data model is the sum of an approximately Gaussian-distributed low-rank component and sparse outliers, the scaling should be chosen according to the statistical properties of a Gaussian distribution $\mathcal{N}\left(0, \sigma^2\right)$. Assuming that almost all data is contained within $3\sigma$, the targeted normalization is $3\sigma = 1$. This means that a robust scaling to the targeted range $[-1, 1]$ can be achieved by scaling the data so that the 68th percentile of the input magnitudes is scaled to a height of 0.33.

After a linear subspace estimate has been fitted to the centered and scaled data the resulting low-rank matrix is upscaled again and the median vector is added. As the explicit computation of $\boldsymbol{L}$ can be computationally demanding, the scaling is applied only to the coordinates $\boldsymbol{Y}$. To reverse the centering process, the subspace basis $\boldsymbol{U}$ is augmented with the median vector and the augmented basis is orthogonalized again.

### 4.2.3. Memory efficient processing for Big Data analysis

With growing dimension of the problem, the limiting factor for running Algorithm 4.1 is the requirement for computing and storing the $m \times n$ low-rank estimate $\boldsymbol{L} = \boldsymbol{UY}$. For $|\Omega| \ll mn$ it is thus reasonable to store only the actually observed entries of $\boldsymbol{X}$ and the factors $\boldsymbol{U}$ and $\boldsymbol{Y}$ instead of storing the full matrix $\boldsymbol{L} = \boldsymbol{UY}$ with zeros at $(i,j) \notin \Omega$. Regarding the computational expense, the theoretical complexity of computing $L_{ij} = \boldsymbol{u}_{i,:}^{\top} \boldsymbol{y}_j$ for all $(i,j) \in \Omega$ is $\mathcal{O}(|\Omega| r)$ and thus much smaller than $\mathcal{O}(mnr)$. In principle, this allows for a significant speedup of a selective matrix product compared to the full matrix multiplication, cf. [9]. Yet, it must be remarked that a conventional full matrix multiplication is a highly optimized operation compared to a prototype implementation of a selective matrix product. Therefore, in a practical implementation of Algorithm 4.1, the actual runtime of the selective matrix multiplication for the specific use-case is compared against the standard matrix product and a decision is made in favor of the faster version.

## 4.3. Evaluation on simulated data

Experiments on artificial test data are essential in order to measure the performance of the algorithm, to compare it against competing methods and to evaluate the influence of the algorithm's parameters. The way how test data is generated and how the recovery abilities of Robust PCA algorithms are evaluated varies across the literature, with the main differences being the dimensions of the problem, the subspace dimensions, the sparsity, the magnitude of the outliers as well as the presence or absence of additive Gaussian noise, and lastly, the percentage of revealed entries for incompletely observed data (Robust Matrix Completion). As a common baseline, all evaluations assume the low-rank-and-sparse data model (1.6) and the general evaluation procedure is the following:

1. Choose dimensions $m, n$ and rank $k$
2. Generate rank-$k$ matrix $\boldsymbol{L}$
3. Generate sparse matrix $\boldsymbol{S}$
4. Generate additive Gaussian noise matrix $\boldsymbol{N}$ (optional)
5. Compute test data $\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S} + \boldsymbol{N}$
6. Draw random observation set $\Omega$ and reveal measurements $\mathcal{P}_{\Omega}(\boldsymbol{X})$ (optional)
7. Run Robust PCA algorithm and retrieve low-rank estimate $\hat{\boldsymbol{L}}$
8. Evaluate the reconstruction error $e\left(\boldsymbol{L}, \hat{\boldsymbol{L}}\right)$

### Dimensions

Test data is often created as a square matrix, i.e. $m = n$ with the dimensions for Robust PCA varying from a few hundred rows and columns [75] to several thousands [19], thereby slowly approaching the memory limitations of a desktop computer. For Robust Matrix Completion the dimensions may be much higher as only a small percentage of entries needs to be stored. However, processing *Big Data* requires some modifications in the algorithm, see the remarks in Section 4.2.3. While large dimensions are interesting for measuring the computational performance of an algorithm, the recovery ability of an algorithm is commonly only depending on the relation $k/m$ between rank and problem dimensions.

### Low-rank component $\boldsymbol{L}$

The low-rank rank component is commonly generated as the product of two matrices $\boldsymbol{V} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{W} \in \mathbb{R}^{k \times n}$, whose entries are i.i.d. Gaussian. Candès et al. [19] propose the distribution $\mathcal{N}\left(0, \frac{1}{m}\right)$ (or, respectively, $\mathcal{N}\left(0, \frac{1}{n}\right)$), resulting in a low-rank matrix with very small entries. A more popular choice for the entries of $\boldsymbol{V}, \boldsymbol{W}$ is the standard distribution $\mathcal{N}(0, 1)$ [85, 75]. In order to determine which scenarios can be recovered by Robust PCA algorithms, the inner dimension of the product i.e. the *relative rank* $k/m$ is commonly varied over the course of multiple experiments.

### Sparse component $\boldsymbol{S}$

The second decisive performance measure besides the relative rank is the *outlier density* $\rho_S$ of $\boldsymbol{S}$, whose support is generally drawn at random except for special cases in which e.g. some columns are considered to be more corrupted than others [24]. The distribution of the nonzero entries of $\boldsymbol{S}$ varies across the literature. Candès et al. [19] choose a $(-1, 1)$ Bernoulli distribution due to its mild behavior while proving recovery bounds. However, it this is not a very realistic scenario as the outliers surpass the entries of $\boldsymbol{L}$ by several magnitudes, cf. the discussion in [75, 38]. This would allow for a naive reconstruction approach that detects outliers from their magnitude and simply removes them in a pre-processing step. Zhou and Tao [85] and Waters et al. [80] choose a standard distribution, whereas Shen et al. [75] additionally scale the entries of $\boldsymbol{S}$ so that they are in the same magnitude range as the entries in $\boldsymbol{L}$. The maximum density at which successful recovery is possible is a common performance measure in the literature.

### Additive Gaussian noise $\boldsymbol{N}$

While no additive Gaussian noise is considered in the original work of Candès et al. [19], the authors state that it would certainly be interesting to evaluate its influence on the recovery guarantees. The *GoDec* method [85] has explicitly been designed with additive Gaussian noise in mind, and its performance is evaluated with $\boldsymbol{N} \sim \mathcal{N}(0, 0.001)$. Shen et al. [75] choose a noise level relative to the norm of the low-rank component, so that $\|\boldsymbol{N}\|_F^2 \,/\, \|\boldsymbol{L}\|_F^2 = 0.01$.

### Observation set $\mathit{\Omega}$

In the classical Robust PCA problem all entries are revealed and the loss function is evaluated at all positions, i.e. $|\mathit{\Omega}| = mn$. If $|\mathit{\Omega}| < mn$ on the other hand, the problem is a Robust Matrix Completion problem in which missing entries of a low-rank matrix are to be recovered in the presence of sparse outliers. The recovery guarantees derived by [19] are based on the assumption of random sampling, thus the sampling positions $(i, j) \in \mathit{\Omega}$ are always drawn randomly. As a general observation throughout the literature, the required percentage of observed entries for successful subspace reconstruction grows with the relative rank of the data and the outlier density. In [85], different bounds $|\mathit{\Omega}|$ are empirically measured for the *GoDec* method and the performance is compared to other matrix completion algorithms [53], but since $\boldsymbol{S} = 0$ the results cannot be compared with the Robust Matrix Completion setting where $\boldsymbol{S} \neq 0$. He et al. [42] evaluate their Robust Subspace Tracking algorithm *GRASTA* on a static setting with sampling rates from 0.1 to 1.0. Shen et al. [75] investigate similar sampling rates with two distinct sparsity rates and measure the maximum possible rank for the respective scenarios. An even more detailed analysis is presented for *SparCS* [80], where the recovery performance for various scenarios is measured while gradually varying $|\Omega|/mn$ and $|\boldsymbol{S}|/|\Omega|$.

**Reconstruction error**

If the ground truth for the low-rank component is known, then the accuracy of a Robust Low-Rank Approximation algorithm can be evaluated. An intuitive measure for the reconstruction quality is the Root Mean Square Error

$$e_{\mathrm{RMSE}}\left(\boldsymbol{L}, \hat{\boldsymbol{L}}\right) := \frac{\left\|\boldsymbol{L} - \hat{\boldsymbol{L}}\right\|_F}{\sqrt{mn}} , \tag{4.16}$$

which measures the average entry-wise Euclidean distance between the two matrices relative to their dimensions. As the full computation of the low-rank matrix can be costly or even impossible due to memory limitations, Boumal and Absil [8] propose an efficient way of computing the RMSE from the factorized low-rank matrices. The distance can be computed at the cost of a $2k$-dimensional QR decomposition and does not require to carry out the matrix multiplication, which allows to evaluate the performance of Low-Rank Approximation methods in large-scale experiments.

Dividing the Frobenius distance by the Frobenius norm of the ground truth, the relative error

$$e_{\mathrm{rel}}\left(\boldsymbol{L}, \hat{\boldsymbol{L}}\right) := \frac{\left\|\boldsymbol{L} - \hat{\boldsymbol{L}}\right\|_F}{\|\boldsymbol{L}\|_F} \tag{4.17}$$

is obtained. The advantage over $e_{\mathrm{RMSE}}(\cdot)$ is the tolerance against variations in the scaling of the entries of $\boldsymbol{L}$. A third measure only involving the basis and not the coordinates is the maximum subspace angle

$$e_{\mathrm{sub}}\left(\boldsymbol{U}, \hat{\boldsymbol{U}}\right) := \arcsin\left(\sigma_{\max}\left(\boldsymbol{U} - \hat{\boldsymbol{U}}\hat{\boldsymbol{U}}^{\top}\boldsymbol{U}\right)\right) \tag{4.18}$$

proposed by Björck and Golub [7]. As both matrices have orthogonal columns, the singular values cannot exceed a value of 1, which corresponds to a maximum angle of 90 degrees. Whenever the two bases coincide perfectly, on the other hand, all singular values and thus also the subspace angle are zero.

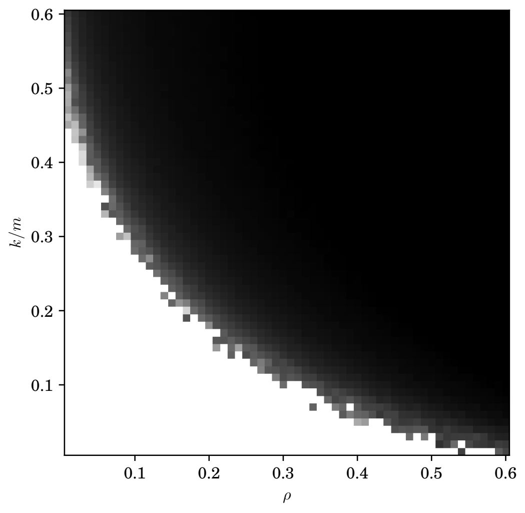
### 4.3.1. Phase transitions in rank and sparsity

A common experiment for measuring and comparing the performance of Robust PCA algorithms are phase transitions in rank and sparsity. The goal of the experiment is to determine under which conditions an algorithm is able to recover a matrix of varying rank in the presence of outliers of varying density. In other words, it is determined how far the statistics of a data set may diverge from the ideal low-rank-and-sparse data model before the recovery performance of the inspected algorithm breaks down.

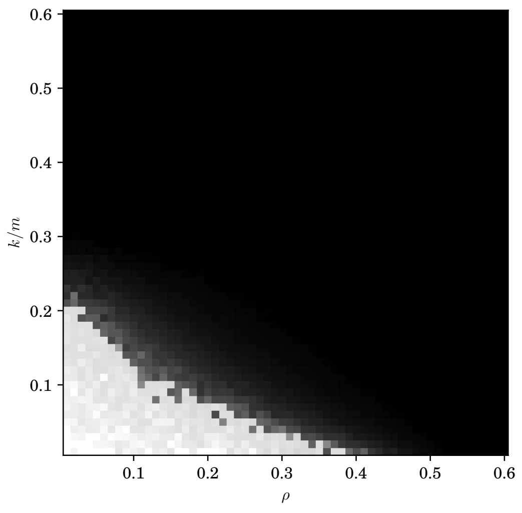
Fur this purpose, test data is generated according to the data model proposed by Shen et al. [75], which is chosen due to the balanced magnitudes of the low-rank and the sparse component. The dimensions are chosen as $m = n = 200$. Both the relative rank $k/m$ and the outlier density $\rho$ are varied in the range $[0.01, 0.6]$. All entries are revealed and no additive Gaussian noise is considered. Like in the original evaluation [75], a trial is considered successful if the relative reconstruction error is in the range of $10^{-8}$. To achieve this requirement, the parameter settings of the proposed *GRPCA* method in this experiment are chosen in favor of reconstruction accuracy. The cost function parameters are $p = 0.1$ and $\mu_{\text{end}} = 10^{-16}$, while $\mu$ is initialized with $\mu_{\text{start}} = 0.1$. Whenever the relative progress falls below $\delta_{\min} = 0.01$, $\mu$ is reduced by a factor of $c_\mu = 0.5$. The line search in the CG does not use subsampling and the parameters are conservatively chosen as $\rho = 0.9$ and $\delta_{\min} = 10^{-6}$ with a maximum of 10 CG iterations. As the rank estimation problem is outside the scope of this work, it is assumed that the true rank is known a priori. The algorithm is initialized with a random orthogonal frame $\boldsymbol{U}_0$ and with $Y_0 = 0$.

Figure 4.3 shows the phase transitions of the proposed method (4.3f) in comparison to competing state of the art methods, whose *MATLAB* implementations have been obtained from the respective authors' web pages[1]. Unsurprisingly, the phase transitions for *LMaFit*

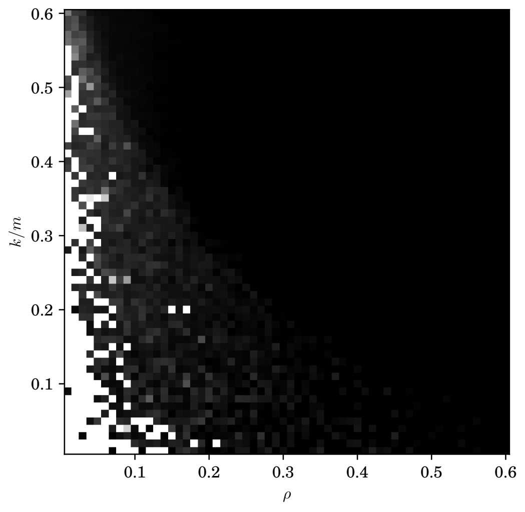
[1] `http://lmafit.blogs.rice.edu` (*LMaFit* & *IALM*)
`https://sites.google.com/site/godecomposition/code` (*GoDec*)
`https://sites.google.com/site/hejunzz/grasta` (*GRASTA*)
`https://people.stanford.edu/lcambier/rmc` (*RMC*)



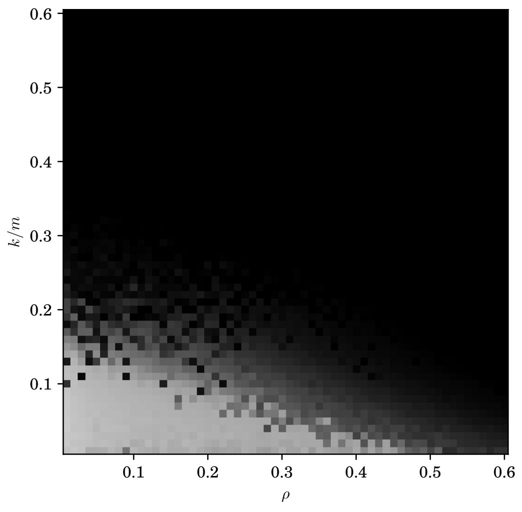
**(a)** Phase transitions for *LMaFit* [75]



**(b)** *IALM* [60]

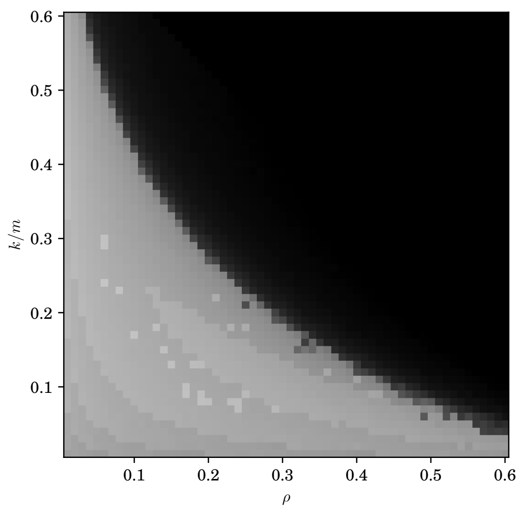


**(c)** *GoDec* [85]

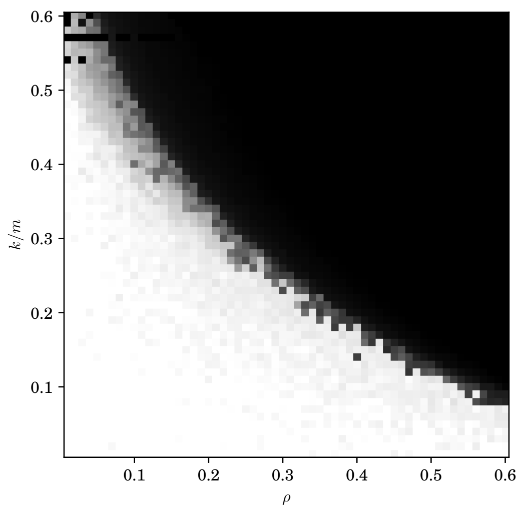


**(d)** *GRASTA* [42]

**Figure 4.3.:** Phase transitions in rank and sparsity for *GRPCA* in comparison to state-of-the-art Robust PCA algorithms. White: Relative Reconstruction Error below $10^{-8}$, Black: Relative Reconstruction Error above $10^{-1}$



(e) *RMC*



(f) *GRPCA*

(Figure 4.3a) and *IALM* (Figure 4.3b) are in line with the original comparison by Shen et al. [75], because the exactly same data model is used for this evaluation. As mentioned by the authors, the factorization approach of *LMaFit* leads to a wider range of scenarios that can be reconstructed than the nuclear norm approach of the *IALM* method. In defense of *IALM*, it needs to be stressed that the algorithm is practically parameter-free and, in contrast to all other competitors in this evaluation, does not require an a priori estimate of the rank. The *GoDec* method is known to be a fast and well-scaling algorithm due to the use of random projections. However, in addition to the rank estimate the algorithm requires an estimate of the outlier cardinality, which is why the actual cardinality is fed into the algorithm. To further maximize the performance, the stopping criteria are tweaked for higher accuracy, i.e. the maximum number of iterations is doubled and the `error_bound` parameter is lowered from $10^{-3}$ to $10^{-8}$. Despite these efforts, the algorithm fails to reach the required relative error, especially when the outlier density is large. Figure 4.3d illustrates the phase transitions for *GRASTA*, which covers about the same range of scenarios as *IALM* with a slightly reduced maximum accuracy. It needs to be mentioned that the algorithm is actually designed for subspace tracking, which renders a comparison against explicit batch processing methods a bit unfair. An algorithm that is very similar to the proposed method

| Method (Parameter settings) | Processing time (seconds) |
|---|---|
| *LMaFit* | 0.05 |
| *IALM* | 0.27 |
| *GoDec* (default) | 0.27 |
| *GoDec* (high accuracy) | 0.40 |
| *RMC* (default) | 1.46 |
| ***GRPCA* (default)** | **1.91** |
| *GRASTA* | 2.30 |
| *RMC* (high accuracy) | 3.78 |
| ***GRPCA* (high accuracy)** | **6.69** |

**Table 4.1.:** Run time comparison for state-of-the-art Robust PCA algorithms. Noise-free test data generated according the model of Shen et al. [75] with dimensions $m = n = 200$, rank $k = 20$ and outlier density $\rho = 0.1$

is the Robust Matrix Completion (*RMC*) method by Cambier and Absil [18]. The cost function is a smoothed variant of the $\ell_1$ norm, whose smoothing parameter is shrunk in the course of the optimization. While the authors propose a shrinkage rate of 0.05 and a maximum number of seven iterations until termination, the parameters have been adjusted in order to increase the accuracy and to make it even more comparable to *GRPCA*. Precisely, the shrinkage rate is set to 0.5 as well and the algorithm is run until complete convergence, i.e. until the improvement in minimizing the cost function is less than $10^{-16}$. *RMC* covers a greater range of scenarios than other competing methods, which indicates that an approach with a smoothed cost function and a shrinkage scheme for the smoothing parameter leads to increased performance compared to other approaches that employ an $\ell_1$ norm without smoothing. Yet, comparing the results to the phase transitions for *GRPCA* (Figure 4.3f) it becomes obvious that the borders of Robust Subspace recovery can be pushed even further with the proposed smoothed $\ell_p$-norm loss function with $p = 0.1$ as a sparsity measure. Yet, this precision comes at the cost of increased computational effort, as a comparison of the particular runtimes of the methods in Table 4.1 reveals for the scenario with both the relative rank and the outlier density set to $k/m = \rho = 0.1$. The fastest method is *LMaFit* with a mere 0.05 seconds of processing time. The next fastest methods are *IALM* (0.27 seconds) *GoDec*, which takes 0.4 seconds (0.27 seconds with the authors' parameters). The algorithms that use manifold optimization require some more time: the non-optimized *MATLAB* prototype implementation of *GRASTA* requires 2.3 seconds and *RMC* with the parameter set tweaked towards increased reconstruction accuracy has a computation time of 3.78 seconds (1.46 seconds with the original parameter set). Finally, *GRPCA* finishes last with 6.69 seconds of processing time. To conclude, the proposed method surpasses the competing manifold optimization methods regarding the reconstruction accuracy and covers the widest range of scenarios of all compared methods. For practical use, however, it may be reasonable to sacrifice some of the accuracy in favor of a faster processing time.

### 4.3.2. Runtime and optimum choice of the shrinkage parameter $\mu$

The runtime of *GRPCA* depends on the range of the cost function parameter $\mu$ and its shrinkage rate. The following experiment sheds some more light on the influence of $\mu$ on the minimization process and justifies the heuristic (4.7) proposed in Section 4.1 of selecting the final value $\mu_{\text{end}}$ according to the estimated noise level. A low-rank-and-sparse data set is created as before, with $k = 20$ and $\rho = 0.1$. This time, however, a Gaussian noise matrix $\boldsymbol{N}$ is added whose entries are i.i.d. according to $\mathcal{N}\left(0, \sigma^2\right)$ with a standard deviation of $\sigma = 0.05$. During the normalization procedure (cf. Section 4.2.2) the data and with it the noise is scaled down by a factor $\alpha$, so that the standard deviation of the noise after scaling is $\tilde{\sigma} := \sigma/\alpha$. As discussed in Section 4.1, the optimum value for $\mu$ is therefore computed as $\mu_{\text{opt}} = 9\,(1-p)\,\tilde{\sigma}^2$. For $m = n = 200$ and $k = 20$, the scaling $\alpha \approx 14$ is obtained. With the choice $p = 0.1$, the optimum terminal value of the smoothing parameter turns out as $\mu_{\text{opt}} = 9\,(1-0.1)\left(\frac{0.05}{14}\right)^2 \approx 10^{-4}$.

In order to determine whether this is actually the optimum choice for the smoothing parameter, the algorithm is run several times with varying values for $\mu_{\text{end}}$, while the progress in minimizing the relative reconstruction error over the number of iterations is observed. Figure 4.4 illustrates the progress. While the initial value $\mu_{\text{start}} = 0.1$ allows for a quick and rough approximation, the value of $\mu$ is halved whenever the relative progress falls below $\delta_{\min} = 10^{-4}$. As long as $\mu > \mu_{\text{opt}}$, the shrinkage causes the algorithm to reduce the relative reconstruction error further. However, once the optimum value for $\mu$ has been reached, a further decrease is not beneficial. Even worse, it can possibly even lead to a suboptimal approximation result.

### 4.3.3. Evaluation of cost function parameter $p$

One major objective of this experimental section is to evaluate the effectiveness of employing an $\ell_p$-norm loss function with $0 < p < 1$ on the Robust PCA problem. Therefore, the above

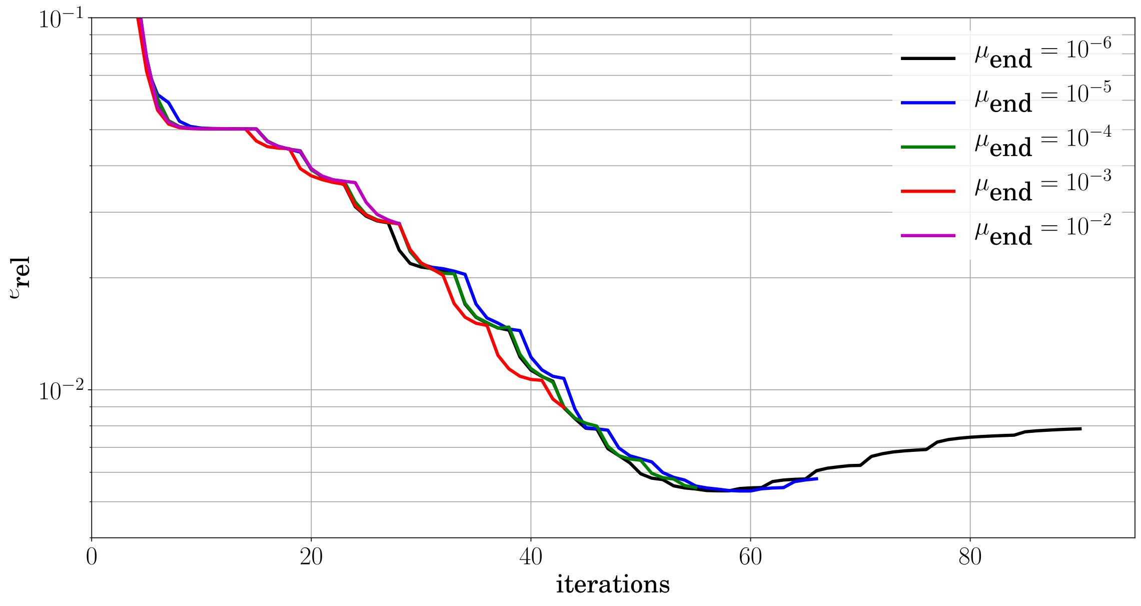
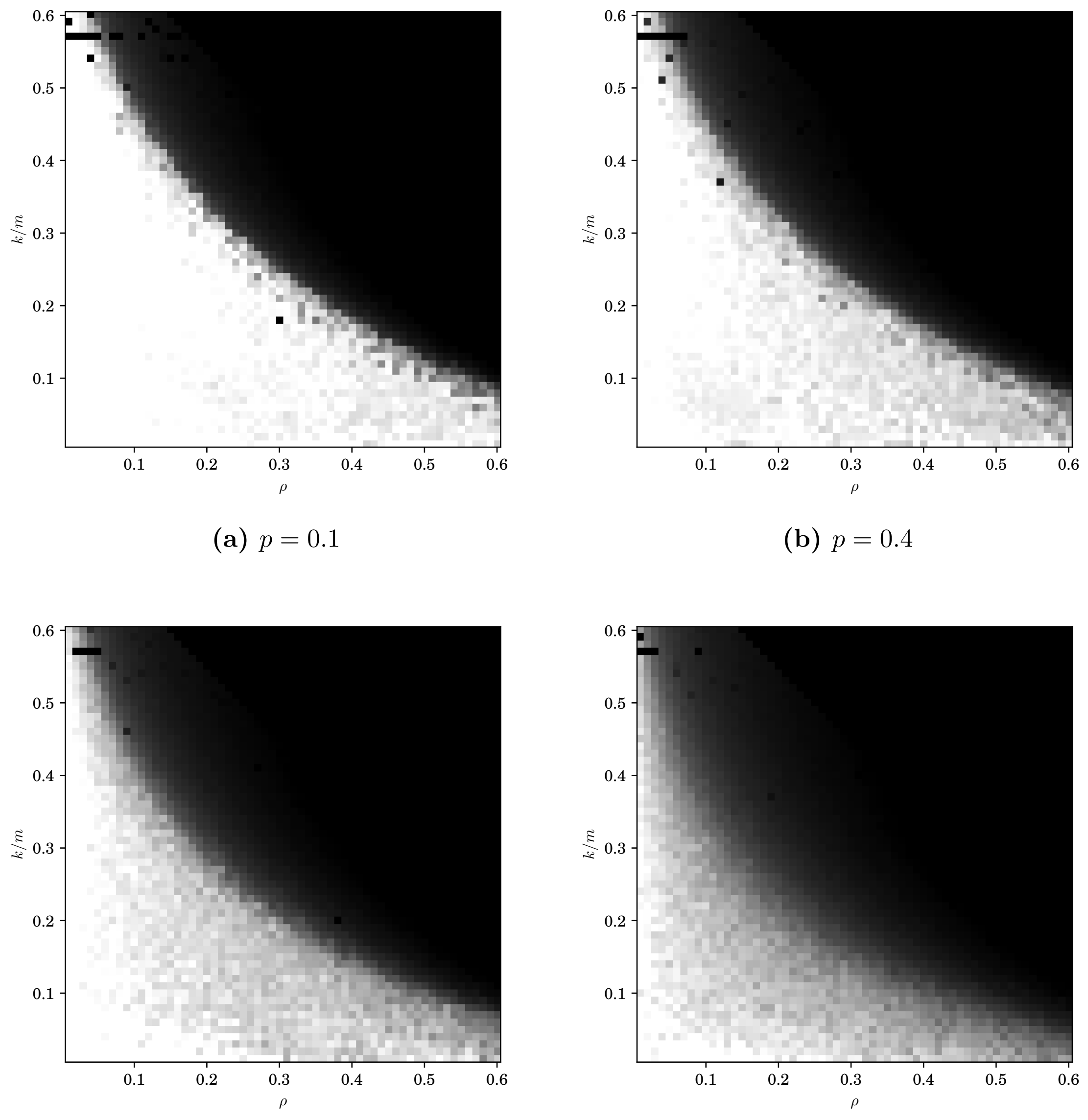


**Figure 4.4.:** Progress in minimizing the Relative Reconstruction Error over the number of iterations for varying minimum values $\mu_{\mathrm{end}}$ of the smoothing parameter $\mu$

conducted experiment is reproduced with different settings for the cost function parameter $p$, ranging from 0.1 to a maximum value of 1.0. This time, the focus is on efficient processing at a reasonably good accuracy, which means that a trial is considered successful when the relative reconstruction error is below $10^{-4}$. The following parameters have been chosen to achieve a good trade-off between reconstruction accuracy and computational efficiency: During the step size estimation, the cost function is subsampled with $|\Psi| = 10^4$. Whenever the relative progress in reducing the cost function is less than 0.01, $\mu$ is reduced by a factor of 0.2, down to a minimum value of $10^{-8}$. The shrinkage parameter $\rho$ for the respective CG methods is initialized with 0.1 and subsequently raised to a final value of 0.9, so that the initial progress is quick but coarse and high precision is reached at termination. Also, the CG is terminated either after 10 iterations or if the relative progress is less than $10^{-4}$. Figure 4.5 depicts the respective phase transitions for various values of $p$. Clearly, the lower $p$ is chosen, the smaller the reconstruction error in challenging scenarios, which shows the effectiveness of the proposed cost function. For $p = 1.0$, the smoothed $\ell_p$-norm loss function resembles the cost function of the *RMC* algorithm [18]. As a consequence, also the phase



**(a)** $p = 0.1$ **(b)** $p = 0.4$

**(c)** $p = 0.7$ **(d)** $p = 1.0$

**Figure 4.5.:** Phase transitions in rank and sparsity for the proposed *GRPCA* method for varying cost function parameters $p$. White: Relative Reconstruction Error below $10^{-4}$, Black: Relative Reconstruction Error above $10^{-1}$

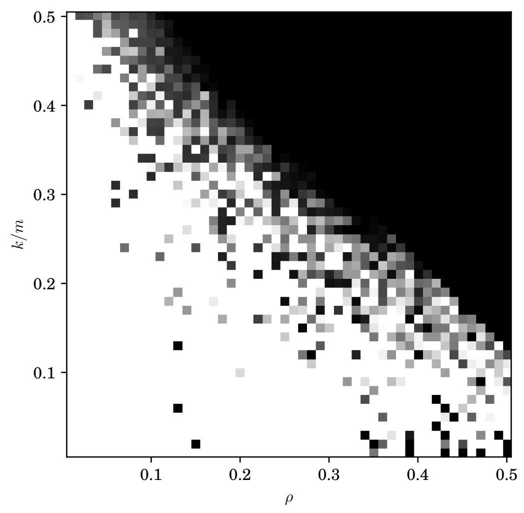
transitions (Figures 4.3e and 4.5d, respectively) are very similar, considering the different reconstruction threshold.

While the changed parameter setting slightly reduces the reconstruction accuracy, it allows for much faster computation. For the above mentioned scenario ($k/m = \rho = 0.1$), *GRPCA* now requires 1.91 seconds at $p = 0.1$ and 1.70 seconds at $p = 1.0$. As can be seen from Table 4.1, this lands the proposed method in the range of the competing methods *GRASTA* and *RMC* with regard to computational complexity, while allowing for successful reconstruction in a broader range of scenarios.

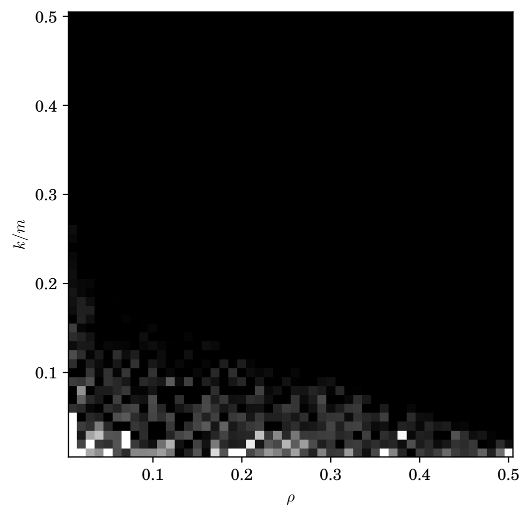
The benefit of the proposed cost function becomes even more obvious when outliers of excessively large magnitudes are considered. Candès et al. [19] consider such data in their evaluation of the *IALM* method, whose phase transitions have been reproduced and are displayed in Figure 4.6c. Shen et al. [75] report that *LMaFit* fails to separate the low-rank and the sparse component in such scenarios, and similar behavior has also been observed for the *RMC* method [18]. As before, this relates directly to the behavior of the proposed method for $p = 1.0$, whose poor reconstruction performance is illustrated in Figure 4.6b. But in the case where $p = 0.1$, a much wider range of scenarios can be recovered again, and the proposed method surpasses the coverage of the *IALM* method.

### 4.3.4. Robust Matrix Completion

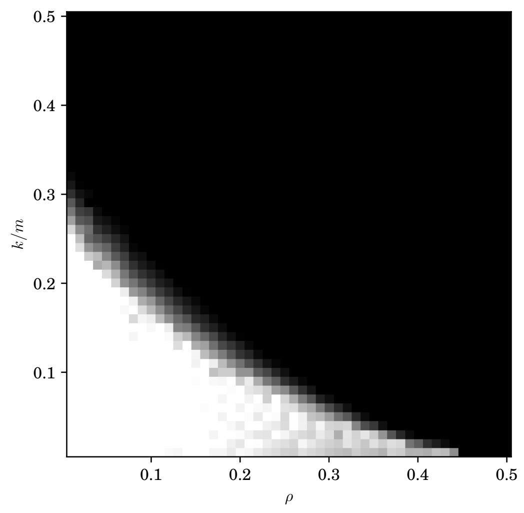
In order to evaluate the reconstruction performance from incompletely observed data, the previously conducted experiment is repeated, this time revealing only a certain percentage of entries. Figure 4.7 shows the phase transitions for the reconstruction from 80% down to 20% of the entries. In general, the range of scenarios decays with the observability of the data. Furthermore, the reduced number of observed data points limits the dimensionality of the subspace more than the robustness against outliers. For scenarios where the rank is less than 10% of the dimension ($k < 0.1m$), the reconstruction is still successful in the presence of up to 40% outlier density, even if only 20% of the data is revealed. Comparing the proposed *GRPCA* method with the *RMC* algorithm, however, it seems that the advantage of using the smoothed $\ell_p$-norm cost function seems to vanish with lower sampling rates. While the range of recovered scenarios is wider for 80% revealed entries (Figure 4.7a versus Figure 4.8a), the coverage of both methods is very similar at 20% (Figure 4.7a versus Figure 4.8b).



**(a)** *GRPCA*, $p = 0.1$

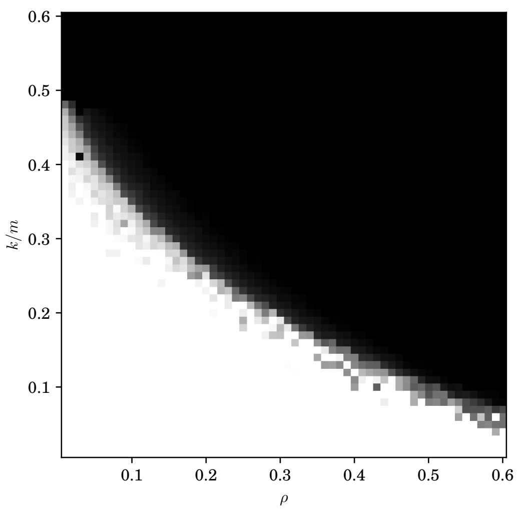


**(b)** *GRPCA*, $p = 1.0$

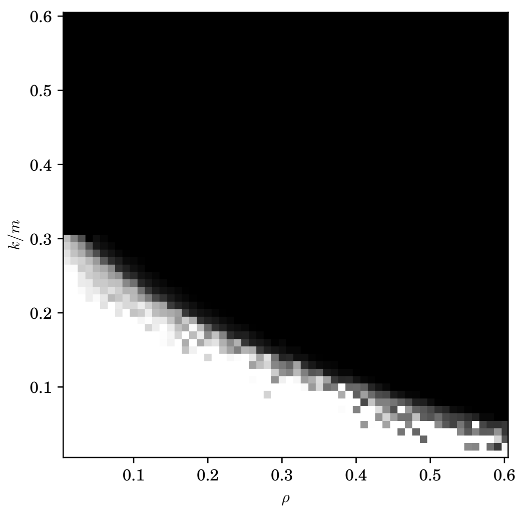


**(c)** *IALM*

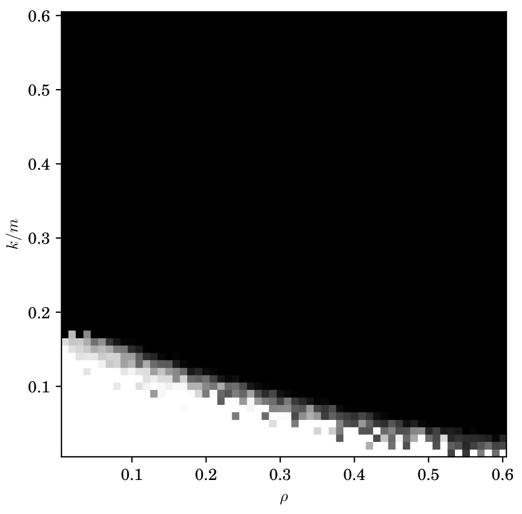
**Figure 4.6.:** Phase transitions in rank and sparsity for the proposed *GRPCA* method for varying cost function parameters $p$ on a data set with outliers of large magnitude. White: Relative Reconstruction Error below $10^{-3}$, Black: Relative Reconstruction Error above $10^{-1}$



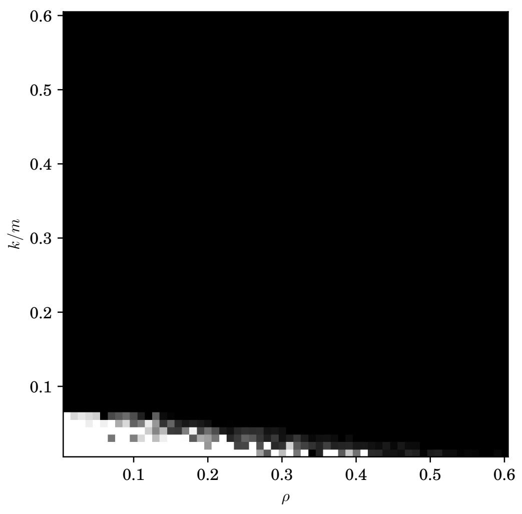
**(a)** $\frac{|\Omega|}{mn} = 0.8$



**(b)** $\frac{|\Omega|}{mn} = 0.6$

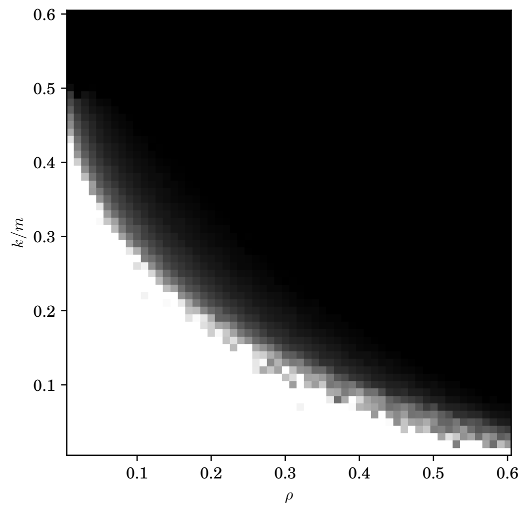


**(c)** $\frac{|\Omega|}{mn} = 0.4$

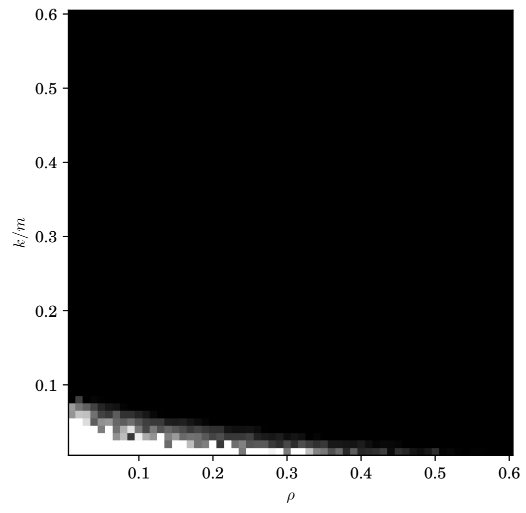


**(d)** $\frac{|\Omega|}{mn} = 0.2$

**Figure 4.7.:** Phase transitions in rank and sparsity for the proposed *GRPCA* method for varying fractions of observed entries. White: Relative Reconstruction Error below $10^{-4}$, Black: Relative Reconstruction Error above $10^{-1}$



**(a)** $\frac{|\Omega|}{mn} = 0.8$



**(b)** $\frac{|\Omega|}{mn} = 0.2$

**Figure 4.8.:** Phase transitions in rank and sparsity for *RMC* for varying fraction of observed entries. White: Relative Reconstruction Error below $10^{-4}$, Black: Relative Reconstruction Error above $10^{-1}$

### 4.3.5. Computational complexity and memory requirements

Robust PCA methods that minimize the nuclear norm (such as e.g. *IALM*) are mostly based on the Singular Value Decomposition. Assuming without loss of generality that the input $\boldsymbol{X}$ is a squared $m \times m$ matrix, the computation of an SVD has the complexity of $\mathcal{O}(m^3)$. Although some approaches manage to reduce the complexity by computing partial SVDs to $\mathcal{O}(m^2k)$ [41], a way towards efficient algorithms should circumvent SVDs of large matrices altogether [75]. The main cost of a first-order optimization method like the proposed alternating CG method lies in the evaluation of the cost function and the computation of the gradient, which both require the computation of the matrix product $\boldsymbol{UY}$ with complexity $\mathcal{O}(m^2r)$. Notice that the computation of the Grassmannian geodesics (Equation (3.12)) requires an SVD of the search direction, but since the dimensions of the direction are $m \times k$, the order is $\mathcal{O}(k^2m)$, which for $k \ll m$ and large $m$ is much smaller than $\mathcal{O}(m^2k)$.

Considering that the step size is determined with a backtracking line search method (Algorithm 3.2), the optimization involves many evaluations of the cost function. As motivated

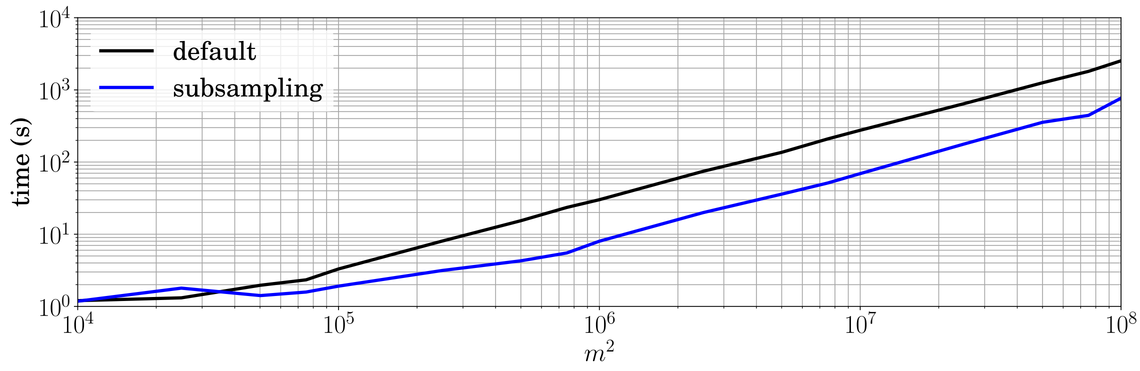


**Figure 4.9.:** Run time comparison for *GRPCA* with full cost function evaluation and subsampled conjugate gradient with $|\Psi| = 10^4$, respectively, for squared input of varying dimension $m$

in Section 3.4, restricting the computation to a subpartition of the coordinates is an effective way to make the line-search more efficient. Figure 4.9 shows the computation times for GRPCA for quadratic input with dimensions ranging from $m = 100$ up to $m = 10^4$. In the *default* setting, the cost function is evaluated on all coordinates, while the *subsampling* approach approximates the cost function on a subpartition of the coordinates with $|\Psi| = 10^4$. For small data sets, the subsampling advantage is small or non-existent, but for larger data sets the computation time is reduced significantly, as the smoothed $\ell_p$-norm cost function needs to be evaluated only on a small fraction of the coordinates during the line search. Yet, it also becomes obvious that for large data sets not the evaluation of the cost function drives the complexity, but the computation of the matrix product $\boldsymbol{UY}$, which is required to compute the actual decrease in the cost function and the gradient for the next iteration. Thus, as mentioned in Section 4.2.3, it can be cheaper to employ a selective matrix-matrix product (*smmprod*) routine to compute $\mathcal{P}_\Psi(\boldsymbol{UY})$ for the line search and to compute $\mathcal{P}_\Omega(\boldsymbol{UY})$ without actually carrying out the full product, whenever the data is incomplete observed with $|\Omega| \ll m^2$. Figure 4.10 showcases the efficiency of this approach for large matrix dimensions, $|\Psi| = 10^4$ and $|\Omega| = 0.1m^2$. To test the accuracy of the approximation, the RMSE is computed according to Equation (4.16). In all experiments an RMSE in the range of $[10^{-5}, 10^{-4}]$ has been observed, regardless of the problem dimension and with no observable difference between subsampling or fully computing the cost function during the line search. Empirically, it has been found that a subsampling set with $|\Psi| = 10^4$ is sufficiently large to approximate the sparsity of cost function to the required precision in

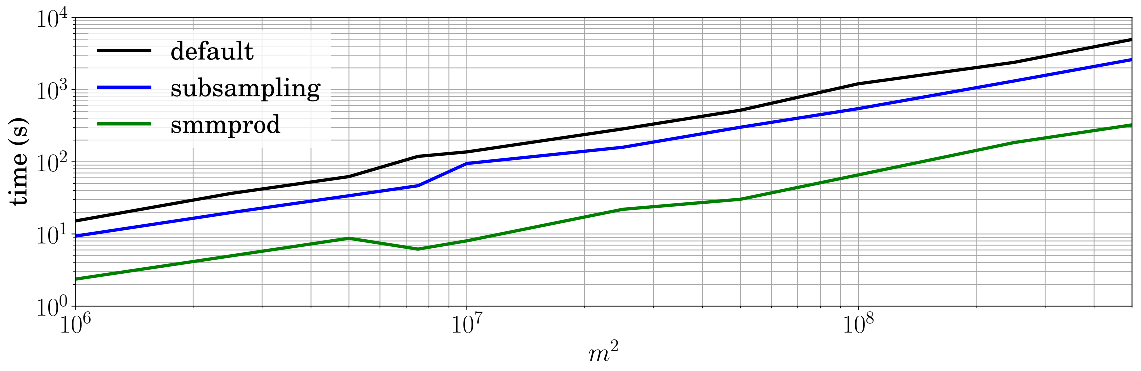


**Figure 4.10.:** Run time comparison for *GRPCA* with full cost function evaluation, subsampled conjugate gradient with $|\Psi| = 10^4$ and additional selective matrix-matrix product, respectively, for squared input of varying dimension $m$, sampled at a rate of $\frac{|\Omega|}{m^2} = 0.1$

all conducted experiments. The algorithm benefits significantly from an efficient implementation of the cost function, the gradient and the selective matrix-matrix product in custom extensions written in *C*. The computational complexity scales approximately linear with the cardinality of the sampling set $\Omega$, which has also been observed in the related approach by Cambier and Absil [18]. Regarding memory requirements, the factorization approach is known to be much more efficient than nuclear norm based approaches, as only the factors $\boldsymbol{U}$ and $\boldsymbol{Y}$ need to be kept in memory. Subsampling the data set allows to even tackle problems with the full $\boldsymbol{L}$ surpassing the available memory in size, as only the actually observed coordinates need to be stored and processed. All this lets the presented approach scale well to large data sets and makes it an effective approach for processing Big Data.

## 4.4. Application on Video Segmentation

In order to illustrate the performance of the proposed *GRPCA* method on real world data, an experiment on Video Segmentation is conducted. The segmentation problem aims at dividing a video recording of a scene into background and foreground objects. Unfortunately, there is no universally accepted definition of what a foreground object in a video is, mostly because human vision is depending on a lot of contextual information. An intuitive definition, however, is to characterize the background as persistent elements of a scene, whose information content does not increase over the course of a prolonged observation, whereas the foreground elements are the parts of a scene that continuously bring in new information to the scene and thus raise the focus of attention of a human observer. While this distinction can very easily made by humans, it is a highly challenging task for a machine. The reason for this is that an imaging sensor of a camera is only sensitive to changes in the intensity level of pixels on an image sensor, but not every intensity change makes a pixel belong to a foreground object in the scene.

The simplest background model is to subsume all static parts of a video in a single background image. This image can then be subtracted from the subsequent video frames, which is why this kind of video segmentation approach is often referred to as *background subtraction*. While this simple solution achieves surprisingly good results in static environments, applying this method to scenes with more challenging backgrounds results in a lot of false positives (i.e. background parts erroneously detected as foreground). Thus, a more elaborate background model needs to be found that is capable of modeling dynamics in the background, such as illumination changes, repetitive motion in objects or jitter resulting from a shaking camera. What unites these phenomena are the limited complexity of the underlying dynamics and a certain temporal coherence and persistence. The foreground objects on the other hand typically appear sparse in both space and time, i.e. they occupy only a fraction of the scene and are present in this location only for a short period of time. The key idea is thus to divide a video sequence into a low-dimensional component that can deal with a dynamic background of limited complexity and into a sparse component containing the foreground objects in the scene.

### 4.4.1. Principles of video segmentation using Robust PCA

As part of a system for human activity recognition, Oliver et al. [68] introduced video segmentation via PCA. As Bouwmans [10] report, such approaches show great advantages over competing methods whenever backgrounds are dynamic and include illumination changes. To visualize the process and to furthermore motivate the use of Robust PCA, a toy example is considered that represents a moving foreground object in front of a background with varying brightness. Consider a video sequence containing of $n$ frames with $m$ pixels each, then a data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ is obtained by vectorizing the frames and concatenating them. Figure 4.11 visualizes this process for a schematized video scene that consists of only three frames of nine pixels each. Taking a closer look at the image sequence it can be observed

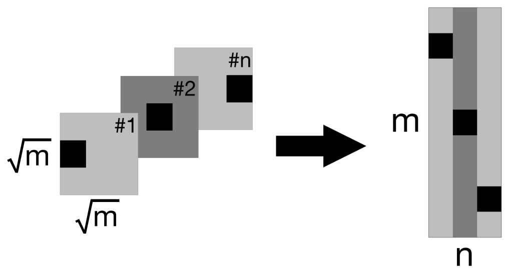
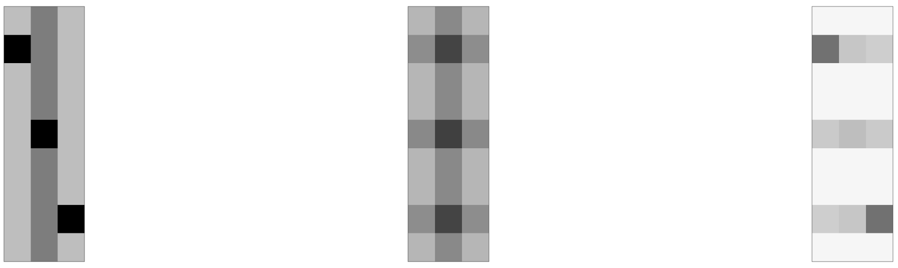


**Figure 4.11.:** Vectorized video sequence containing $n$ frames with $m$ pixels each ($m = 9, n = 3$)

that one black pixel is present in all three frames but at different positions. The remaining pixels in each frame are all gray and have the same intensity in frame #1 and #3 but take on a darker tone in frame #2. So technically, the intensity of every pixel changes from frame to frame. Yet, the intuitive subjective perception of the video sequence is that of a moving black dot in front of a uniform and grayish background, which is sometimes brighter and sometimes darker. The data matrix obtained by vectorizing and concatenating the frames is a $9 \times 3$ matrix and has full rank $k = n = 3$. Without the black pixels, however, the first and third columns are identical and the second is just a scaled version, which would constitute a matrix of rank 1. The goal of segmenting the video sequence is therefore to recover the one-dimensional background $\boldsymbol{L}$ and to subtract it from $\boldsymbol{X}$, so that $\boldsymbol{X} - \boldsymbol{L}$ is zero everywhere except for the positions of the black foreground pixels. As discussed previously, the standard tool for computing low-rank approximations is PCA, and Figure 4.12 shows the decomposition of the video sequence. Precisely, the $\ell_2$-rank-one approximation obtained by PCA is shown in Figure 4.12b, while the residual is visualized in Figure 4.12c. It needs to be mentioned that only the magnitude of the residual is relevant because a dark foreground element on a bright background is equally relevant as a bright foreground element on a dark background. As targeted, the first and third column of the reconstructed background are identical and the second column is a scaled version. However, due to the quadratic loss function the black dots do not vanish completely, as PCA tries to distribute the residual error evenly across all entries in the matrix. As a consequence, artifacts appear at other positions, which becomes even more clear if the background and foreground matrices are unraveled again into a sequence of frames, as displayed in Figure 4.13. The quadratic error causes the foreground objects to leak into the background model, resulting in so-called ghost images, i.e. foreground objects appear in the background at times and positions when



**(a)** Input $\boldsymbol{X}$ **(b)** Rank-1 approx. $\boldsymbol{L}_{PCA}$ **(c)** Residual $|\boldsymbol{X} - \boldsymbol{L}_{PCA}|$

**Figure 4.12.:** Rank-one approximation of a simplified video sequence with common PCA

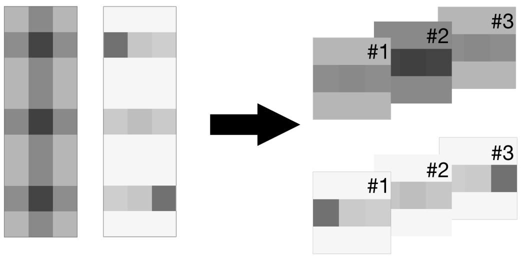
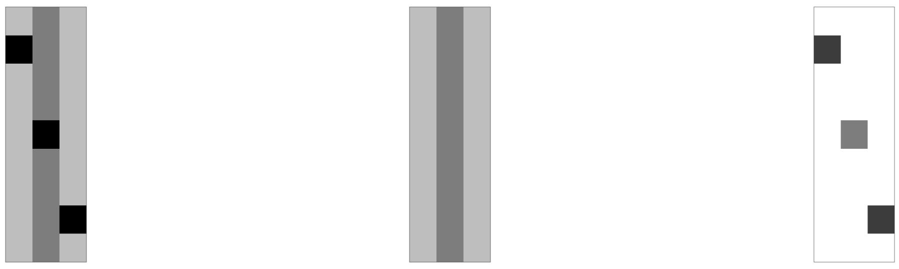


**Figure 4.13.:** Frame-wise reconstruction of the video segmentation results obtain from common PCA. left/above: background, right/below: foreground



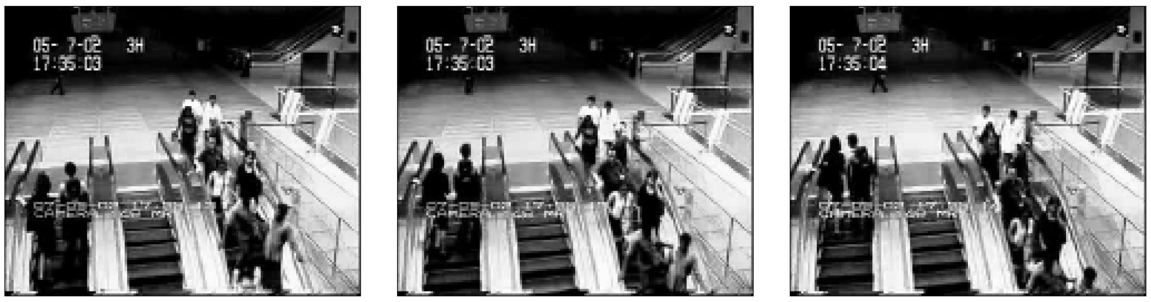
**(a)** Input $\boldsymbol{X}$ **(b)** Rank-1 approx. $\boldsymbol{L}_{RPCA}$ **(c)** Residual $|\boldsymbol{X} - \boldsymbol{L}_{RPCA}|$

**Figure 4.14.:** Rank-one approximation of a simplified video sequence with Robust PCA

they are not present in the original sequence. As a consequence, these effects also appear in the residual, which is nonzero at all positions. Determining an appropriate threshold to extract the position of the foreground objects is nearly impossible, making common PCA a suboptimal tool for foreground-background segmentation.

The key to success is to employ Robust Low-Rank Approximation techniques that are tailored for the low-rank-and-sparse data model (1.6). Figure 4.14 shows the decomposition of $\boldsymbol{X}$ with Robust PCA into a rank-1 component containing the background and a sparse component with the estimated foreground. As the residual error between the video sequence and the low-rank approximation is enforced to be sparse, the foreground pixels do not leak into the background. Thus no ghost images appear and the background is reconstructed perfectly. As the entries in the sparse matrix are created by subtracting the background from the input sequence, their intensity depends on the intensity of the obscured background at that position. Generating a foreground sequence is thus a two-stage process, which involves thresholding the residual to determine the support, followed by applying a binary mask on the original input to single out the foreground elements. As an inherent drawback of the method, foreground elements cannot be recovered if they have the same (or very similar) intensity as the background model. But this is not specific to PCA or Robust PCA but to any background subtraction approach.

As a conclusion from this toy example, Robust PCA offers substantial advantages over common ($\ell_2$) PCA when applied to video segmentation. This makes video segmentation an ideal application to showcase the performance of Robust PCA methods.



**(a)** Frame # 1806 **(b)** Frame # 1813 **(c)** Frame # 1820

**Figure 4.15.:** Frames # 1806, 1814 and 1820 of the *escalator* sequence.

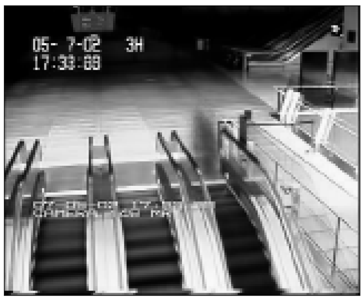
### 4.4.2. Real-world example

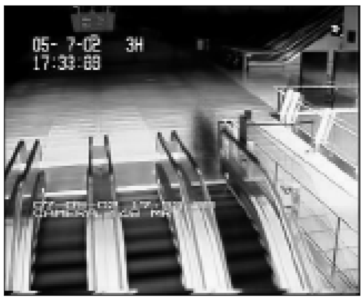
A data set that is often used to visualize the video segmentation performance of Robust PCA algorithms is the data set by Li et al. [58]. The data set contains RGB image sequences from surveillance cameras in different environments. The majority of the data set, such as the popular *hall* sequence (appearing in the evaluation of *IALM* [19], *LMaFit* [75], *GoDec* [85] and *GRASTA* [42]), considers backgrounds that are static except for occasional illumination changes. Pleasing results have been obtained for these scenarios, but it can be argued that modeling static backgrounds is rather simple and does not challenge the full potential of Robust PCA approaches on video segmentation, even if some attempts have been undertaken to increase the difficulty of the task (cf. the artificial panning experiment in [42]). An arguably more challenging sequence within the data set is the *escalator* sequence, of which three frames are depicted in Figure 4.15. The sequence contains the surveillance capture of a somewhat crowded indoor environment with three continuously running escalators. Intuitively, the goal of the video segmentation is therefore to capture the repetitive motion of the escalators while separating out the people riding them and walking in and out of the scene. The following evaluation aims to evaluate the segmentation performance of the proposed *GRPCA* method for varying values of the cost function parameter $p$ and to compare the best configuration to competing Robust PCA methods. The video sequence is converted to grayscale images and is processed at full resolution ($160 \times 130$ pixels) and full length (3417 frames), resulting in an input of dimension $20,800 \times 3417$. To visualize the results of the comparison, the frames #1806, #1813 and #1820 have been picked. On the one hand, the temporal distance between the frames relates to the observed period length of about 21 frames for the escalator movement. On the other hand, this observation instant captures the largest human crowd in the sequence, as the original frames of the sequence in Figure 4.15 indicate. Empirically, the rank of the approximation has been chosen as $k = 4$ and a detection threshold $\tau$ of 20 intensity levels has been selected. The smoothing parameter $\mu$ is computed according to the heuristic in Equation (4.7) using the value for the detection threshold $\tau$. As the heuristic would lead to a value of $\mu = 0$ in the degenerate case $p = 1.0$, $\mu$ is instead computed for a value 0.9.

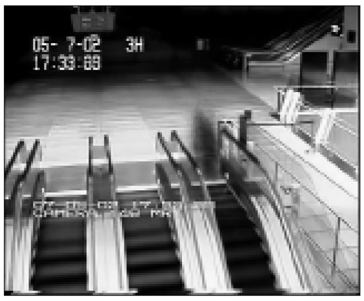
The background frames in Figure 4.16 illustrate that the *GRPCA* method is successful at capturing the dynamics of the escalator. However, for large values of $p$ the large crowd on the right escalator produces ghost images in the background model, whereas ghosting can be suppressed for lower values of $p$ without deteriorating the dynamic background model for the escalators.

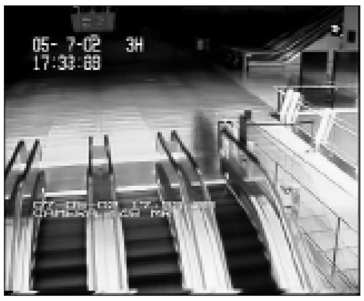
Moving along to the comparison of the competing methods in Figure 4.17, the *IALM* method (second row) captures the dynamics very well but also produces the highest level of ghosting. This indicates that the heuristic of Candès et al. [19] for the selection of the parameter $\lambda$ weighing between the low-rank of the approximation and the sparsity of the residual may be suboptimal for suppressing ghosting artifacts. The third row shows the result for *LMaFit*, which is somewhat similar to the result of *GRPCA* for $p = 1.0$ with some ghosting. Finally, the *GoDec* manages to suppress the ghosting artifacts well, likely profiting from the additional bound on the cardinality of the residual (i.e. an estimate of $\|S\|_0$), which is empirically selected as 3% of the overall number of pixels.

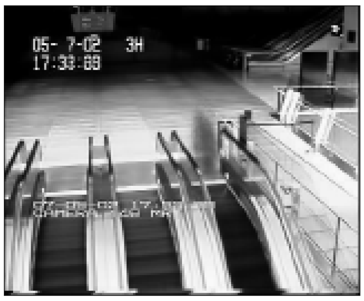
For a last subjective evaluation, the extracted foreground elements for frame # 1813 are visualized in Figure 4.18. While *GoDec* directly returns an estimate of the sparse residual (due to the additive Gaussian noise term in the data model), the residual for the other methods is obtained by subtracting the background estimates from the input. Following the procedure outlined in Section 4.4.1, the residual $\boldsymbol{X} - \boldsymbol{L}$ is thresholded at $\tau = 20$ intensity levels to extract the sparse foreground elements and the resulting binary mask is post-processed with a $3 \times 3$ median filter. The final foreground frame is then obtained by masking the input frame to extract foreground pixels. Comparing the results in Figure 4.18 visually, one observes that *GRPCA* captures the largest number of foreground pixels. This includes some details of the escalator steps, but also the reflection of some people on the glass wall next to the escalator. While there should be no discussion that the former is a false positive, the second would likely also count as a false positive, even though the reflection of

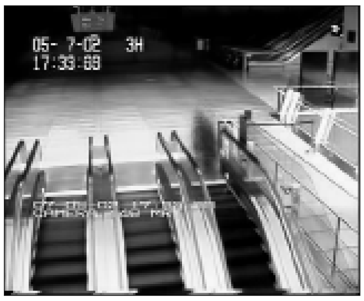

**(a)** bg frame # 1806, $p = 1.0$

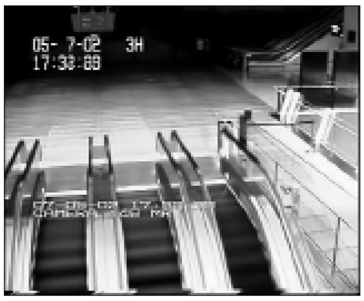

**(b)** bg frame # 1813, $p = 1.0$

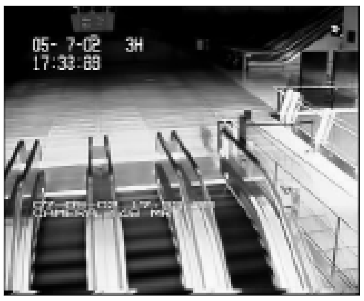

**(c)** bg frame # 1820, $p = 1.0$

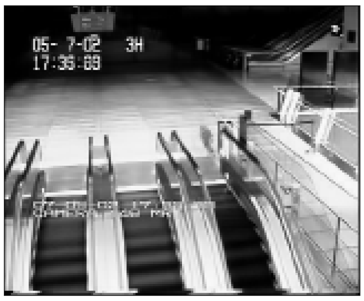

**(d)** bg frame # 1806, $p = 0.7$

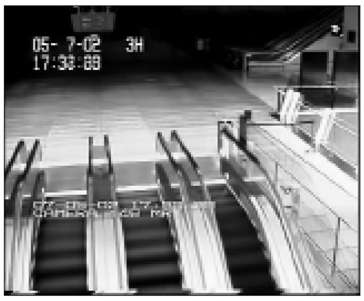

**(e)** bg frame # 1813, $p = 0.7$

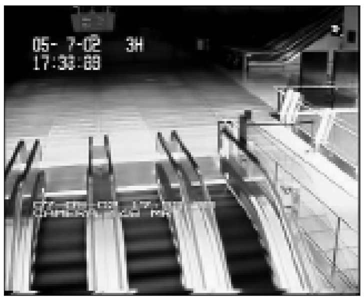

**(f)** bg frame # 1820, $p = 0.7$


**(g)** bg frame # 1806, $p = 0.4$


**(h)** bg frame # 1813, $p = 0.4$


**(i)** bg frame # 1820, $p = 0.4$


**(j)** bg frame # 1806, $p = 0.1$


**(k)** bg frame # 1813, $p = 0.1$

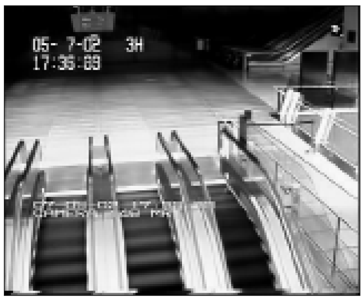
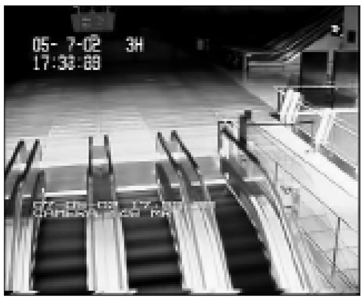
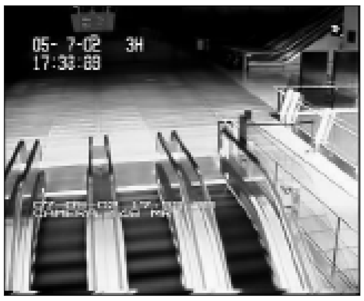
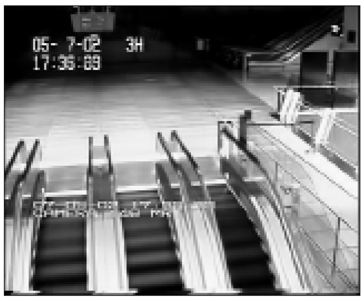
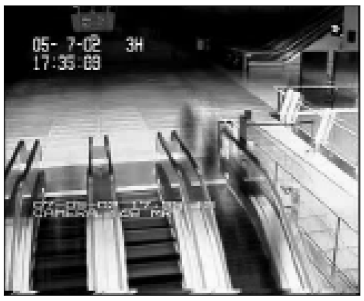

**(l)** bg frame # 1820, $p = 0.1$

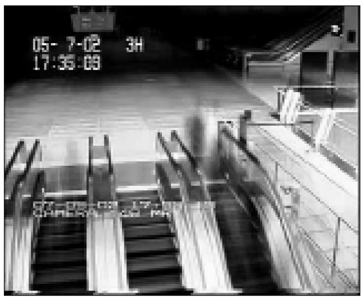
**Figure 4.16.:** Background estimates for frames # 1806, 1814 and 1820 of the *escalator* sequence, obtained using *GRPCA* with varying cost function parameter $p$.

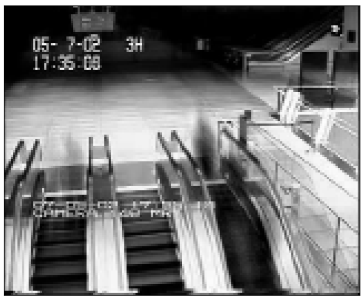

**(a)** bg frame # 1806, *GRPCA*

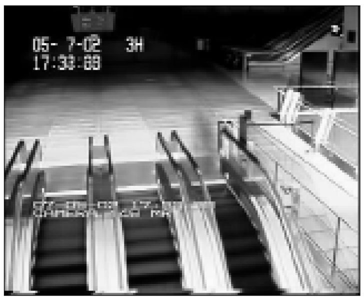

**(b)** bg frame # 1813, *GRPCA*

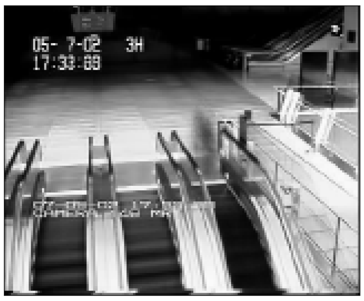

**(c)** bg frame # 1820, *GRPCA*

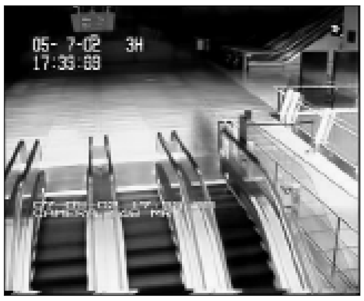

**(d)** bg frame # 1806, *IALM*

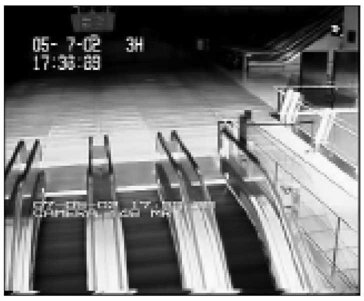

**(e)** bg frame # 1813, *IALM*

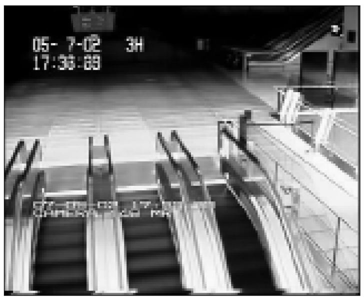

**(f)** bg frame # 1820, *IALM*


**(g)** bg frame # 1806, *LMaFit*


**(h)** bg frame # 1813, *LMaFit*


**(i)** bg frame # 1820, *LMaFit*


**(j)** bg frame # 1806, *GoDec*


**(k)** bg frame # 1813, *GoDec*

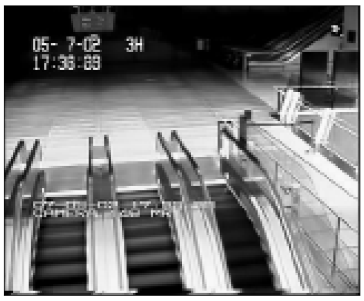
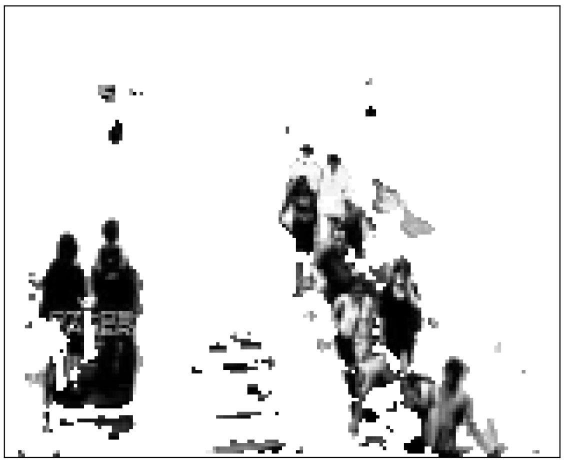
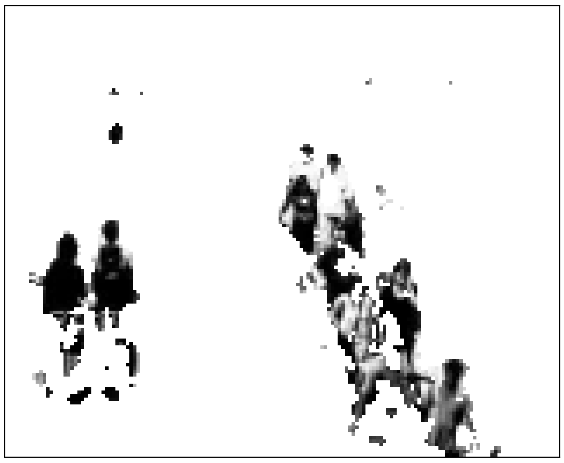
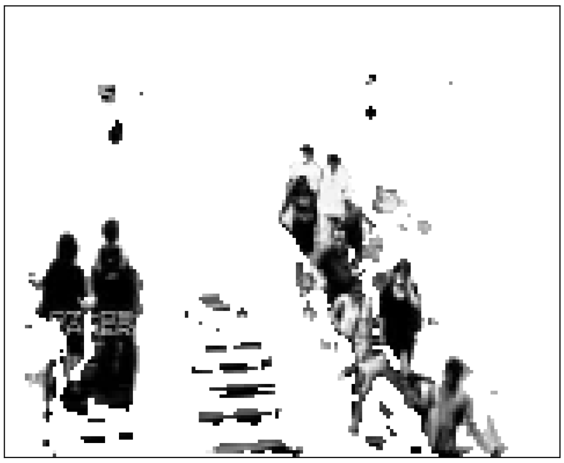

**(l)** bg frame # 1820, *GoDec*

**Figure 4.17.:** Comparison of background estimates for frames # 1806, 1814 and 1820 of the *escalator* sequence. From top to bottom: *GRPCA* with $p = 0.1$, *IALM*, *LMaFit*, *GoDec*


**(a)** foreground frame # 1813, *GRPCA*


**(b)** foreground frame # 1813, *IALM*


**(c)** foreground frame # 1813, *LMaFit*

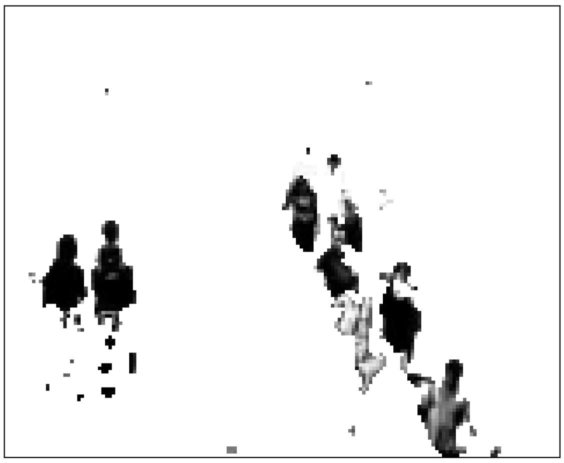

**(d)** foreground frame # 1813, *GoDec*

**Figure 4.18.:** Comparison of foreground estimates for frame # 1813 of the *escalator* sequence. From top left to bottom right: *GRPCA* with $p = 0.1$, *IALM*, *LMaFit*, *GoDec*

| Method | Processing time in seconds |
|---|---|
| ***GRPCA*** | **140** |
| *LMaFit* | 170 |
| *GoDec* | 435 |
| *IALM* | 1750 |

**Table 4.2.:** Run time comparison of state-of-the-art Robust PCA algorithms for foreground-background segmentation of the *escalator* video sequence (dimensions $20,800 \times 3417$)

a person has the same properties as the person itself. Also, the last digit of the time stamp in the upper left is detected as a foreground object. This illustrates how ambiguous such a comparison can be and makes it difficult to draw general conclusions about the superiority of one approach above the other.

What can be measured and compared well, on the other hand, is the run time of the respective algorithms, listed in Table 4.2. Here, the proposed method is the fastest with about 140 (at $p = 0.1$) to 170 (at $p = 1.0$) seconds, *LMaFit* is in the same range with about 170 seconds, *GoDec* requires 435 seconds and, lastly, *IALM* requires 1750 seconds, which supports the observation of Shen et al. [75] that factorization approaches are much more efficient for large scale data than methods employing nuclear norm minimization.

To summarize, the proposed *GRPCA* method is able to perform the foreground-background segmentation of a video of 3417 frames at a resolution of $160 \times 130$ within about two to three minutes, which is slightly faster than competing factorization-based models, while being a magnitude faster than a nuclear-norm-minimization approach. The proposed smoothed $\ell_p$-norm loss function leads to improved robustness against ghosting artifacts while still being able to capture repetitive motion of dynamic backgrounds.

# Chapter 5.

# Robust Subspace Tracking on the Grassmannian

The setting discussed in Chapter 4 assumed an underlying subspace being constant over time, whereas in this chapter the case of temporally evolving subspaces will be discussed. A sequence of data points $\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(n)}$ will be considered, with $\boldsymbol{x}_{(j)} \in \mathbb{R}^m$ denoting a single data sample at time $j$, which is assumed to follow the data model

$$\boldsymbol{x}_{(j)} = \boldsymbol{l}_{(j)} + \boldsymbol{s}_{(j)}, \quad \boldsymbol{l}_{(j)} = \boldsymbol{V}_{(j)} \boldsymbol{w}_{(j)}, \quad \left\| \boldsymbol{s}_{(j)} \right\|_0 \ll m \tag{5.1}$$

with $\boldsymbol{V}$ denoting a $k \times m$-dimensional orthogonal frame, $\boldsymbol{w} \in \mathbb{R}^k$ and $\boldsymbol{s} \in \mathbb{R}^m$ being sparse. The data model resembles the low-rank-and-sparse data model from Equation (1.6), in which $n$ samples of $m$-dimensional data $\boldsymbol{X} \in \mathbb{R}^{m \times m}$ are assumed to be the sum of a low-rank matrix $\boldsymbol{L} \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(\boldsymbol{L}) \leq k$ and a sparse matrix $\boldsymbol{S} \in \mathbb{R}^{m \times n}$ with $\|\boldsymbol{S}\|_0 \ll mn$. Considering a constant $\boldsymbol{V}$ in Equation (5.1), the data models are equivalent. However, it will be assumed in the following that $\boldsymbol{V}$ and thereby the underlying subspace of $\boldsymbol{l}_{(j)}$ may vary over time.

To address this problem, a Robust Subspace Tracking algorithm will be presented that learns a subspace incrementally, while being robust against outliers in the observed data samples. As outlined in Section 3.1, a $k$-dimensional subspace in $m$-dimensional surrounding space may be represented with an element $\boldsymbol{U} \in \mathrm{St}_{k,m}$ of the Stiefel manifold. Although this matrix representation is not unique in general, a subspace can uniquely be identified with an element of the Grassmannian, which is the equivalence class $[\boldsymbol{U}] \in \mathrm{Gr}_{k,m}$, see Section 3.2. Therefore, if the underlying subspace of an observed data set changes gradually over time this can be identified with a trajectory across the Grassmann manifold.

## 5.1. Algorithmic Description

The Robust Subspace Tracking problem searches for a sequence of vectors $\hat{\boldsymbol{l}}_{(1)}, \hat{\boldsymbol{l}}_{(2)}, \ldots, \hat{\boldsymbol{l}}_{(n)}$ that minimize the sample-wise residual error between $\boldsymbol{x}_{(j)}$ and $\hat{\boldsymbol{l}}_{(j)}$ at every time instance. Following the data model in Equation (5.1), a vector $\hat{\boldsymbol{l}}_{(j)}$ will be modeled as the product of an $m \times k$-dimensional Stiefel matrix $\boldsymbol{U} \in \mathrm{St}_{k,m}$ with a $k$-dimensional vector $\boldsymbol{y} \in \mathbb{R}^k$, such that $\hat{\boldsymbol{l}} = \boldsymbol{U}\boldsymbol{y}$. As the residual error is assumed to be sparse, the smoothed $\ell_p$-norm loss function from Equation (4.8) will be employed as a sparsifying function. A sample may be observed either fully or partially on the current observation set $\Omega_{(j)}$, so that the loss function to be minimized over $\boldsymbol{U}$ and $\boldsymbol{y}$ for every sample $j$ writes as $h_\mu\left(\mathcal{P}_{\Omega_{(j)}}\left(\boldsymbol{x}_{(j)} - \boldsymbol{U}\boldsymbol{y}\right)\right)$.

At first glance, the problem seems ill-posed as infinitely many solutions exist that achieve a residual error of zero, e.g. $\boldsymbol{u}_1 = \boldsymbol{x}/\left\|\boldsymbol{x}\right\|_2$ and $\boldsymbol{y} = \boldsymbol{e}_1$. To avoid such trivial solutions and to make the rank constraint and the whole subspace tracking process meaningful, the admissible set of solutions for $\boldsymbol{U}_{(j+1)}$ must be reduced to a neighborhood $\mathcal{U}$ around a previous subspace estimate $\left[\boldsymbol{U}_{(j)}\right]$. Recalling the discussion on matrix representations of subspaces in Section 3.1, it becomes obvious that $\mathcal{U}$ should be defined on the Grassmannian. The neighborhood is implicitly defined by limiting the admissible distance between $\left[\boldsymbol{U}_{(j)}\right]$ and $\left[\boldsymbol{U}_{(j+1)}\right]$. Only a single gradient step on the Grassmannian is taken for updating $\boldsymbol{U}$, while bounding the step size by selecting a small value $t_{\text{start}}$ for the backtracking line search algorithm (Algorithm 3.2). The coordinates $\boldsymbol{y}$, on the other hand, are updated just as in the batch case with a CG method on the full $\mathbb{R}^k$, which may take several CG iterations. Yet, the tracking algorithm alternates just once per sample between updating $\boldsymbol{U}$ and $\boldsymbol{y}$. A detailed description of the procedure can be found in Algorithm 5.1.

The algorithm reads and processes one data sample at a time, which reduces the memory requirements tremendously, as the biggest storage requirement is now the $m \times k$ matrix $\boldsymbol{U}$ with $k$ commonly being small. A good way of initializing the tracking algorithm is to accumulate a certain number of samples and to run the batch *GRPCA* algorithm (Algorithm 4.1) in order to provide a first subspace estimate as a starting point of the tracking algorithm. But it is also possible to start the tracking algorithm without any initialization samples, so that the subspace is learned incrementally. The estimation of $\boldsymbol{y}$ can be initialized in various ways: If no assumptions on $\boldsymbol{y}$ can be made a priori, $\boldsymbol{y}$ is either initialized randomly or with all zeros. This, however, requires a larger number of CG iterations for the minimization of the residual to fully converge. In some applications (e.g. video segmentation) it is reasonable

**Algorithm 5.1** Alternating minimization scheme for Robust Subspace Tracking

```
Input: Partially observed data sequence $\mathcal{P}_{\Omega_{(j)}}\left(\boldsymbol{x}_{(j)}\right), \quad j = 1, \dots, n$
Optional: Choose set of init samples and initialize $\boldsymbol{U}$ with Alg. 4.1
Or: Initialize $\boldsymbol{U}$ randomly
Choose smoothing parameter $\mu$ according to noise level
for every new sample $\boldsymbol{x}_{(j+1)}$ observed on $\Omega_{(j+1)}$ do
    Update $\boldsymbol{y}_{(j+1)} = \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} h_\mu\left(\mathcal{P}_{\Omega_{(j+1)}}\left(\boldsymbol{x}_{(j+1)} - \boldsymbol{U}_{(j)}\boldsymbol{y}\right)\right)$
    Update $\boldsymbol{U}_{(j+1)} = \arg\min_{[\boldsymbol{U}]\in\mathrm{Gr}_{k,m}} h_\mu\left(\mathcal{P}_{\Omega_{(j+1)}}\left(\boldsymbol{x}_{(j+1)} - \boldsymbol{U}\boldsymbol{y}_{(j+1)}\right)\right)$ with single-step gradi-
    ent descent on $\mathrm{Gr}_{k,m}$ (Alg. 3.1)
    Compute low-rank estimate $\hat{\boldsymbol{l}}_{(j+1)} = \boldsymbol{U}_{(j+1)}\boldsymbol{y}_{(j+1)}$
end for
Outputs:
$\hat{\boldsymbol{l}}_{(j)}, \quad j = 1, \dots, n$
$\hat{\boldsymbol{s}}_{(j)}, \quad j = 1, \dots, n, \quad \hat{\boldsymbol{s}}_{(j)} = \boldsymbol{x}_{(j)} - \hat{\boldsymbol{l}}_{(j)}$
```

to assume that subsequent coordinate vectors $y$ are close to each other, especially whenever the input samples are similar. In this case the previous estimate of $\boldsymbol{y}$ should serve as a starting point for the next estimation based on the new input sample, as this likely leads to much faster convergence than random initialization. Lastly, if it is known that subsequent estimates of $\boldsymbol{y}$ are not close to each other but the outlier density in the current input is low, it may be reasonable to initialize the CG method with $\boldsymbol{y}_0 = \boldsymbol{U}_{(j)}^\top \boldsymbol{x}_{(j+1)}$, which are the coordinates of the orthogonal projection of the current input sample onto the previously found subspace. Both the cost function and its gradient follow straightforwardly as the special case $n = 1$ from Equation (4.12) and Equation (4.14), respectively, and the minimization via CG is performed in the same way as the minimization of $\boldsymbol{Y}$ in the batch case. Apart from the dimensions the only difference is that the subsampling approach proposed in Section 3.4 is not reasonable in the tracking case, because the optimization alternates only once per sample between $\boldsymbol{U}$ and $\boldsymbol{y}$, so that the estimate of the coordinates is final after the CG optimization. In general, this also forbids to terminate the optimization of $\boldsymbol{y}$ prematurely, which is why the CG method is commonly run until a reasonable convergence level is attained. The convergence is observed through evaluating the relative progress in decreasing the cost function, which is computed analogously to the relative progress in the batch case, Equation (4.15). The optimization of $\boldsymbol{y}$ is considered converged if the relative progress reaches a threshold of $10^{-8}$. On the upside, the dimension of $\boldsymbol{y}$ is commonly very small and typically allows for quick convergence in practice. This is a great advantage in comparison to the related method *GRASTA*. As the *GRASTA* method approaches the Robust Subspace Tracking problem with an augmented Lagrangian multiplier scheme, three variables need to be updated in an Alternating Minimization process in order to update the coordinates of the subspace, whereas in this approach one variable is sufficient.

After the coordinates of the current data sample in the previously determined subspace have been found, the subspace representative $\boldsymbol{U}$ is updated by minimizing the cost function $f_U(\boldsymbol{U}) = h_\mu\left(\mathcal{P}_{\Omega_{(j+1)}}\left(\boldsymbol{x}_{(j+1)} - \boldsymbol{U}\boldsymbol{y}_{(i+1)}\right)\right)$, which uses the updated coordinates. Following Algorithm 3.1, the search direction on the manifold needs to be determined from the projected Euclidean gradient of the cost function. Computing the geodesics (3.12) requires an SVD of the search direction, which is the costliest part of a gradient descent on the Grassmannian in the batch case. However, as mentioned by Balzano et al. [4], computing an actual SVD can be avoided in the tracking setting, which will be illustrated in the following. Assuming without loss of generality that all entries are observed and omitting the temporal indices for ease of notation, the negative Riemannian gradient (i.e. the search direction for a gradient descent) writes as

$$-\boldsymbol{g} = \left(\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top\right) \nabla h_\mu \left(\boldsymbol{x} - \boldsymbol{U}\boldsymbol{y}\right)\ \boldsymbol{y}^\top =: \boldsymbol{a}\boldsymbol{y}^\top, \tag{5.2}$$

which is a rank-one matrix and allows for a simple Singular Value Decomposition

$$\sigma \boldsymbol{u}\boldsymbol{v}^\top \quad \text{with} \quad \sigma := \|\boldsymbol{a}\|_2 \|\boldsymbol{y}\|_2\,, \quad \boldsymbol{u} := \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_{\boldsymbol{2}}}, \quad \boldsymbol{v} := \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_{\boldsymbol{2}}}\ , \tag{5.3}$$

which does not involve any costly computation.

Once all the required ingredients are derived, the subspace is updated with a single gradient descent step on the Grassmannian. Subsequently, the current low-rank estimate $\hat{\boldsymbol{l}} = \boldsymbol{U}\boldsymbol{y}$ is computed and the next input vector is read.

## 5.2. Evaluation on Simulated Data

The evaluation of the *GRPCA* method in Section 4.3 has shown under which circumstances employing a smoothed $\ell_p$-norm as a loss function outperforms competing methods based on the $\ell_1$ norm. As the proposed method for Robust Subspace Tracking follows the same principles and uses the same cost function to measure the residual error, the main goal of the following evaluation on simulated data is to investigate how well the results transfer to the tracking case and what influence the choice of the cost function parameter $p$ has in an online setting. No initialization phase with *GRPCA* is performed in the experiments, but the low-rank approximation is performed on a sample-by-sample basis. As the main focus of this chapter is on a practical application and realtime processing, the parameters for the following experiments have been chosen in favor of fast convergence speed at reasonably good accuracy.

The first experiment uses the data model proposed by He et al. [42] for evaluating the *GRASTA* method on a stationary subspace identification task. Data samples are generated via

$$\boldsymbol{x}_{(j)} = \boldsymbol{U}\boldsymbol{y}_{(j)} + \boldsymbol{s}_{(j)} + \boldsymbol{n}_{(j)} \tag{5.4}$$

where $\boldsymbol{U}$ is a random Stiefel matrix, the coordinates $\boldsymbol{y}$ are randomly distributed according

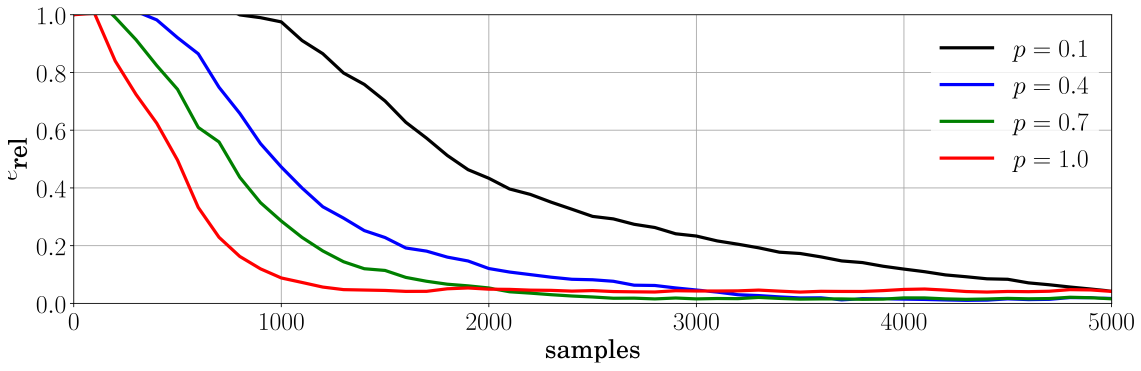


**Figure 5.1.:** Progression of relative error during the online identification of a static subspace of dimension $k = 5$ within surrounding dimension $m = 500$ from 10% of samples with 10% sparse outliers, evaluated for different values of cost parameter $p$

to $\mathcal{N}(0,1)$, $\boldsymbol{s}$ contains sparse outliers and $\boldsymbol{n}$ is additive Gaussian noise with $\sigma^2 = 10^{-5}$. As the coordinate vectors are drawn independently, there is no temporal coherence in the data, so that an initialization with a previous estimate would not be beneficial, which is why the minimization $\boldsymbol{y}$ is initialized with an all-zero vector.

### Robust Subspace Tracking with sparse outliers and missing data

In the first scenario a problem of dimension $m = 500$ with rank $k = 5$ is considered, the outlier density is chosen as 0.1 and $|\Omega|/m = 0.1$, i.e. 10% of the entries are observed. A total of $n = 5000$ samples are generated according to the data model (5.4) and they are processed one at a time to determine the subspace. The maximum step size of the gradient descent for updating $\boldsymbol{U}$ is limited by initializing the line search with $t_{\text{start}} = 1$. Algorithm 5.1 is run with varying values for the cost function parameter $p$, which together with the noise level determines the choice of the smoothing parameter $\mu$. As the heuristic proposed in Section 4.1 would lead to an infeasible value $\mu = 0$ whenever $p = 1$, a value for $\mu$ corresponding to $p = 0.9$ has been selected instead. The progression of minimizing the relative error over the number of observed samples is visualized in Figure 5.1. The experiment shows that the subspace is reconstructed in all cases up to a certain accuracy, while the speed of convergence slows down the lower $p$ is chosen. The processing time is about 40 ms per sample regardless of the value for $p$. This could lead to the conclusion that the (smoothed) $\ell_1$ norm is always a better choice in a tracking scenario than $p < 1$. Yet, recalling the results of Chapter 4, it

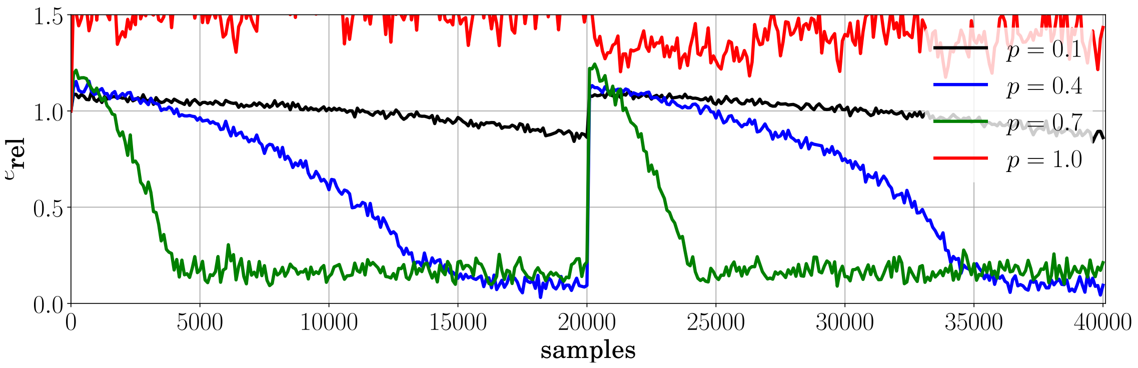


**Figure 5.2.:** Progression of relative error while tracking subspaces of dimension $k = 5$ within surrounding dimension $m = 50$ with 60% dense outliers, with rapid subspace change at $j = 15000$, evaluated for different values of cost parameter $p$

has been observed that this norm may be a suboptimal choice whenever the low-rank-and-sparse assumptions are not exactly met. Therefore, another experiment with dense outliers and a moderate rank is considered.

### Robust Subspace Tracking with changing subspace and dense outliers

For this experiment, two data sets of length $\frac{n}{2} = 20,000$ are created. Again, the data model (5.4) is used, but the surrounding dimensions are reduced to $m = 50$ while raising the rank to $k = 10$ and setting the outlier density to 50%. The two data sets are concatenated, resulting in one overall data set with an abrupt subspace change at $j = \frac{n}{2}+1$. All entries are revealed. He et al. [42] have shown that an $\ell_1$-based robust subspace tracking method can reconstruct a subspace even with dense outliers, but they investigated a data model with a relative rank of $k/m = 0.01$, whereas in this experiment $k/m = 0.2$. The results in Figure 5.2 demonstrate that the proposed robust subspace tracking method fails to reconstruct the subspace with $p = 1.0$, while lower values for $p$ such as 0.7 or even 0.4 allow for successful reconstruction. However, an overly small choice $p = 0.1$ slows down the convergence such that the subspace cannot be identified before it is changing again. The processing time again is independent of $p$ with about 80 ms per sample.

The conducted experiments on simulated data indicate that the proposed robust subspace tracking algorithm is capable of learning a low-dimensional subspace online from incompletely observed and outlier-contaminated data. As for the Robust PCA case, the tracking algorithm is able to resolve a broader range of scenarios whenever a small value is chosen for the cost function parameter $p$. Yet, an overly small value for $p$ slows down the convergence of the tracking algorithm.

## 5.3. pROST - A specialized Robust Subspace Tracking algorithm for real-time background separation in video

The following section, which is based in great parts on the publication [72], describes the *pROST* algorithm. The *smoothed $\ell_p$-norm Robust Online Subspace Tracking method for real-time background subtraction in video* has been developed in order to evaluate the presented robust subspace tracking algorithm on a video segmentation benchmark. As the name of the method suggests, an explicit constraint was to achieve realtime performance, i.e. to be able to read, process and segment video frames at a rate of at least 25 frames per second on a standard computer. For this purpose an efficient implementation of the proposed subspace tracking method using GPU acceleration has been prepared. The original code for the publication [72] was written in *C/C++* and used the *CUDA* framework as well as some specialized libraries, see Hage et al. [39] for more details. Meanwhile, a second implementation has been prepared in *Python* and is available at `https://github.com/FlorianSeidel/GOL`. The new implementation uses the widely-popular libraries *NumPy*, *SciPy* and the GPU-supporting library *Theano*. It is therefore easier to install and to configure than the former implementation at the cost of slightly reduced computational performance.

### 5.3.1. Motivation

In Section 4.4.1, the principles of background subtraction via Low-Rank Approximation have been introduced and discussed, and it became obvious that a classical PCA approach is not able to model outliers well. In the practical context of background modeling, the following problems can be specified: Firstly, in common PCA undue weight is given to the foreground elements when fitting the background model to camera frames during the segmentation process. This limits the admissible size of foreground objects and can only be overcome by employing additional workarounds such as adaptive thresholding [82]. Secondly, images containing foreground objects can lead to corruption of the background model and ghosting artifacts. Batch-processing large video sequences assumes a static statistical distribution of the data and may result in tremendous memory requirements.

Experimental results in Section 4.4.2 have demonstrated that the proposed *GRPCA* algorithm addresses the first two problems, as the algorithm is capable of reconstructing a subspace in the presence of a great number of outliers and of learning a background model without requiring a clean observation of the background. Be that as it may, the third problem remains unsolved, as the algorithm processes the whole video sequence at once. Besides the memory issue, a batch algorithm lacks flexibility in uncertain real-world conditions. For example, it might not be feasible to observe a scene in all possible lighting or weather conditions during an initial training phase. Thus, the algorithm must be able to adapt the background model dynamically. One of the first attempts of modeling backgrounds dynamically with a PCA-like approach has been proposed by Li [59], who present an iterative PCA algorithm with a weighted cost function as an outlier treatment for foreground objects. Instead of learning a PCA model and updating the eigenvectors, Subspace Tracking on the Grassmannian identifies a learned background model with a single point on the manifold and is able to gradually track a change in the model by moving along the manifold. Balzano et al. [4] developed a Grassmannian Subspace Tracking method *GROUSE* and subsequently developed the robust counterpart *GRASTA* by replacing the $\ell_2$ norm with the $\ell_1$ norm. The *pROST* algorithm [72] tries to push the robustness against outliers even further by employing a smoothed $\ell_p$ norm and is especially designed for the video segmentation problem.

### 5.3.2. Description of the algorithm

The processing steps of *pROST* follow the algorithmic framework of the proposed robust subspace tracking method in Algorithm 5.1. That is, the algorithm reads and processes one sample (one vectorized image frame) at a time. Firstly, the optimum coordinates within the previously estimated subspace are obtained. Due to the coherence of subsequent video frames the previous estimate for $\boldsymbol{y}$ serves as a good initialization, such that the coordinate update can typically be achieved within few CG iterations. Subsequently, the subspace estimate is refined with a single gradient step along the geodesics on the Grassmannian. As video segmentation does not only require an estimate of the background $\boldsymbol{l}$ in the current frame, the foreground has to be reconstructed explicitly by thresholding the residual error

$\boldsymbol{x} - \boldsymbol{l}$ and extracting the actual foreground frame from $\boldsymbol{x}$ with a binary mask. Further pre- and post-processing steps tailor the algorithm to the video segmentation task.

### Weighted smoothed $\ell_p$-norm cost function

The pROST algorithm is designed with background subtraction for video streams in mind, and thus the cost function can be optimized for this setting. In video data it is sensible to assume that corresponding pixels in consecutive frames are likely to have the same label. This knowledge can be used to further increase the robustness of the residual loss. The idea is to reduce the contribution of labeled foreground pixels to the residual error by introducing additional pixel weights $w_i \in \mathbb{R}^+$, whose magnitudes depend on the labels assigned to the pixels in the previous frame. If the pixel was previously labeled a foreground pixel and is therefore likely to remain a foreground pixel in the current frame, the weighting should be small to avoid foreground objects compromising the background. In the reverse case, if the pixel was labeled a background pixel before, the weight should be equal to one to allow for model maintenance. In this way the algorithm avoids erroneously fitting the background model to already known foreground objects and it can focus on fitting the background model to the scene background instead. This extension to the cost function further eases to learn a background from corrupted training data and to deal with large foreground objects. *pROST* enforces a sparse residual error with the weighted smoothed $\ell_p$-norm cost function

$$h_{\mu,w} : \mathbb{R}^m \to \mathbb{R}^+, \boldsymbol{x} \mapsto \sum_{i=1}^{m} w_i \left( x_i^2 + \mu \right)^{\frac{p}{2}}, \quad 0 < p < 1, \tag{5.5}$$

The loss function is not normalized since *pROST* operates at constant resolution and processes full video frames, so that the number of entries is always constant. Depending on the pixel-wise foreground/background classification, the weights are assigned at the end of each iteration as

$$w_i = \begin{cases} 1 & \text{if } |x_i - \boldsymbol{u}_i^\top \boldsymbol{y}| < \tau \\ \omega & \text{else} \end{cases}, \tag{5.6}$$

where $\omega$ is the weight for the foreground pixel reconstruction error. In order to be able to slowly incorporate foreground pixels into the background, $\omega$ should be set to a small but non-zero value.

### RGB channels

If colored video is available, it is clearly advantageous to use the information provided in the color channels, especially to cope with the problem of camouflaging (cf. Section 4.4.1). In *pROST*, a colored and vectorized video frame of $m$ pixels is represented as a vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_G \\ \boldsymbol{x}_B \end{bmatrix} \in \mathbb{R}^{3m}, \tag{5.7}$$

where the $i$-th entry of $\boldsymbol{x}_R, \boldsymbol{x}_G, \boldsymbol{x}_B \in \mathbb{R}^m$ is given by the respective channel value at pixel $i$. The background is accordingly represented by a matrix

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_R \\ \boldsymbol{U}_G \\ \boldsymbol{U}_B \end{bmatrix} \in \mathrm{St}_{k,3m}\,. \tag{5.8}$$

The pixel $i$ is classified as foreground if the difference between the reconstructed background of either of the channels surpasses a certain threshold $\tau$, i.e. if

$$\max\{|x_i - \boldsymbol{u}_{i,:}^\top \boldsymbol{y}|, |x_{i+m} - \boldsymbol{u}_{i+m,:}^\top \boldsymbol{y}|, |x_{i+2m} - \boldsymbol{u}_{i+2m,:}^\top \boldsymbol{y}|\} \geq \tau. \tag{5.9}$$

### Pre- and post-processing

A sliding average is computed for every pixel and the resulting mean image is subtracted from each frame before *pROST* is applied. This means the algorithm has to model only the varying parts of a scene, which has proven beneficial especially for capturing dynamic backgrounds. Furthermore, the images are scaled to the range $[0, 1]$. To achieve fast and uniform processing, all videos are resampled to an internal resolution of $120 \times 90$[1]. This concerns only the resolution of the segmentation masks, which can be upscaled again to the full resolution such that the input sequence can be segmented at full scale. After processing the frame and thresholding the residual $|\boldsymbol{x} - \boldsymbol{l}|$, a $3 \times 3$ median filter is applied to the resulting

[1]The C++ / CUDA implementation presented in [72] achieved realtime performance on a resolution of $160 \times 120$ frames, which shows that the perks of having an easy-to-use Python / Theano implementation come at the cost of losing some computational performance

foreground segmentation mask. This serves to fill small holes and to get rid of small clusters of erroneously labeled pixels.

### Initialization phase

Often times, algorithms for background subtraction require an initialization phase in which the background is learned. A major advantage of the robust subspace tracking approach is that the algorithm can learn a subspace online, i.e. no batch initialization is required. Furthermore, a subspace can be learned robustly from samples containing outliers, i.e. the related background subtraction method does not require a separate learning phase in which the non-obscured background is observed. The algorithm is initialized with a random subspace representative $\boldsymbol{U} \in \mathrm{St}_{k,3m}$. Subsequently, the background subspace is learned one frame at a time. Even though an initialization phase is not required it is useful in practice to start with a comparatively large step size $t_{\text{init}}$ and to reduce the step size gradually, down to a value $t_{\text{online}}$, which is kept for the online updates. The initial step size should be chosen quite high ($t_{\text{init}} \in [10^{-4}, 1]$) to facilitate quick initialization, while the latter should be chosen quite small ($t_{\text{online}} \in [10^{-7}, 10^{-4}]$) in order to restrict the search range for $[\boldsymbol{U}]$, cf. the discussion in Section 5.1. During this initialization period, the step size for the subspace update at the $j$-th frame are defined by the step-size rule

$$t = \max\{e^{-\alpha j} t_{\text{start}}, t_{min}\}, \tag{5.10}$$

where $\alpha$ is a parameter controlling the shrinkage rate for the step size reduction. Whenever an initialization phase is defined by an exact number of frames $J_{\text{init}}$, the parameter $\alpha$ can be calculated as

$$\alpha = \frac{-\log\left(\frac{t_{\text{online}}}{t_{\text{init}}}\right)}{J_{\text{init}}} \,. \tag{5.11}$$

### 5.3.3. Evaluation on the *changedetection.net* benchmark

One of the main difficulties with comparing different background subtraction methods has been the lack of an accepted benchmark. Various data sets exist (e.g. [58] and [78]) that provide video sequences and few manually segmented test images. However, the lack of pixel-level ground truth for whole video sequences has led to rather subjective evaluation as criticized in [13]. The authors overcome the cumbersome task of hand-segmenting video sequences by providing an artificially rendered scene, which allows a very detailed and precise segmentation. But although the animation is claimed to be close to photo-realistic the overall visual impression and statistics are fundamentally different from real video.

In order to establish a benchmark on real-world video sequences, the *changedetection.net* data set [37] has been introduced. The data set consists of six categories of videos and provides ground truth for each frame. Categories vary from strictly static (*baseline*) over *dynamic* backgrounds to shaking cameras (*jitter*), scenes with particular objects changing positions (*intermittent object motion*) and sequences of *thermal* images. The ground truth contains information about background and foreground objects as well as their boundaries and shadows (specifically evaluated in the *shadows* category).

For the evaluation of *pROST* on the *changedetection.net* benchmark, one overall set of parameters needs to be chosen. Obviously, this trade-off leads to suboptimal results as some scenarios require a different parameter setting than others. What follows is a brief discussion of the parameter settings and their influence while a much more detailed evaluation can be found in [72].

### Subspace dimension

The admissible dimension $k$ of the subspace defines the inner dimensions of the optimization variables and thereby the size of the search space for the optimization problem. As this defines the computational complexity, $k$ should be chosen as small as possible, while still offering sufficient degrees of freedom for modeling complex backgrounds. Empirical results show that $k$ can be chosen very small if the background is static, while a value of about 10 to 15 is required for complex dynamic backgrounds.

### Initial and online step sizes

The choice of the step size defines how fast the subspace tracking algorithm adjusts to changes in the background model. Large step sizes come with the advantage of fast adaptation to changes in the background and allow to learn high-dimensional backgrounds within few frames. But fast adaptation also increases the risk of foreground objects leaking into the background, as foreground objects are never ideally sparse in space and time. They often appear in the same position over the course of several frames, which in combination with large step sizes leads to the aforementioned issues like foreground leaking and ghost images. The temporal threshold for the transition from foreground to background (i.e. how long does an element have to be present in the scene to blend into background) highly depends on human perception, which is why the step size needs to be hand-tweaked. But even manual selection is difficult if one setting must be defined that should fit both static backgrounds that are constant over time as well as scenes with dynamic backgrounds or foreground-background transitions. For the initialization phase, $t_{\text{init}} = 5 \times 10^{-3}$ is selected and the online step size is chosen to be $t_{\text{online}} = 10^{-4}$. This allows the algorithm to learn a background rather quickly in the beginning and leads to a reasonable trade-off between background adaptation speed and leakage of slowly moving foreground objects.

### Foreground weighting parameter

The pixel weighting in Equation (5.6) adds a second time scale to the subspace tracking algorithm. While the overall progress in learning and adjusting a background model is controlled by the step size $t$, the foreground weighting parameter $\omega$ offers additional control on how fast foreground objects are incorporated into an existing background model. The pixel weighting has a large effect on the algorithm's capability of dealing with highly dynamic complex backgrounds. An empirical value of $\omega = 5 \times 10^{-5}$ allows learning such backgrounds from input sequences that are heavily corrupted with foreground objects, while still being able to incorporate such foreground objects into a background model if they are persistent over an extended period of time.

### Detection threshold

As the classification between foreground and background is a binary decision, the optimum value for the detection threshold of foreground objects can be determined as to maximize the overall F-score across the categories. For *pROST* the value $\tau = 0.15$ has been selected, which corresponds to about 40 intensity levels for 8-bit unsigned integer input. Again, the optimum threshold depends on the statistics of the video sequence, especially on the intensity difference between foreground and background objects.

### Cost function parameters

In contrast to the $\ell_1$ norm, the freedom of choice for the parameters $p$ and $\mu$ in the smoothed $\ell_p$-norm cost function offers additional control over the required robustness against outliers in the data. As shown in the experimental evaluation of Chapter 4 and Section 5.2, a lower value for $p$ leads to increased robustness against outliers, but it also slows down the convergence of the subspace tracking algorithm. This is why a moderately low value of $p = 0.25$ has been chosen. The value for the smoothing parameter $\mu$ is set to $\mu = 0.01$ following the heuristic (4.7) and according to the choice for $p$, the threshold $\tau$ and the intensity range after scaling.

## 5.3.4. Results and discussion

In order to compare the results of *pROST* with the segmentation ability of a state-of-the art algorithm for Robust Subspace Tracking, the *GRASTA* algorithm [42] has been evaluated on the *changedetection.net* benchmark. The authors' MATLAB implementation has been slightly adjusted in order to be able to process the sequences from the benchmark. *GRASTA* is intended to work with gray scale images, whereas *pROST* works with RGB color images. To make a comparison possible, GRASTA has been modified to work with such images. The subspace dimension is set to be $k = 15$ and video is segmented at a resolution of $160 \times 120$. The algorithm performs an initialization phase in which an initial background model is learned from a batch of training images. Within this phase, the respective initialization frames at the beginning of the sequences are processed in random order, three times in a row. The default parameters of the MATLAB implementation are used, except for the detection threshold and the percentage of pixels used for updating the subspace during the tracking stage. The demo implementation suggests to use 10% of the pixels, but as the realtime constraint is still met with more pixels, 25% are reveled to allow for a fairer comparison with *pROST*, which observes and processes full frames. While He et al. [42] suggest a threshold of $\tau = 0.1$ for segmenting the normalized image frames, the best observed value on the *changedetection.net* benchmark is 0.2, which has therefore been selected. As a post-processing step, the obtained segmentation masks are filtered by a $3 \times 3$ median filter.

The *changedetection.net* data set comes with an evaluation tool, which computes the fol-

**Table 5.1.:** Results per category for *pROST* on the *changedetection.net* benchmark.

| Category | Recall | Specif. | FPR | FNR | PWC | Prec. | F-score |
|---|---|---|---|---|---|---|---|
| baseline | 0.842 | 0.9937 | 0.0063 | 0.159 | 1.16 | 0.818 | 0.829 |
| camera jitter | 0.769 | 0.9872 | 0.0128 | 0.231 | 2.04 | 0.734 | 0.748 |
| dynamic background | 0.731 | 0.9952 | 0.0048 | 0.269 | 0.66 | 0.597 | 0.618 |
| interm. object motion | 0.516 | 0.9317 | 0.0683 | 0.484 | 8.52 | 0.474 | 0.413 |
| shadow | 0.754 | 0.9791 | 0.0209 | 0.246 | 2.99 | 0.624 | 0.677 |
| thermal | 0.429 | 0.9872 | 0.0128 | 0.571 | 4.15 | 0.794 | 0.526 |
| overall | 0.674 | 0.9790 | 0.0210 | 0.327 | 3.25 | 0.673 | 0.635 |

lowing seven statistical measures (FG=foreground, BG=background) from the segmentation masks:

- Recall: Out of all FG pixels, how many have been estimated as FG
- Specificity: Out of all BG pixels, how many have been estimated as BG
- False Positive Rate: Out of all BG pixels, how many have been estimated as FG
- False Negative Rate: Out of all BG pixels, how many have been estimated as FG
- Percentage of Wrong Classification: Out of all pixels, how many have been classified incorrectly
- Precision: How many pixels estimated as FG are actually FG pixels
- F-score: Harmonic mean between Precision and Recall

For some of the videos, the segmentation is evaluated only for certain regions of interest (ROI), while for others the whole image is evaluated. The evaluation starts after a certain number of frames, which can be used for initialization. However, these training samples have the same foreground-background distribution as the ones used for evaluation and can therefore contain foreground objects. One overall score is computed as well as separate scores for each category. All reported results are conveniently accessible on the website of the benchmark. The detailed results for pROST in the current Python implementation and for *GRASTA* are listed in Table 5.1 and Table 5.2, respectively. As the performance is evaluated per category, the subsequent discussion of the results is structured in the same way.

**Table 5.2.:** Results per category for *GRASTA* on the *changedetection.net* benchmark.

| Category | Recall | Specif. | FPR | FNR | PWC | Prec. | F-score |
|---|---|---|---|---|---|---|---|
| baseline | 0.609 | 0.9926 | 0.0074 | 0.391 | 2.13 | 0.740 | 0.664 |
| camera jitter | 0.622 | 0.9282 | 0.0718 | 0.378 | 8.36 | 0.354 | 0.434 |
| dynamic background | 0.701 | 0.9760 | 0.0240 | 0.299 | 2.61 | 0.262 | 0.355 |
| interm. object motion | 0.311 | 0.9842 | 0.0158 | 0.689 | 6.32 | 0.515 | 0.359 |
| shadow | 0.608 | 0.9554 | 0.0446 | 0.392 | 6.09 | 0.536 | 0.529 |
| thermal | 0.344 | 0.9851 | 0.0149 | 0.656 | 6.13 | 0.726 | 0.428 |
| overall | 0.533 | 0.9702 | 0.0298 | 0.467 | 5.27 | 0.522 | 0.461 |

### Baseline

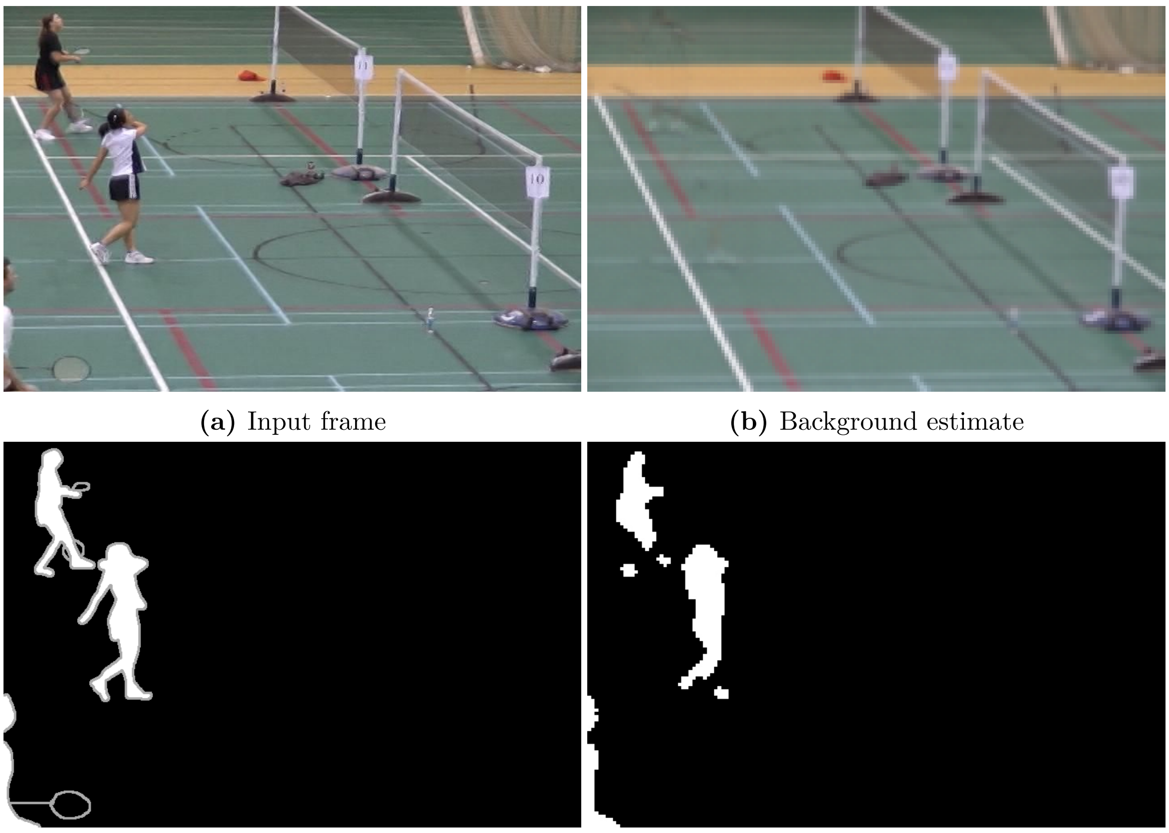
The *baseline* category contains videos with static backgrounds and foreground objects moving on different time scales throughout the sequences. This is clearly the simplest scenario and could already be modeled quite well by naive subtraction of a static background image. As a consequence, all algorithms in the benchmark including *pROST* perform very well. One minor flaw of *pROST* is the incorporation of very slowly moving foreground objects, which could easily be dealt with by selecting smaller values for the step size $t$ and the weights $\omega$. Such a parameter setting, however, would severely decrease the performance for dynamic backgrounds.

### Camera Jitter

The backgrounds in the *camera jitter* category mainly consist of static elements. But due to the shaking (*jittering*) movement of the camera the actually captured backgrounds are highly more dynamic than the previous category. This is the category in which *pROST* achieves better results than the majority of the competition, as it ranks 9th out of 40 methods when comparing the F-score. The jittery nature of the subspace can be learned quickly and accurately and the foreground objects are well extracted. Figure 5.3 illustrates the segmentation result for *pROST* on the *badminton* sequence.

### Dynamic Background

The *dynamic background* category contains videos whose background contains dynamic elements such as water surfaces, fountains or waving trees. As the experiment in Section 4.4



**(a)** Input frame **(b)** Background estimate

**(c)** Segmentation ground truth **(d)** Segmentation estimate

**Figure 5.3.:** Input, ground truth and segmentation result of *pROST* for frame # 1150 of the *badminton* sequence from the *changedetection.net* benchmark

demonstrates, pROST is well-suited to model repetitive motion, and the learned backgrounds in the dynamic background category actually contain repetitive motion. However, the kind of backgrounds contained in this particular category of the benchmark exhibit higher-dynamic or quasi-random movement, as the movement is mostly influenced by wind. As a result, their dynamics are too complex and their movements are too erratic to be modeled well by repetitive motion. It needs to be remarked that the F-score of *pROST* varies tremendously throughout the sequences within this category, reaching a top value of 0.93 for a sequence in which a canoe on a river crosses the scene, and a low of 0.12 in a sequence with a car passing a scenery of water fountains in the far distance. In the latter case, the distinction between foreground and background is arguably non-trivial, even for a human observer.

### Intermittent Object motion

While most foreground objects in other categories are moving about in the scene on a comparably fast timescale, in this category the foreground objects exhibit intermittent motion. This means that they are placed into the scene but stay at a fixed position for some time before being relocated or removed again. As previously discussed, *pROST* distinguishes between foreground and background objects solely based on their temporal evolution. Yet, the segmentation in this category requires additional contextual information about the changes in the scene, which is why the method performs rather poorly here. Whenever an object remains at a certain position for a longer period, *pROST* slowly incorporates it into the background model, which results in false negative errors in this category. Even worse, when the object is being moved again after having been incorporated in the background, the resulting innovation is handled by pROST in the same way as a suddenly appearing foreground object. This phantom object is erroneously classified as foreground (false negative error) until it slowly vanishes again by being incorporated into the background model. This is a model-specific problem and could only be overcome by considering additional information about the texture of the object or by tracking the development of objects in the scene.

### Shadow

As the name of the category suggests, the *shadow* category evaluates whether an algorithm can differentiate between a foreground object and the shadow it casts. The category contains several sequences with static backgrounds and foreground objects, whose shadows should explicitly be detected and classified as a third entity besides foreground and background. Such a distinction is not implemented in *pROST*, so that the resulting segmentation is similar to the *baseline* videos, with the occasional shadows being falsely classified as either foreground or background, depending on their relative intensity compared to the actual background.

### Thermal

Lastly, the *thermal* category contains sequences recorded with an infrared light camera. The grayscale videos are of much lower contrast than the conventional videos, which leads to a comparably poor recall value in this category, i.e. many foreground pixels remain undetected. A simple way to compensate for the low contrast could be to lower the detection threshold, but that again would increase the rate of false positives in all other categories. Furthermore, as the scenes are all static, a step size adjustment as discussed for the *baseline* category could likely decrease the risk of leakage.

### Conclusion

Overall, *pROST* achieves an F-score of about 0.65, whereas other state of the art methods achieve results reaching from about 0.5 to about 0.85, which lands *pROST* at the middle to lower end of the spectrum. It is important to note, however, that it is possible to achieve better performance in every single category by tuning the parameters individually to the specific task. The comparison with *GRASTA* shows that the proposed adjustments for the specific task of background segmentation are highly effective, as the overall F-score is raised by 0.17 points, which is almost half way from *GRASTA*'s performance to the top of the benchmark. The performance of *pROST* is best whenever the input matches the low-rank-and-sparse data model well, i.e. when the background has limited dynamics, and when the foreground objects are actually sparse in space and time. The more these constraints are violated (erratic movements in the background, foreground objects being persistent in the scene for a longer time), the more the performance of *pROST* deteriorates. Compared to competing methods, *pROST* is especially well at learning the dynamic backgrounds caused by camera jitter in steady presence of a large number of foreground objects. A general drawback of the approach is the lack of contextual information, which might alleviate the problems of relocated objects. Furthermore, camouflaging remains an inherent problem even if the joint processing of the color channels improves the performance compared to grayscale image processing. All in all, the *pROST* algorithm proves that Robust Subspace Tracking with a smoothed $\ell_p$-norm cost function is a viable approach on video segmentation, and an efficient implementation on the GPU allows processing videos at a reasonably detailed internal resolution.

# Chapter 6.

# Robust Structured Low-Rank Approximation on the Grassmannian

The robust low-rank approximation methods discussed so far ensure that the rank of the approximation is bounded and that the residual error between the input and the approximation is minimal according to a particular metric. This chapter investigates the case where an additional constraint is considered, which furthermore requires the low-rank approximation to have a pre-defined structure, such as e.g. Hankel or Toeplitz matrices. This chapter contains the derivation of an algorithm for Robust Structured Low-Rank Approximation, which uses the same factorization model with orthogonality constraints and the smoothed $\ell_p$-norm loss function as the methods presented in the previous chapters for the unstructured robust low-rank approximation problems. It is shown how the manifold setting allows to speed up the online analysis of time series via Structured Low-Rank Approximation with Hankel matrices.

## 6.1. Linear matrix structures

In order to work with structured matrices, these structures need to be defined together with instructions on how to construct and describe such matrices, and how to find the closest structured matrix to an unstructured one. An intuitive derivation of these concepts can be found e.g. in the work of Ishteva et al. [48], who propose a structured low-rank approximation method based on the factorization model with an $\ell_2$-based loss function. As their method is closely related to the method proposed here, a similar notation for the structural constraints will be used.

A matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ belongs to the set $\mathcal{S}$ if it follows a certain number of linear structural constraints. That is, there exists a mapping from the $N$ entries of a data vector $\boldsymbol{x} \in \mathbb{R}^N$ to the $m \times n$ entries of the structured matrix, which will be denoted as $\boldsymbol{X}_{\mathcal{S}}$. To describe this mapping, a set of binary $m \times n$ matrices $\{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_N\}$ is defined, with $\boldsymbol{S}_l$ containing ones at the positions where the entry $x_l$ of the data vector appears in the structured matrix $\boldsymbol{X}_{\mathcal{S}}$ and zeros elsewhere, so that the structured matrix can be composed as

$$\boldsymbol{X}_{\mathcal{S}} = \sum_{l=1}^{N} x_i \boldsymbol{S}_l \ . \tag{6.1}$$

By vectorizing and concatenating the structural matrices, the structuring operator

$$\boldsymbol{S} := \begin{bmatrix} \operatorname{vec}(\boldsymbol{S}_1) & \operatorname{vec}(\boldsymbol{S}_2) & \ldots & \operatorname{vec}(\boldsymbol{S}_N) \end{bmatrix} \in \mathbb{R}^{mn \times N} \tag{6.2}$$

is obtained. Let $\operatorname{vec}^{-1}(\cdot)$ denote the inverse process of reordering an $mn$-dimensional vector into an $m \times n$ matrix, then the structuring process in Equation (6.1) can be simplified to writing $\boldsymbol{X}_{\mathcal{S}} = \operatorname{vec}^{-1}(\boldsymbol{S}\boldsymbol{x})$. The knowledge of $\boldsymbol{S}$ furthermore allows to compute the closest structured matrix to an existing one in terms of the Frobenius norm, i.e. to solve

$$\min_{\boldsymbol{X}_{\mathcal{S}} \in \mathcal{S}} \|\boldsymbol{X}_{\mathcal{S}} - \boldsymbol{X}\|_F^2 \tag{6.3}$$

in closed form. As shown in [48], the orthogonal projection $\Pi_{\mathcal{S}}(\boldsymbol{X})$ with respect to the standard inner product can be computed via

$$\Pi_{\mathcal{S}}(\boldsymbol{X}) := \operatorname{vec}^{-1}\left(\boldsymbol{S}\boldsymbol{S}^{\dagger}\operatorname{vec}(\boldsymbol{X})\right) \ . \tag{6.4}$$

The Moore-Penrose pseudoinverse

$$\boldsymbol{S}^{\dagger} := \left(\boldsymbol{S}^{\top}\boldsymbol{S}\right)^{-1} \boldsymbol{S}^{\top} \tag{6.5}$$

of $\boldsymbol{S}$ computes the $l$-th entry of the data vector $\boldsymbol{x}$ of the structured matrix $\boldsymbol{X}_{\mathcal{S}}$ as a least-squares approximation of the particular entries of $\boldsymbol{X}$ on the support of the structural matrix $\boldsymbol{S}_{(l)}$.

## Example: Hankel structure

The introduced concepts shall be illustrated at the example of a Hankel-structured matrix $\boldsymbol{X}_{\mathcal{H}}$ with dimensions $m = n = 2$. As Hankel matrices have identical elements on the antidiagonals, the binary structural matrices constituting $\mathcal{H}$ are

$$\boldsymbol{S}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{S}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \boldsymbol{S}_3 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{6.6}$$

Let $\boldsymbol{S}$ be the structuring operator resulting from vectorizing and stacking $\boldsymbol{S}_1$ through $\boldsymbol{S}_3$ and consider a data vector $\boldsymbol{x} = \begin{bmatrix} a & b & c \end{bmatrix}^\top \in \mathbb{R}^3$. A Hankel matrix $\boldsymbol{X}_{\mathcal{H}}$ can then be constructed as

$$\boldsymbol{X}_{\mathcal{H}} = \operatorname{vec}^{-1}(\boldsymbol{S}\boldsymbol{x}) = \operatorname{vec}^{-1}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}\right) = \operatorname{vec}^{-1}\left(\begin{bmatrix} a \\ b \\ b \\ c \end{bmatrix}\right) = \begin{bmatrix} a & b \\ b & c \end{bmatrix}.$$

To visualize the projection onto $\mathcal{H}$, consider an arbitrary unstructured $2 \times 2$ matrix $\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ with $\operatorname{vec}(\boldsymbol{A}) = \begin{bmatrix} a & b & c & d \end{bmatrix}^\top$, then the data vector of the closest Hankel-structured matrix can be computed as

$$\boldsymbol{x} = \boldsymbol{S}^\dagger \operatorname{vec}(\boldsymbol{A}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ \frac{b+c}{2} \\ d \end{bmatrix}. \tag{6.7}$$

Finally, after multiplying again with the structuring operator and reordering the entries, one obtains the orthogonal projection of $\boldsymbol{X}$ onto the space of Hankel-structured matrices

$$\mathit{\Pi}_{\mathcal{H}}(\boldsymbol{A}) = \operatorname{vec}^{-1}\left(\boldsymbol{S}\boldsymbol{S}^\dagger \operatorname{vec}(\boldsymbol{A})\right) = \begin{bmatrix} a & \frac{b+c}{2} \\ \frac{b+c}{2} & d \end{bmatrix}. \tag{6.8}$$

As a result, the orthogonal projection onto the set of Hankel-structured matrices is equivalent to averaging over the anti-diagonals of a matrix (appearing as the diagonal averaging step in the SSA method [12]).

## 6.2. Alternating minimization framework

As discussed in Chapter 4, the factorization approach for Low-Rank Approximation restricts the search space of possible approximations to those solutions that have an inherent upper bound on the rank. Ishteva et al. [48] investigate this approach in the context of structured low-rank approximation and propose the Structured Low-Rank Approximation by Factorization method (abbreviated in the following as *SLRAbyF*), which searches for the closest structured low-rank approximation in $\ell_2$ sense. As observed among others by Chu et al. [26] and Markovsky [61], there exists no general description of the topology of structured low-rank matrices and thus no viable approach that optimizes directly on the intersection of the two spaces. Using the concepts discussed in the previous section, however, structural constraints can be enforced on any low-rank approximation $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{Y}$ by introducing the structural penalty term

$$\tfrac{1}{mn} \left\| \boldsymbol{U}\boldsymbol{Y} - \mathit{\Pi}_{\mathcal{S}}\left(\boldsymbol{U}\boldsymbol{Y}\right) \right\|_F^2 , \tag{6.9}$$

which penalizes the residual error between a low-rank matrix and its projection onto the space of structured matrices. This residual, which is equivalent to the projection $\mathit{\Pi}_{\mathcal{S}_\perp}(\boldsymbol{U}\boldsymbol{Y})$ onto the orthogonal complement of $\mathcal{S}$, vanishes only if $\boldsymbol{U}\boldsymbol{Y}$ fulfills the structural constraints of $\mathcal{S}$.

While the structural constraint guarantees to find a structured low-rank matrix, a data fitting term is still required to ensure that the found approximation is close to the original data (cf. the discussion of Cadzow's method [16] in [26]). In principle, a separable loss function that sums the entry-wise residual error between input $\boldsymbol{X}$ and structured low-rank approximation $\boldsymbol{L}$ can be employed for this purpose. However, this does not take into account the number of positions at which the entries of the data vector appear in the full structured matrix. Thus, whenever some entries of the data vector appear more often in the structured matrix than others, the data fit is biased towards these entries unless additional weights are introduced. Another point is that whenever the input $\boldsymbol{X}$ is already structured (i.e. $\boldsymbol{X} \in \mathcal{S}$), fitting $\boldsymbol{L}$ to $\boldsymbol{X}$ over the whole coordinate set is unnecessarily more expensive than minimizing the residual error based on the difference $\boldsymbol{x} - \boldsymbol{l}$ of the underlying data vectors. Therefore, the robust loss function (4.8) from Chapter 4 is replaced by

$$h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{x} - \boldsymbol{S}^\dagger \operatorname{vec}(\boldsymbol{UY}) \right) \right), \tag{6.10}$$

which measures the discrepancy between the input data vector $\boldsymbol{x}$ and the least-squares fit to the entries of $\boldsymbol{UY}$, which for $\boldsymbol{UY} \in \mathcal{S}$ is the underlying data vector of the structured low-rank approximation. The residual is evaluated only on the index set $\Omega$ with $|\Omega| \leq N$, where $N$ is the length of the data vector $x$. Ishteva et al. [48] propose to join the two constraints with an Augmented Lagrangian Multiplier (ALM) method [6].

The augmented Lagrangian function of a Robust Structured Low-Rank Approximation problem with the proposed smoothed $\ell_p$-norm loss function writes as

$$\begin{gathered}\mathcal{L}(\boldsymbol{U}, \boldsymbol{Y}, \boldsymbol{\Lambda}) = \\ h_\mu \left( \mathcal{P}_\Omega \left( \boldsymbol{x} - \boldsymbol{S}^\dagger \operatorname{vec}(\boldsymbol{UY}) \right) \right) + \langle \boldsymbol{\Lambda}, \tfrac{1}{mn} (\boldsymbol{UY} - \Pi_\mathcal{S}(\boldsymbol{UY})) \rangle + \tfrac{\rho}{2mn} \| \boldsymbol{UY} - \Pi_\mathcal{S}(\boldsymbol{UY}) \|_F^2 .\end{gathered} \tag{6.11}$$

The general idea of the ALM scheme is to start with a small value for the parameter $\rho$ and to alternate between the optimization problems

$$\min_{[\boldsymbol{U}] \in \mathrm{Gr}_{k,m}} f_U(\boldsymbol{U}), \quad \min_{\boldsymbol{Y} \in \mathbb{R}^{k \times n}} f_Y(\boldsymbol{Y}) \quad \text{and} \quad \min_{\boldsymbol{\Lambda} \in \mathbb{R}^{m \times n}} f_{\boldsymbol{\Lambda}}(\boldsymbol{\Lambda}) \tag{6.12}$$

with the separate cost functions defined as

$$f_{\boldsymbol{U}} : \mathrm{Gr}_{k,m} \to \mathbb{R}, \quad \boldsymbol{U} \mapsto \mathcal{L}(\boldsymbol{U}, \boldsymbol{Y}_0, \boldsymbol{\Lambda}_0), \tag{6.13}$$

$$f_{\boldsymbol{Y}} : \mathbb{R}^{k \times n} \to \mathbb{R}, \quad \boldsymbol{Y} \mapsto \mathcal{L}(\boldsymbol{U}_0, \boldsymbol{Y}, \boldsymbol{\Lambda}_0) \quad \text{and} \tag{6.14}$$

$$f_{\boldsymbol{\Lambda}} : \mathbb{R}^{m \times n} \to \mathbb{R}, \quad \boldsymbol{\Lambda} \mapsto \mathcal{L}(\boldsymbol{U}_0, \boldsymbol{Y}_0, \boldsymbol{\Lambda}), \tag{6.15}$$

respectively, where $\boldsymbol{U}_0, \boldsymbol{Y}_0$ and $\boldsymbol{\Lambda}_0$ describe intermediate estimates for $\boldsymbol{U}, \boldsymbol{Y}$ and $\boldsymbol{\Lambda}$, which are held constant during the optimization of other variables. After each iteration the parameter $\rho$ is increased until the side condition holds up to a certain accuracy. While the simpler penalty method ensures the side condition only for $\rho \to \infty$, the augmented Lagrangian multiplier allows to terminate the algorithm much sooner in practice [6].

The optimization of $f_U$ and $f_Y$ are performed in the same way as for the unstructured low-rank approximation problem. Assuming that the input is fully observed, the respective gradients can be derived as

$$\begin{aligned}\nabla f_U(\boldsymbol{U}) = [&-\operatorname{vec}^{-1}\left((\boldsymbol{S}^\dagger)^\top \nabla h_\mu\left(\boldsymbol{x} - \boldsymbol{S}^\dagger \operatorname{vec}(\boldsymbol{U}\boldsymbol{Y}_0)\right)\right) + \\ &\tfrac{1}{mn}\left((\boldsymbol{\Lambda}_0 - \Pi_S(\boldsymbol{\Lambda}_0)) + \rho(\boldsymbol{U}\boldsymbol{Y}_0 - \Pi_S(\boldsymbol{U}\boldsymbol{Y}_0))\right)]\,\boldsymbol{Y}_0^\top\end{aligned} \tag{6.16}$$

and

$$\begin{aligned}\nabla f_Y(\boldsymbol{Y}) = \boldsymbol{U}_0^\top\,[&-\operatorname{vec}^{-1}\left((\boldsymbol{S}^\dagger)^\top \nabla h_\mu\left(\boldsymbol{x} - \boldsymbol{S}^\dagger \operatorname{vec}(\boldsymbol{U}_0\boldsymbol{Y})\right)\right) + \\ &\tfrac{1}{mn}\left((\boldsymbol{\Lambda}_0 - \Pi_S(\boldsymbol{\Lambda}_0)) + \rho(\boldsymbol{U}_0\boldsymbol{Y} - \Pi_S(\boldsymbol{U}_0\boldsymbol{Y}))\right)]\end{aligned} \tag{6.17}$$

with the full derivation given in Appendix A.2. As for the unstructured case, missing observations appear as a zero entry in the gradient of the loss function.

Algorithm 6.1 outlines the proposed Grassmannian Robust SLRA approach, abbreviated as *GRSLRA*. The algorithm considers a partial observation $\mathcal{P}_\Omega(\boldsymbol{x})$ of the data vector with $\Omega$ denoting the observation set. Besides the data vector, the algorithm requires a description of the structure $\mathcal{S}$. $\boldsymbol{U}$ is initialized randomly, and $\boldsymbol{Y}$ is initialized with all zeros. The weighting factor $\rho$ is initialized sufficiently small (e.g. $\rho = 1$), so that the data fitting term is the dominant term at the beginning of the optimization and the approximation stays close to the input data with respect to the used distance measure. As proposed by Ishteva et al. [48], the optimization consists of an inner loop and an outer loop. In the inner loop, a low-rank approximation $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{Y}$ is found by alternatingly optimizing over $\boldsymbol{U}$ and $\boldsymbol{Y}$ until the process converges to an intermediate solution. Subsequently, the Lagrangian Multiplier is updated with a single update step, $\rho$ is increased and the process is repeated until $\rho$ is large enough to guarantee that the structural side condition holds up to a certain accuracy. The data vector is then obtained via the projection onto the structure $\mathcal{S}$.

Apart from the added structural constraint, the inner low-rank approximation problem differs from the algorithm for the unstructured case in three main aspects: Firstly, the cost function is always evaluated over $\Omega$ as subsampling the line search is neither applicable nor required due to the different nature of the residual error. Secondly, empirical results show that the parameter $\mu$ can be held constant during the approximation as $\rho$ is altered whenever the inner loop converges to an intermediate solution. Thirdly, the criterion for

**Algorithm 6.1** Alternating minimization scheme for Grassmannian Robust SLRA

```
**Input**: $\mathcal{P}_{\Omega}(\boldsymbol{x})$, structural constraints of $\mathcal{S}$
Choose $c_\rho > 1$
Initialize $\boldsymbol{U}_0, \boldsymbol{Y}_0, \rho = \rho_{\text{start}}$
**while** $\rho \leq \rho_{\text{end}}$ **do**
  **while** $\delta > \delta_{\min}$ **do**
    $\boldsymbol{U} \leftarrow \arg \min_{[\boldsymbol{U}] \in \mathrm{Gr}_{k,m}} f_Y(\boldsymbol{Y})$    (6.13)
    $\boldsymbol{Y} \leftarrow \arg \min_{\boldsymbol{Y} \in \mathbb{R}^{k \times n}} f_U(\boldsymbol{U})$    (6.14)
  **end while**
  $\boldsymbol{\Lambda} \leftarrow \boldsymbol{\Lambda} - \frac{\rho}{mn} (\boldsymbol{U}\boldsymbol{Y} - \Pi_{\mathcal{S}}(\boldsymbol{U}\boldsymbol{Y}))$
  $\rho \leftarrow c_\rho \rho$
**end while**
**Outputs:**
$\hat{\boldsymbol{l}} = \boldsymbol{S}^{\dagger} \mathrm{vec}(\boldsymbol{U}\boldsymbol{Y})$
```

convergence of the alternating minimization needs to be modified, as the second term in the Lagrangian funtion (6.11) may lead to a non-monotone decrease of the cost function. Ishteva et al. [48] resolve this issue by observing the progress in the column space of the first factor instead, which is somewhat questionable as the *SLRAbyF* method does not consider orthogonal columns. For the *GRSLRA* method, on the other hand, the subspace angle

$$\delta^{(i+1)} := e_{\text{sub}}\left(\boldsymbol{U}^{(i)}, \boldsymbol{U}^{(i+1)}\right) \tag{6.18}$$

following the definition (4.18) is a meaningful measure due to the imposed orthogonality constraints. Following the recommendation of Ishteva et al. [48], $c_\rho$ is adaptively chosen between $(1.5, 100)$ according to the iteration count of the inner loop, so that the overall number of iterations depends on the convergence speed of the inner loop. Typically, convergence is slow in the beginning (i.e. when the data fitting term dominates) and is fast whenever $\rho$ becomes large. Thus, finding a good initial estimate for $\boldsymbol{U}$ and $\boldsymbol{Y}$ is crucial in order to speed up the algorithm. Certain structures allow to reuse previous estimates for initialization, as will be outlined in the following.

## 6.3. An efficient algorithm for Online Time Series Analysis

Time Series Analysis is closely related to the System Identification problem. Let

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{b}\, u(t), \quad \boldsymbol{y}(t) = \boldsymbol{c}^\top \boldsymbol{x}(t) \tag{6.19}$$

be the state-space model of a Single-Input-Single-Output (SISO) Linear Time Invariant (LTI) system. According to Ho and Kalman [44], the minimum realization of an LTI system of degree $k$ allows to represent the stationary impulse response written as a Hankel-structured matrix as

$$\boldsymbol{Y}_{\mathcal{H}} = \begin{bmatrix} \boldsymbol{c}^\top\boldsymbol{b} & \boldsymbol{c}^\top\boldsymbol{A}\boldsymbol{b} & \boldsymbol{c}^\top\boldsymbol{A}^2\boldsymbol{b} & \dots \\ \boldsymbol{c}^\top\boldsymbol{A}\boldsymbol{b} & \boldsymbol{c}^\top\boldsymbol{A}^2\boldsymbol{b} & \iddots & \\ \boldsymbol{c}^\top\boldsymbol{A}^2\boldsymbol{b} & \iddots & & \\ \vdots & & & \end{bmatrix}. \tag{6.20}$$

Thus, system identification is the task of estimating a low-rank Hankel matrix that explains the observed impulse response. In principle, the mapping

$$\mathbb{R}^{k\times k} \times \mathbb{R}^{k} \times \mathbb{R}^{k} \to \mathcal{H}, \quad (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \mapsto \boldsymbol{Y}_{\mathcal{H}} \tag{6.21}$$

would allow for a direct approach of finding such a low-rank Hankel matrix. But not only would such a model lead to an excessively large number of unknowns. The exponential development of the system matrix $\boldsymbol{A}$ on the main diagonal would render such an approach infeasible in practice, which is why the above discussed approach is to be preferred. Once an LTI system has been identified, the future development of its impulse response can be predicted, which is why the forecasting of arbitrary time series is often formulated as a system identification problem [61]. In an online setting, a time series is not observed and analyzed as a whole but a sliding window of reasonable length is considered, stretching from the current point in time to a fixed number of previous observations. SLRA can be employed to identify the system by finding a low-rank Hankel-structured matrix whose data vector is a close approximation of the given observation window. Considering now two subsequent overlapping observations of a time series, the question is how the two subsequent systems obtained by performing SLRA are related. If the time series is the actual noise-free impulse response of an LTI system, the systems identified from subsequent observations must be identical as the system is known to be time-invariant. Regarding the problem from a factorization point of view, whenever $\boldsymbol{U}$ spans the subspace of the first low-rank Hankel matrix $\boldsymbol{X}_{(j)}$, then the subsequent low-rank Hankel matrix $\boldsymbol{X}_{(j+1)}$ lies in the same subspace, i.e. $\boldsymbol{U}_{(j)} = \boldsymbol{U}_{(j+1)}$. The coordinate vectors (i.e. the columns of $\boldsymbol{Y}$), on the other hand, are shifted by one column to the left, as the next data point only affects the last column of $\boldsymbol{X}_{(j+1)}$ and therefore also of $\boldsymbol{Y}_{(j+1)}$. Assuming that the systems are equal, this last column must lie in the previously estimated subspace and the structural constraint must hold, so that the entries can be filled in based on the previous columns, which renders the approach obsolete if the system is time-invariant and observed without noise. Under real-world conditions, however, subsequently observed impulse responses will lead to slightly different systems due to additive noise. Also, the dynamics of the underlying system may change over time, in which case the system becomes a Linear Time Varying (LTV) system. In this setting, system identification can be used to find an LTI system that explains the time series for that particular observation window, conceptually similar to the linearization of a nonlinear function at a given point. Whenever the observation window is shifted, the subsequent low-rank Hankel matrix will correspond to a different subspace. Yet, due to the large overlap of the data vectors it may be assumed that the subspaces are closely related. Recalling the original motivation of finding a good initialization for the factors $\boldsymbol{U}$ and $\boldsymbol{Y}$ in subsequent SLRA problems, this similarity should be exploited to boost the convergence speed of Algorithm 6.1 in an online setting.

Algorithm 6.2 outlines a method for efficient Online Time Series Analysis using Grassmannian Robust SLRA. The algorithm reads a window of $2m-1$ samples length, the minimum

**Algorithm 6.2** Efficient Online Time Series Analysis using GRSLRA

```
Input: $\mathcal{P}_{\Omega}(\boldsymbol{x})$, Input length $N$, Hankel structure $\mathcal{H}^{m \times m}$
Initialize $\boldsymbol{U}$ randomly and $\boldsymbol{Y} = \mathbf{0}_{k \times m}$
for $j = 2m-2 : N-1$ do
    $\mathcal{P}_{\Omega_{(j+1)}}\left(\boldsymbol{x}_{(j+1)}\right) = \mathcal{P}_{\Omega}\left(\boldsymbol{x}\left[j-2m+3 : j+1\right]\right)$
    Initialize Alg. 6.1 with $\boldsymbol{U}_0 = \boldsymbol{U}_{(j)}$ and $\boldsymbol{Y}_0 = [\boldsymbol{Y}_{(j)}[\,:\,, 2 : m-1] \mid \mathbf{0}_k]$
    Run Alg. 6.1 to obtain $\boldsymbol{U}_{(j+1)}, \boldsymbol{Y}_{(j+1)}$ and $\hat{\boldsymbol{l}}_{(j+1)}$
end for
```

required amount of samples for an $m \times m$-dimensional Hankel matrix. While $\boldsymbol{U}$ is randomly initialized in the first iteration, the *GRSLRA* method is subsequently initialized with the previous estimate for $\boldsymbol{U}$. In a similar way as for the proposed Robust Subspace Tracking method in Chapter 5, the step size for the update of $\boldsymbol{U}$ is restricted to the neighborhood of $\left[\boldsymbol{U}_{(j)}\right]$. This reduces the angle between subsequent subspace estimates, which speeds up the convergence. $\boldsymbol{Y}$ is initialized with a shifted version of the previous estimate, so that the first $m-1$ columns of the initialization $\boldsymbol{Y}_0$ correspond to columns 2 to $m$ of $\boldsymbol{Y}_{(j)}$ and the $m$-th column of $\boldsymbol{Y}_0$ is all zero. In the first iterations of Algorithm 6.1 the parameter $\rho$ is small, making the first term of the Lagrangian function (6.11) the dominant term in the respective cost functions. This can be interpreted as lifting the structural constraint and allows to reach a different structured low-rank matrix in a different subspace. Yet, due to the limitations on the step size for the update of $\boldsymbol{U}$ and the initialization of $\boldsymbol{Y}$, the new solution will be close to the former one and is commonly found within few iterations compared to a random initialization of Algorithm 6.1. In the following, several experiments on simulated and real-world data will be conducted to examine the performance of the *GRSLRA* method and the proposed algorithm for online time series analysis.

## 6.4. Experiments on System Identification and Time Series Forecasting

The first experiment is the identification of a SISO LTI system with the *GRSLRA* method. For this purpose, an impulse response is randomly generated according to the state-space model (6.20) with rank $k = 5$. The vectors $\boldsymbol{b}$ and $\boldsymbol{c} \in \mathbb{R}^5$ are drawn as normalized random vectors and the system matrix $\boldsymbol{A} \in \mathbb{R}^{5\times 5}$ is a random symmetric positive definite matrix with all singular values equal to one, i.e. the system is neither damped nor attenuated. The goal is to identify the system from a sequence of length $N = 80$ samples and to forecast the next $N_f = 20$ samples using the estimated system model. The dimensions of the Hankel matrix are chosen as $m = 20$ and $n = N + N_f - m + 1 = 81$. As before, the rank is assumed to be known. The observation of the impulse response is considered to be noisy and incomplete as $\mathcal{P}_\Omega(\boldsymbol{x})$ with $\boldsymbol{x} := \boldsymbol{y} + \boldsymbol{s} + \boldsymbol{n}$, where $\boldsymbol{y}$ is the actual impulse response of the LTI system, $\boldsymbol{s}$ is a sparse vector of density 0.05 whose nonzero entries are randomly drawn as $\pm 1$ with equal probability. The entries of the noise vector $\boldsymbol{n}$ are i.i.d. according to $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.05$. Only half of the entries are revealed, i.e. $|\Omega| = 40$. The cost function parameter $p = 0.1$ is chosen and $\mu$ is set according to the noise level, following the heuristic (4.7). The convergence threshold for the subspace angle is chosen as $\delta_{\text{min}} = 1°$ for the inner loop of Algorithm 6.1. Whenever more than 3 iterations were required during the inner loop, $c_\rho = 1.5$ is chosen to increase the structural constraint parameter $\rho$, otherwise $c_\rho = 100$. The parameter is varied between $\rho_{\text{start}} = 1$ and $\rho_{\text{end}} = 10^8$. The proposed method is compared against a custom *Python* implementation of the *SLRAbyF* method [48]. The sparse least-squares problems are solved with the `scipy.sparse.linalg.lsqr()` routine, which is about five times faster than solving the same problem in MATLAB using the backslash operator.

The result of the system identification experiment is visualized in Figure 6.1. While the ground truth impulse response is drawn in gray, the actual measurements are visualized with

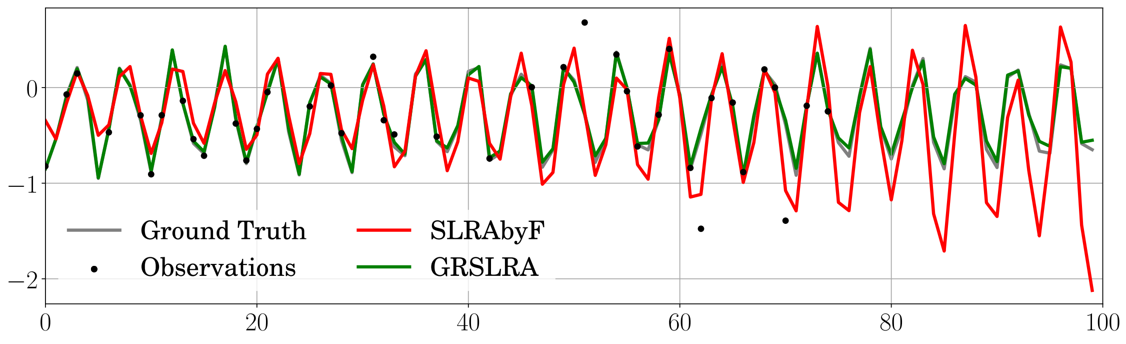


**Figure 6.1.:** Results for the identification of a SISO LTI system. Gray: Noise-free impulse response, Black: Sample points of the impulse response with additive Gaussian noise and sparse outliers at a sampling rate of 0.5 for $t \in [0, 50]$. Red: Low-Rank approximation with *SLRAbyF* ($\ell_2$ data fit). Green: Robust Low-Rank approximation with the proposed *GRSLRA* method.

black dots. The *GRSLRA* method reconstructs and forecasts the impulse response quite well despite the missing observations, the noise and the injected outliers. The reconstruction with *SLRAbyF* [48], on the other hand, overfits against the outlier-contaminated entries, leading to an inaccurate prediction.

### Performance comparison

The presented method is quite similar to the *SLRAbyF* method as the cost function is identical up to the used metric in the data fitting term. As shown by Ishteva et al. [48], the $\ell_2$ distance allows to solve the minimization problems over the two matrix factors in closed form, making the approach quite fast for small dimensions. The system identification task is solved by *SLRAbyF* in about 2.5 seconds, whereas the proposed *GRSLRA* method requires about 10 seconds. For larger dimensions, however, the proposed method scales much better than *SLRAbyF*. Doubling the dimensions of the problem ($m = 40, n = 160, N_f = 40$), the runtime of *GRSLRA* increases by five seconds, whereas *SLRAbyF* now requires 30 seconds. Scaling the dimensions by a factor of 10, *GRSLRA* is still able to find a meaningful approximation within a minute, while the problem becomes impossible to solve in acceptable time with the *SLRAbyF* method. The reason for the inferior scaling is that the structural constraint appears in the optimization problem in form of the $m \times n$ projection matrix $\Pi_S$, which mainly defines the dimension of the least-squares problem. *GRSLRA* also requires the computation of the projection to evaluate the cost function, but performs it much more efficiently by multiplying $\boldsymbol{v} := \boldsymbol{S}^{\dagger}\operatorname{vec}(\boldsymbol{UY})$ first and then multiplying $\boldsymbol{Sv}$ at a cost of $\mathcal{O}(mnN)$ each, which for $N \ll mn$ is much cheaper than executing the full multiplication of complexity $\mathcal{O}(m^2n^2)$. Ishteva et al. [48] explicitly state that the proposed method is suitable for small dimensions only and that an efficient implementation is subject to further research. For large scale problems the apparent benefit of a closed-form solution becomes a drawback compared to a first-order optimization method such as the proposed *GRSLRA* algorithm.

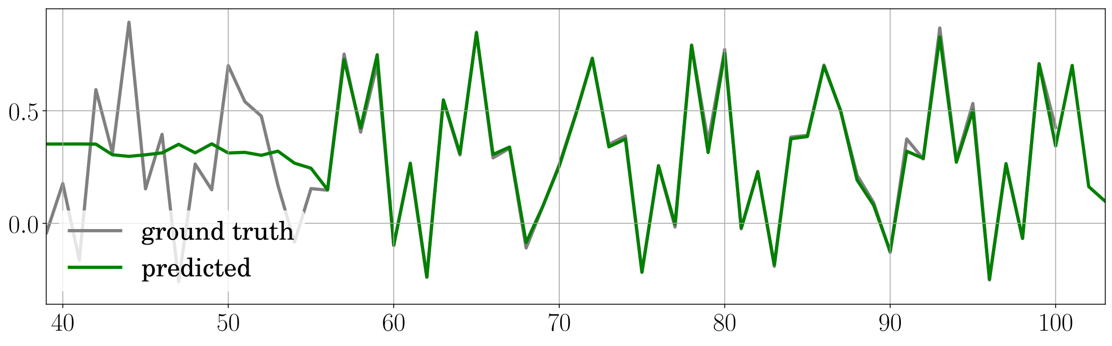
**Online System Identification of LTI / LTV systems**

In order to demonstrate how the *GRSLRA* method can benefit from a good initialization, two experiments on online system identification are carried out. In the first experiment, a noise-free observation of an LTI system's impulse response is considered. Data is created as previously, only that all entries are revealed and the actual impulse response $\boldsymbol{y}$ is observed, so that the smoothing parameter $\mu = 10^{-4}$ is chosen. A sliding window of length $2m-1$ with $m = 20$ is considered and the goal is to forecast the system's impulse response three samples into the future. Figure 6.2a shows the actual impulse response as well as the prediction, from which only the furthest predicted sample is drawn. The angle between subsequent subspace estimates $U_{(j)}$ and $\boldsymbol{U}_{(j+1)}$ is drawn in Figure 6.2b, as to visualize the progress of learning the subspace over the course of several approximations. The algorithm requires some time to identify the system, which leads to incorrect predictions of the impulse response in the beginning. Once the system is identified, however, the impulse response is correctly predicted up to some numerical fluctuations, and the subspace remains almost constant. The initial iterations take up to one second, while as soon as the system is identified the processing time per iteration goes down to 0.3 seconds.

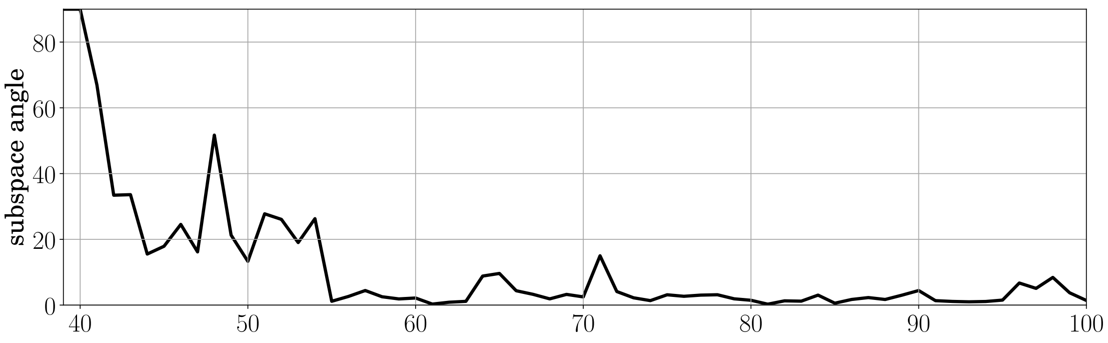
In a second experiment, the impulse response of a Linear Time Varying (LTV) system is considered. The data is generated according to the state space model

$$\begin{aligned} \dot{\boldsymbol{x}}_{(j+1)} &= \boldsymbol{A}_{(j)}\boldsymbol{x}_{(j)} + \boldsymbol{b}^{\top}\boldsymbol{u}_{(j)} \\ \boldsymbol{y}_{(j)} &= \boldsymbol{c}^{\top}\boldsymbol{x}_{(j)} \end{aligned} \tag{6.22}$$

with $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^k$ being normalized random Gaussian vectors and $\boldsymbol{A}_{(j)} \in \mathbb{R}^{k\times k}$ being a temporally varying system matrix that fulfills the differential equation $\dot{\boldsymbol{A}} = \boldsymbol{ZA}$ with $\boldsymbol{Z}^{\top} = -\boldsymbol{Z}$

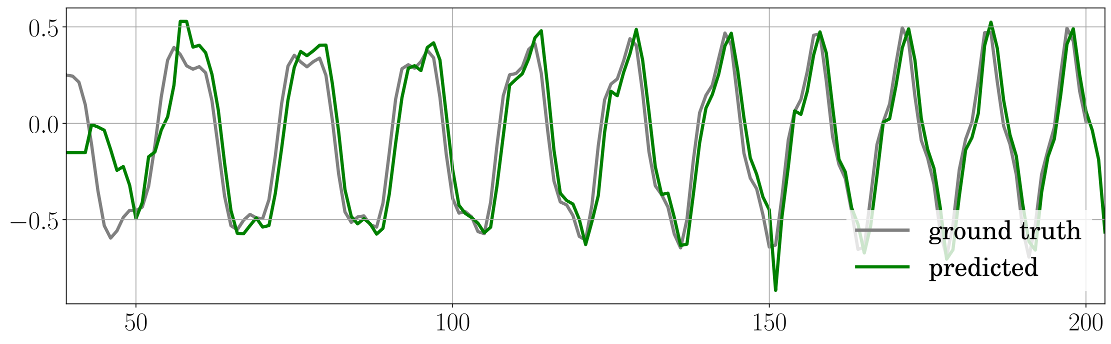


**(a)** Actual impulse response and prediction three samples into the future

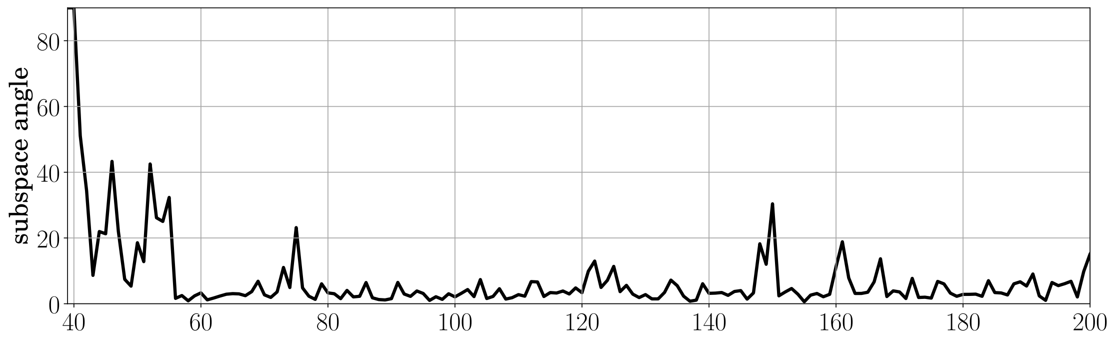


**(b)** Progression of the angular subspace error

**Figure 6.2.:** Online identification of a Single-Input-Single-Output Linear Time Invariant system



**(a)** Actual impulse response and prediction three samples into the future



**(b)** Progression of the angular subspace error

**Figure 6.3.:** Online identification of a Single-Input-Single-Output Linear Time Varying system

and the initial condition $\boldsymbol{A}_{(0)} = \boldsymbol{I}_k$. For a better visualization of the experiment the temporal scale is changed and the beginning of the observation is shifted by setting

$$\boldsymbol{A}_{(j)} = e^{0.002(j+200)\,\boldsymbol{Z}}. \tag{6.23}$$

The degree of the system is chosen as $k = 5$ and the first 200 samples of the impulse response are considered. The parameters are chosen as for the previous task and the results are displayed in Figure 6.3a and Figure 6.3b, respectively. As before, the algorithm requires several iterations to identify the system and to make a meaningful prediction of the impulse response. In contrast to the LTI system, the dynamics of the LTV system vary over time, so that the LTI approximation needs to be adapted slightly, even after a meaningful approximation has been found. This can be seen from the repetitive spikes in the evolution of the subspace angle. The runtimes are in the same range as for the LTI experiment, which shows that the initialization with a previously found solution successfully reduces the required number of iterations of *GRSLRA*, even if the underlying dynamics of the time series require constant updates of the subspace.

#### Time Series Forecasting on Real-World Data

In order to investigate the performance on real-world data, the proposed online time series forecasting method is evaluated on a time series forecasting task. It is compared with *SLRAbyF* and the *forecast* routine in MATLAB with a 12 month seasonal ARIMA(0,1,1) model[1]. The time series is the well-known *Airline Passenger* data set from [11], which contains the development of airline passenger counts in the USA in the years between 1949 and 1960. The upper bound on the rank of the approximation is chosen as $k = 8$, and the goal is to forecast 6 samples from $2m - 1$ samples with $m = 18$, which corresponds to projecting the monthly amount of passengers half a year into the future from observing the past three years. The cost function parameters for the *GRSRLA* method are selected as $p = 0.1$, allowing for high robustness against large outliers (despite not being present in this data set) and a moderate value of $\mu = 0.03$ for the smoothing parameter, as real-world data is obviously contaminated with a certain level of noise. The forecasting results are visualized in Figure 6.4. In general, all three methods perform quite well in this task, with *ARIMA* reaching the best accuracy at a relative error of 0.07, *SLRAbyF* following in second place with a relative error of 0.12 and *GRSLRA* finishing in the same range as *SLRAbyF* with a relative error of 0.14. Considering the run times, *SLRAbyF* is the slowest method with an average processing time of 1.7 seconds per sample, while *ARIMA* is twice as fast with 0.8 seconds on average. Finally, *GRSLRA* is the fastest with about 0.4 seconds on average, which proves the high efficiency of the method.

To show that the proposed robust cost function is beneficial on a data set that is less well-behaving than the classical one, a second experiment on airline passenger data is considered. For this purpose, the system-wide (domestic and international) number of passenger enplanements in the USA for the years $1996 - 2014$ has been obtained from the American Bureau of Transportation Statistics [14]. This data is obviously more challenging, as it

[1]`http://mathworks.com/help/econ/forecast-airline-passenger-counts.html`

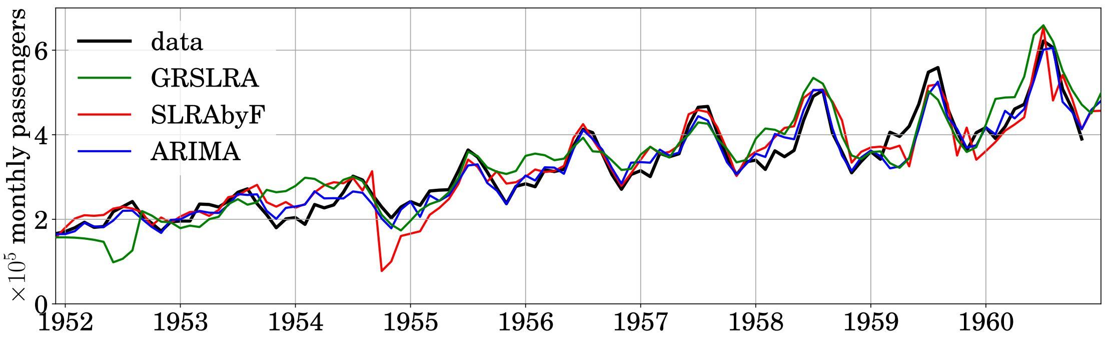


**Figure 6.4.:** Six-month forecast of monthly airline passenger data from the years 1952-1960 based on a 3-year observation period. Actual data and predictions using *GRSLRA*, *SLRAbyF* and a seasonal *ARIMA* model, respectively

contains an unexpected drop in passenger counts after the 9/11 events. Figure 6.5 shows the data set and the six month forecasts of the three compared methods with the same experimental setup and choice of parameters as before. Again, the *ARIMA* forecasting routing shows a good performance. Despite the error of predicting a second post-9/11 drop in the subsequent year, it reaches an overall relative error of 0.08. The *SLRAbyF* method is even more severely affected by the unexpected notch, which is likely due to the non-robust $\ell_2$ measure. The prediction for the years 2002 and 2003 become highly unstable, leading to an overall relative error of 0.16. Due to the more robust loss function, the *GRSLRA* method suffers less from the outliers than *SLRAbyF* and reaches an overall relative error of only 0.08, just like *ARIMA*. The computation times are similar to the previous experiment with *GRSLRA* requiring an average of 0.5 seconds, *ARIMA* being slightly faster than previously with 0.6 seconds and *SLRAbyF* trailing the comparison with 1.5 seconds per prediction.

Overall, the accuracy of SLRA-type methods is worse or equal than *ARIMA*, which however requires an initial estimate of the seasonality. It can be seen that the proposed *GRSLRA* approach performs equally well as the $\ell_2$-based *SLRAbyF* method whenever no outliers are present. In the presence of outliers, however, the proposed method demonstrates increased robustness, even when the assumptions of the low-rank-and-sparse data model are not exactly met. The proposed algorithm for online time series analysis benefits from an efficient initialization with previously found subspace and coordinate estimates. As a result,

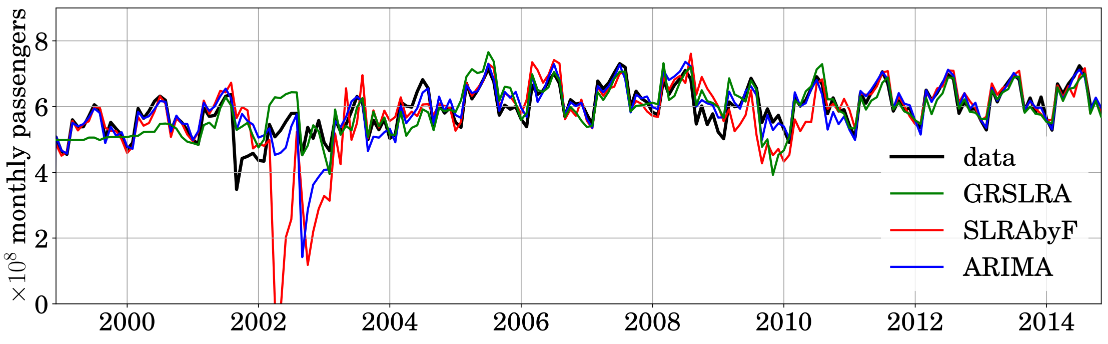


**Figure 6.5.:** Six-month forecast of monthly airline passenger data from the years 1999-2014 based on a 3-year observation period. Actual data and predictions using *GRSLRA*, *SLRAbyF* and a seasonal *ARIMA* model, respectively

it operates much faster than repeated estimation with *SLRAbyF* and slightly faster than the standard forecasting routine with a seasonal *ARIMA* model in MATLAB.

# Chapter 7.

# Conclusion

This thesis deals with three problems and related applications that are based on the low-rank-and-sparse data model. Namely, three non-convex approaches to Robust PCA, Robust Subspace Tracking and Robust Structured Low-Rank Approximation are presented. The collected results indicate that the methods achieve better results than related convex methods, both in terms of computational and memory efficiency and concerning robustness against deviations of the data from the ideal low-rank and sparsity assumptions. To address the low-rankness, the algorithms discussed in this thesis employ the factorization approach with orthogonality constraints on one of the factors. This regularizes the problem in terms of scaling ambiguities and attainability of the upper bound of the rank. Furthermore, it offers an intuitive model with one matrix representing the basis of the low-dimensional subspace to be found and the other one the coordinates within this subspace. The resulting geometry of the problem has been discussed and first-order line search methods for solving optimization problems on the Grassmannian have been developed. In order to reduce the computational effort of the cost function evaluation, an efficient subsampling heuristic has been proposed that is applicable to separable sparsifying functions. Sparsity is enforced on the residual error with an effective smoothed non-convex $\ell_0$-surrogate loss function, which is based on the extension of the $\ell_p$ norm to the case $0 < p < 1$, with the parameter $p$ controlling the slope of the cost function. The function approximates the behavior of the $\ell_0$ norm for large inputs while the introduction of a smoothing parameter $\mu$ makes the function differentiable and convex in a small region around zero, leading to a certain robustness against additive Gaussian noise. A heuristic for choosing an appropriate value for $\mu$ is provided, which depends on the value for the cost function parameter $p$ and the estimated noise level in the data.

Using a differentiable loss function, a Grassmannian Robust PCA algorithm has been presented to find a meaningful low-rank approximation in the presence of large outliers in the data. The optimization is performed with an alternating minimization scheme over the two matrix factors using a conjugate gradient method without additional slack variables on the Grassmannian and in Euclidean space, respectively. For a small choice of the cost function parameter $p$, the method is able to reconstruct an underlying subspace in the presence of outliers even in scenarios where other state-of-the-art methods fail because the low-rank and/or the sparsity assumption only hold to a certain extent. In contrast to $\ell_1$-based factorization methods, the reconstruction performance is not breaking down when the outlier magnitude becomes overly large. The algorithm is able to reconstruct subspaces from incomplete observations, even though the advantage of the non-convex sparsity measure seems to be less significant at low sampling rates. The proposed cost-efficient subsampled line search and the inherent memory efficiency of the factorization approach let the algorithm scale well to large dimensions and allow for low processing times. This is demonstrated in a real-world video segmentation experiment, where the task is to determine a dynamic background model and to extract foreground objects via background subtraction. While the algorithm faster than competing methods, the proposed loss function is shown to be useful at suppressing ghosting artifacts.

For the online setting, a Robust Subspace Tracking algorithm has been proposed that uses the same concepts as in the batch case but learns a subspace incrementally. For every new sample, a single small gradient step on the Grassmannian is taken in order to allow the subspace to slowly evolve over time, while the coordinates are subsequently found with a CG method. Experimental results show that the general findings of the previous evaluation for the static scenario carry over to the tracking case. Yet, it is observed that choosing a small value for the cost function parameter $p$ slows down the convergence speed of the tracking method. As a consequence, a trade-off needs to be found between maximum sparsity enforcement and convergence speed. The presented *pROST* method is an adaptation of the Robust Subspace Tracking algorithm that is specialized to the task of video segmentation. Through its efficient implementation with GPU acceleration, the method achieves realtime segmentation at a reasonable internal resolution. The evaluation on a public benchmark demonstrates that the method is capable of learning and maintaining dynamic backgrounds in the steady presence of foreground objects. Yet, the general limitations of a purely spatio-temporal approach become obvious as some tasks cannot be solved well without additional contextual information, and the varying time scales of foreground objects throughout the categories make it difficult to determine one parameter setting that fits all scenarios.

Furthermore, a robustification of the factorization-based Structured Low-Rank Approximation problem is addressed. The Grassmannian Structured Low-Rank Approximation algorithm builds upon an existing Lagrangian multiplier scheme but measures the residual error with a smoothed $\ell_p$ norm loss function instead of the Euclidean distance in order to make the approach robust against outliers in the data. Furthermore, the proposed framework with a CG algorithm lets the algorithm scale better to large dimensions than the existing framework based on least-squares optimization. Experimental results show that the method is capable of identifying an LTI system from a subsampled noisy and outlier-contaminated observation of its impulse response. Finally, an efficient online method for the prominent application of time series analysis is presented and evaluated, which gains enhanced efficiency compared to repeated re-initialization by reusing previously determined low-rank Hankel approximations. A comparison with an $\ell_2$-based method and a MATLAB forecasting routine on real-world time series demonstrates the practical applicability of the proposed method and its tolerance against large outliers in the data.

# Bibliography

1. P.A. Absil, C.G. Baker, and K.A. Gallivan. Trust-region methods on Riemannian manifolds. In *Foundations of Computational Mathematics*, 7(3), pp. 303–330, 2007.

2. P.A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds.* Princeton University Press, Princeton, NJ, 2008.

3. M. Ayazoglu, M. Sznaier, and O.I. Camps. Fast algorithms for structured robust principal component analysis. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1704–1711. IEEE, 2012.

4. L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In *Allerton Conference on Communication, Control, and Computing*, pp. 704–711. 2010.

5. R.M. Bell and Y. Koren. Lessons from the Netflix prize challenge. In *ACM SIGKDD Explorations Newsletter*, 9(2), pp. 75–79, 2007.

6. D.P. Bertsekas. *Constrained optimization and Lagrange multiplier methods.* New York: Academic Press, 1982.

7. A. Björck and G.H. Golub. Numerical methods for computing angles between linear subspaces. In *Mathematics of computation*, 27(123), pp. 579–594, 1973.

8. N. Boumal and P.A. Absil. Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. In *Linear Algebra and its Applications*, 475, pp. 200–239, 2015.

9. N. Boumal and P.A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In *Advances in neural information processing systems*, pp. 406–414. 2011.

10. T. Bouwmans. Subspace learning for background modeling: A survey. In *Recent Patents on Computer Science*, 2(3), pp. 223–234, 2009.

11. G.E.P. Box and G.M. Jenkins. *Time series analysis: forecasting and control, revised ed.* Holden-Day, 1976.

12. D.S. Broomhead and G.P. King. Extracting qualitative dynamics from experimental data. In *Physica D: Nonlinear Phenomena*, 20(2), pp. 217–236, 1986.

13. S. Brutzer, B. Höferlin, and G. Heidemann. Evaluation of Background Subtraction Techniques for Video Surveillance. In *Computer Vision and Pattern Recognition*, pp. 1937–1944. IEEE, 2011.

14. Bureau of Transportation Statistics, United States Department of Transportation. US Air Carrier travel statistics. 2015. URL `http://www.rita.dot.gov/bts`.

15. R. Cabral, F. De la Torre, J.P. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 2488–2495. IEEE, 2013.

16. J.A. Cadzow. Signal enhancement-a composite property mapping algorithm. In *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(1), pp. 49–62, 1988.

17. J. Cai, E.J. Candès, and Z. Shen. A Singular Value Thresholding Algorithm for Matrix Completion. In *SIAM J. on Optimization*, 20, pp. 1956–1982, 2010.

18. L. Cambier and P.A. Absil. Robust low-rank matrix completion via Riemannian optimization. In *SIAM J. on Scientific Computing*, to appear.

19. E. Candès, X. Li, Y. Ma, and J. Wright. Robust Principal Component Analysis? In *Journal of ACM*, 58(3), pp. 1–37, 2011.

20. E.J. Candès and Y. Plan. Matrix completion with noise. In *Proceedings of the IEEE*, 98(6), pp. 925–936, 2010.

21. E.J. Candès and B. Recht. Exact matrix completion via convex optimization. In *Foundations of Computational mathematics*, 9(6), pp. 717–772, 2009.

22. E.J. Candès, M.B. Wakin, and S.P. Boyd. Enhancing sparsity by reweighted $\ell_1$ minimization. In *Journal of Fourier analysis and applications*, 14(5-6), pp. 877–905, 2008.

23. V. Chandrasekaran, S. Sanghavi, P.A. Parrilo, and A.S. Willsky. Rank-sparsity incoherence for matrix decomposition. In *SIAM Journal on Optimization*, 21(2), pp. 572–596, 2011.

24. Y. Chen, H. Xu, C. Caramanis, and S. Sanghavi. Robust Matrix Completion and Corrupted Columns. In *International Conference on Machine Learning*, volume 2, pp. 873–880. 2011.

25. Y. Chi, Y.C. Eldar, and R. Calderbank. PETRELS: Subspace estimation and tracking from partial observations. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 3301–3304. IEEE, 2012.

26. M.T. Chu, R.E. Funderlic, and R.J. Plemmons. Structured low rank approximation. In *Linear Algebra and Its Applications*, 366, pp. 157–172, 2003.

27. P. Comon and G.H. Golub. Tracking a few extreme singular values and vectors in signal processing. In *Proceedings of the IEEE*, 78(8), pp. 1327–1343, 1990.

28. W. Dai and O. Milenkovic. SET: an algorithm for consistent matrix completion. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 3646–3649. IEEE, 2010.

29. C. Ding, D. Zhou, X. He, and H. Zha. R1-PCA: Rotational invariant L1-norm Principal Component Analysis for robust subspace factorization. In *23rd international conference on Machine learning*, pp. 281–288. ACM, New York, NY, USA, 2006.

30. D.L. Donoho. Compressed Sensing. In *IEEE Transactions on Information Theory*, 52(4), pp. 1289–1306, 2006.

31. D.L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via $\ell_1$ minimization. In *Proceedings of the National Academy of Sciences*, 100(5), pp. 2197–2202, 2003.

32. A. Edelman, T.A. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. In *SIAM Journal on Matrix Analysis and Applications*, 20(2), pp. 303–353, 1998.

33. J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. In *Journal of the American statistical Association*, 96(456), pp. 1348–1360, 2001.

34. D. Gabay. Minimizing a differentiable function over a differential manifold. In *Journal of Optimization Theory and Applications*, 37(2), pp. 177–219, 1982.

35. G. Gasso, A. Rakotomamonjy, and S. Canu. Recovering Sparse Signals With a Certain Family of Nonconvex Penalties and DC Programming. In *IEEE Transactions on Signal Processing*, 57(12), pp. 4686 –4698, dec. 2009.

36. H. Golub and C.F. Van Loan. *Matrix computations.* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

37. N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. changedetection.net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops*, pp. 1 –8. june 2012.

38. C. Hage and M. Kleinsteuber. Robust PCA and subspace tracking from incomplete observations using $\ell_0$-surrogates. In *Computational Statistics*, pp. 1–21, 2014.

39. C. Hage, F. Seidel, and M. Kleinsteuber. GPU Implementation for Background-Foreground-Separation via Robust PCA and Robust Subspace Tracking. In *Background Modeling and Foreground Detection for Video Surveillance.* Chapman and Hall / CRC Press, 2014.

40. C. Hage and M. Kleinsteuber. Robust Structured Low-Rank Approximation on the Grassmannian. In *International Conference on Latent Variable Analysis and Signal Separation*, pp. 295–303. Springer, 2015.

41. N. Halko, P.G. Martinsson, and J.A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. In *SIAM review*, 53(2), pp. 217–288, 2011.

42. J. He, L. Balzano, and A. Szlam. Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1568–1575. 2012.

43. M.R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. In *Journal of Research of the National Bureau of Standards*, 49, pp. 409–436, December 1952.

44. B. Ho and R.E. Kalman. Effective construction of linear state-variable models from input/output functions. In *Regelungstechnik*, 14(1-12), pp. 545–548, 1966.

45. P.J. Huber et al.. Robust estimation of a location parameter. In *The Annals of Mathematical Statistics*, 35(1), pp. 73–101, 1964.

46. M. Hubert, P.J. Rousseeuw, and K. Vanden Branden. ROBPCA: a new approach to Robust Principal Component Analysis. In *Technometrics*, 47(1), pp. 64–79, 2005.

47. K. Hüper, M. Kleinsteuber, and H. Shen. Averaging complex subspaces via a Karcher mean approach. In *Signal Process.*, 93(2), pp. 459–467, February 2013.

48. M. Ishteva, K. Usevich, and I. Markovsky. Factorization approach to structured low-rank approximation with applications. In *SIAM Journal on Matrix Analysis and Applications*, 35(3), pp. 1180–1204, 2014.

49. P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 665–674. ACM, 2013.

50. J. Karhunen and J. Joutsensalo. Learning of robust principal component subspace. In *Neural Networks, 1993. IJCNN'93-Nagoya. Proceedings of 1993 International Joint Conference on*, volume 3, pp. 2409–2412. IEEE, 1993.

51. R.H. Keshavan and A. Montanari. Matrix completion from noisy entries. In *The Journal of Machine Learning Research*, 11, pp. 2057–2078, 2010.

52. R.H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. In *Information Theory, IEEE Transactions on*, 56(6), pp. 2980–2998, 2010.

53. R.H. Keshavan and S. Oh. A gradient descent algorithm on the Grassman manifold for matrix completion. In *arXiv preprint arXiv:0910.5260*, 2009.

54. Y. Koren. The BellKor solution to the netflix grand prize. In *Netflix prize documentation*, 81, 2009.

55. N. Kwak. Principal Component Analysis based on $\ell_1$-norm maximization. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9), pp. 1672–80, 2008.

56. R.M. Leahy and B.D. Jeffs. On the design of maximally sparse beamforming arrays. In *Antennas and Propagation, IEEE Transactions on*, 39(8), pp. 1178–1187, 1991.

57. K. Lee and Y. Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. In *Information Theory, IEEE Transactions on*, 56(9), pp. 4402–4416, 2010.

58. L. Li, W. Huang, I.Y.H. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. In *Image Processing, IEEE Transactions on*, 13(11), pp. 1459 –1472, nov. 2004.

59. Y. Li. On incremental and robust subspace learning. In *Pattern Recognition*, 37, pp. 1509–1518, 2004.

60. Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. In *Arxiv preprint arXiv:1009.5055*, 2010.

61. I. Markovsky. Structured low-rank approximation and its applications. In *Automatica*, 44(4), pp. 891–909, 2008.

62. I. Markovsky. *Low-Rank Approximation - Algorithms, Implementation, Applications.* Springer, 2014.

63. I. Markovsky and K. Usevich. Structured low-rank approximation with missing data. In *SIAM Journal on Matrix Analysis and Applications*, 34(2), pp. 814–830, 2013.

64. G. Mateos and G.B. Giannakis. Robust PCA as bilinear decomposition with outlier-sparsity regularization. In *Signal Processing, IEEE Transactions on*, 60(10), pp. 5176–5190, 2012.

65. G. Meyer, S. Bonnabel, and R. Sepulchre. Linear regression under fixed-rank constraints: A Riemannian approach. In *International Conference on Machine Learning*, pp. 545–552. 2011.

66. D. Needell and J.A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. In *Applied and Computational Harmonic Analysis*, 26(3), pp. 301–321, 2009.

67. J. Nocedal and S. Wright. Numerical optimization. In *Springer, New York*, 2006.

68. N. Oliver, B. Rosario, and A. Pentland. A Bayesian computer vision system for modeling human interactions . In *Transactions on Pattern Analysis and Machine Intelligence*, 22(8), pp. 831 –843, aug 2000.

69. K. Pearson. On lines and planes of closest fit to systems of points in space. In *Philosophical Magazine*, 2(6), pp. 559–572, 1901.

70. C. Qiu and N. Vaswani. Reprocs: A missing link between recursive Robust PCA and recursive sparse recovery in large but correlated noise. In *arXiv preprint arXiv:1106.3286*, 2011.

71. B. Recht, M. Fazel, and P.A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. In *SIAM review*, 52(3), pp. 471–501, 2010.

72. F. Seidel, C. Hage, and M. Kleinsteuber. pROST : A Smoothed $\ell_p$-norm Robust Online Subspace Tracking Method for Realtime Background Subtraction in Video. In *Machine Vision and Applications*, 25(5), pp. 1227–1240, July 2014.

73. S. Shalev-Shwartz, A. Gonen, and O. Shamir. Large-Scale Convex Minimization with a Low-Rank Constraint. In *International Conference on Machine Learning*, pp. 329–336. 2011.

74. U. Shalit, D. Weinshall, and G. Chechik. Online learning in the manifold of low-rank matrices. In *Advances in Neural Information Processing Systems*, 23, pp. 2128–2136, 2010.

75. Y. Shen, Z. Wen, and Y. Zhang. Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization. In *Optimization Methods and Software*, 29(2), pp. 239–263, 2014.

76. J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 1994.

77. F. De la Torre and M.J. Black. Robust principal component analysis for computer vision. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pp. 362–369. IEEE, 2001.

78. K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and Practice of Background Maintenance. In *International Conference on Computer Vision*, 1, pp. 255–261, 1999.

79. B. Vandereycken. Low-rank matrix completion by Riemannian optimization. In *SIAM Journal on Optimization*, 23(2), pp. 1214–1236, 2013.

80. A.E. Waters, A.C. Sankaranarayanan, and R.G. Baraniuk. SpaRCS: Recovering low-rank and sparse matrices from compressive measurements. In *Advances in Neural Information Processing Systems*, pp. 1089–1097. 2011.

81. J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices via Convex Optimization. In *Advances in Neural Information Processing Systems*, pp. 2080–2088. 2009.

82. Z. Xu, P. Shi, and I.Y.H. Gu. An Eigenbackground Subtraction Method Using Recursive Error Compensation. In Y. Zhuang, S. Yang, Y. Rui, and Q. He (eds.), *PCM*, volume 4261 of *Lecture Notes in Computer Science*, pp. 779–787. 2006.

83. B. Yang. Projection approximation subspace tracking. In *Signal Processing, IEEE Transactions on*, 43(1), pp. 95–107, 1995.

84. X. Yuan and J. Yang. Sparse and low-rank matrix decomposition via alternating direction methods. In *preprint*, 12, 2009.

85. T. Zhou and D. Tao. GoDec: Randomized low-rank & sparse matrix decomposition in noisy case. In *International Conference on Machine Learning*, pp. 33–40. 2011.

# Appendix A.

## A.1. Subspaces and Norms

### A.1.1. Vector spaces and subspaces

Considering a set of data samples with $m$ features each, every sample is a vector within the vector space $\mathbb{R}^m$, which constitutes itself by axioms deducted from the concepts of vector addition and scalar multiplication. A set $\mathcal{V}$ of vectors in $\mathbb{R}^m$ is a linear *subspace* within the vector space $\mathbb{R}^m$ if the following conditions are satisfied:

- $\mathcal{V}$ contains the all-zero vector $\mathbf{o}_m$
- If $\boldsymbol{a} \in \mathcal{V}$ then $c\boldsymbol{a} \in \mathcal{V}$ for all $c \in \mathbb{R}$ (closure under multiplication)
- if $a, b \in \mathcal{V}$ then $a + b \in \mathcal{V}$ (closure under addition)

A *basis* of a linear subspace $\mathcal{V}$ is the smallest possible set of vectors that span this subspace. Vice versa, given a matrix $\boldsymbol{U}$ the subspace $\mathcal{V} = \operatorname{span}(\boldsymbol{U})$ is the set of all vectors that lie in its column space. If $\boldsymbol{U}$ contains $k$ orthogonal columns of length $m$, then $\mathcal{V}$ is a $k$-dimensional subspace in $m$-dimensional surrounding space.

### A.1.2. Vector and matrix norms

Given a vector space $V$, a norm describes a mapping $g : V \to \mathbb{R}$, i.e. every element $\boldsymbol{x} \in V$ is assigned a real number. While a *seminorm* is constituted by the two conditions

$$g(\alpha\boldsymbol{x}) = |\alpha|\ g(\boldsymbol{x}) \quad \text{(homogeneity)} \tag{A.1}$$

$$g(\boldsymbol{x} + \boldsymbol{y}) \leq g(\boldsymbol{x}) + g(\boldsymbol{y}) \quad \text{(triangle inequality)} \tag{A.2}$$

for vectors $\boldsymbol{x}, \boldsymbol{y} \in V$ and scalar $\alpha$, a *norm* also fulfills the third condition

$$g(\boldsymbol{x}) = 0 \quad \text{if and only if} \quad \boldsymbol{x} = 0 \quad \text{(separation).} \tag{A.3}$$

The $\ell_p$ norm of a vector $\boldsymbol{x} \in \mathbb{R}^m$ is defined as

$$\|\boldsymbol{x}\|_p^p := \sum_{i=1}^{m} |x_i|^p, \quad p > 0 \tag{A.4}$$

and it fulfills all three necessary conditions of a norm for $p \geq 1$.

A well-known and intuitively used norm is the *Euclidean norm*

$$\|\boldsymbol{x}\|_2^2 := \sum_{i=1}^{m} x_i^2, \tag{A.5}$$

which is also referred to as the $\ell_2$ norm as it is the special case $k = 2$ of the $\ell_p$ norm (A.4). Analogously, one can derive the $\ell_1$ norm

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{m} |x_i|, \tag{A.6}$$

as the special case $k = 1$, which is also known as the *Manhattan norm.*

Whenever $p$ is in the range $0 < p < 1$, the measure becomes concave and is not a true norm any longer, as the triangle inequality (A.2) does not hold.

Another measure that has gained a lot of attention in the context of Compressive Sensing [31] is the $\ell_0$ norm

$$\|\boldsymbol{x}\|_0 = \text{"number of nonzero entries in } \boldsymbol{x}\text{"}. \tag{A.7}$$

The norm can be interpreted as the limit $p \to 0$ for (A.4), where every entry of $\boldsymbol{x}$ contributes to the sum with a 1 except when the entry is exactly zero. Clearly, the $\ell_0$ norm fulfills neither (A.1) nor (A.2) and is thus not a proper norm, which is why it is often referred to as a *pseudo norm* in the literature.

All discussed vector norms can be extended to entry-wise matrix norms. The entry-wise

$\ell_p$ norm for matrices writes as

$$\|\boldsymbol{X}\|_p^p := \sum_{i=1}^{m}\sum_{j=1}^{n} |X_{ij}|^p, \quad p > 0 \tag{A.8}$$

from which the other entry-wise matrix norms can straightforwardly be deducted. It needs to be remarked that the entry-wise $\ell_2$ norm obtained by summing over all squared entries of a matrix $\boldsymbol{X}$ is identified as the squared Frobenius norm $\|\boldsymbol{X}\|_F^2$, and it must not be confused with the spectral norm $\|\boldsymbol{X}\|_2$, which measures the maximum singular value of $\boldsymbol{X}$.

Finally, the *nuclear norm*

$$\|\boldsymbol{X}\|_* = \sum_{i=1}^{n} \sigma_i(\boldsymbol{X}) \tag{A.9}$$

of a matrix $\boldsymbol{X}$ denotes the sum over its singular values. It often appears in the context of rank-minimization problems.

## A.2. Gradient derivation for the Robust SLRA problem

The cost functions of the *GRSLRA* algorithm in Chapter 6 result from minimizing the Lagrangian (6.11) over the respective parameters. The separate cost functions are obtained by considering the Lagrangian function with the other variables fixed. Assuming w.l.o.g. that the data is fully observed, the cost function $f_U$ can be stated as

$$\begin{aligned}
f_U(\boldsymbol{U}) &= h_\mu\left(\boldsymbol{x} - \boldsymbol{S}^\dagger \operatorname{vec}(\boldsymbol{U}\boldsymbol{Y}_0)\right) + \tfrac{1}{mn}\operatorname{vec}(\boldsymbol{\Lambda}_0)^\top \left(\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\operatorname{vec}(\boldsymbol{U}\boldsymbol{Y}_0)\right) \\
&\quad + \tfrac{\rho}{2mn}\left\|\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\operatorname{vec}(\boldsymbol{U}\boldsymbol{Y}_0)\right\|_2^2 \\
&= h_\mu\left(\boldsymbol{x} - \boldsymbol{S}^\dagger\left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right)\operatorname{vec}(\boldsymbol{U})\right) + \tfrac{1}{mn}\operatorname{vec}(\boldsymbol{\Lambda}_0)^\top \left(\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right)\operatorname{vec}(\boldsymbol{U})\right) \\
&\quad + \tfrac{\rho}{2mn}\left\|\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right)\operatorname{vec}(\boldsymbol{U})\right\|_2^2 .
\end{aligned}$$

Let

$$\boldsymbol{A}_Y := \left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right).$$

Using the definition $(\boldsymbol{A} \otimes \boldsymbol{B})^\top = \boldsymbol{A}^\top \otimes \boldsymbol{B}^\top$ for the transpose of a Kronecker product, the transpose is obtained as

$$\boldsymbol{A}_Y^\top = (\boldsymbol{Y}_0 \otimes \boldsymbol{I}_m)\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right).$$

Since $\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)$ is a projector,

$$\boldsymbol{A}_Y^\top \boldsymbol{A}_Y = (\boldsymbol{Y}_0 \otimes \boldsymbol{I}_m)\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right).$$

For a differentiable function $f : \mathbb{R}^{m\times n} \to \mathbb{R}$, the following relation holds between an arbitrary direction $\boldsymbol{H} \in \mathbb{R}^{m\times n}$ and the Euclidean gradient $\nabla f(\boldsymbol{X})$:

$$\lim_{t\to 0} \quad \frac{d}{dt}\, f(\boldsymbol{X} + t\boldsymbol{H}) = \langle \nabla f(\boldsymbol{X}), \boldsymbol{H}\rangle.$$

The gradient with respect to $\boldsymbol{U}$ can thus be derived as follows:

$$\begin{aligned}
&\lim_{t\to 0} \quad \frac{d}{dt}\, f_U(\boldsymbol{U} + t\boldsymbol{H}) \\
= &\lim_{t\to 0} \quad \frac{d}{dt}\, h_\mu\left(\boldsymbol{x} - \boldsymbol{S}^\dagger\left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right)(\mathrm{vec}(\boldsymbol{U}) + t\ \mathrm{vec}(\boldsymbol{H}))\right) \\
&\quad + \tfrac{1}{mn}\mathrm{vec}(\boldsymbol{\Lambda}_0)^\top \boldsymbol{A}_Y (\mathrm{vec}(\boldsymbol{U}) + t\ \mathrm{vec}(\boldsymbol{H})) \\
&\quad + \tfrac{\rho}{2mn}(\mathrm{vec}(\boldsymbol{U}) + t\ \mathrm{vec}(\boldsymbol{H}))^\top \boldsymbol{A}_Y^\top \boldsymbol{A}_Y (\mathrm{vec}(\boldsymbol{U}) + t\ \mathrm{vec}(\boldsymbol{H})) \\
&= -\left(\nabla h_\mu\left(\boldsymbol{x} - \boldsymbol{S}^\dagger \mathrm{vec}(\boldsymbol{U}\boldsymbol{Y}_0)\right)\right)^\top \boldsymbol{S}^\dagger \left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right)\mathrm{vec}(\boldsymbol{H}) \\
&\quad + \tfrac{1}{mn}\mathrm{vec}(\boldsymbol{\Lambda}_0)^\top \boldsymbol{A}_Y \mathrm{vec}(\boldsymbol{H}) \\
&\quad + \tfrac{\rho}{mn}\mathrm{vec}(\boldsymbol{U})^\top \boldsymbol{A}_Y^\top \boldsymbol{A}_Y \mathrm{vec}(\boldsymbol{H}) \\
&:= \quad \langle \mathrm{vec}(\nabla f_U(\boldsymbol{U})), \mathrm{vec}(\boldsymbol{H})\rangle
\end{aligned}$$

The vectorized gradient is therefore

$$\begin{aligned}
\mathrm{vec}(\nabla f_U(\boldsymbol{U})) = &-(\boldsymbol{Y}_0 \otimes \boldsymbol{I}_m)(\boldsymbol{S}^\dagger)^\top \nabla h_\mu\left(\boldsymbol{x} - \boldsymbol{S}^\dagger \mathrm{vec}(\boldsymbol{U}\boldsymbol{Y}_0)\right) \\
&+ \tfrac{1}{mn}(\boldsymbol{Y}_0 \otimes \boldsymbol{I}_m)\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\mathrm{vec}(\boldsymbol{\Lambda}_0) \\
&+ \tfrac{\rho}{mn}(\boldsymbol{Y}_0 \otimes \boldsymbol{I}_m)\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^\dagger\right)\left(\boldsymbol{Y}_0^\top \otimes \boldsymbol{I}_m\right)\mathrm{vec}(\boldsymbol{U}),
\end{aligned}$$

which finally leads to the gradient

$$\nabla f_U(\boldsymbol{U}) = [-\mathrm{vec}^{-1}\left((\boldsymbol{S}^{\dagger})^{\top}\nabla h_{\mu}\left(\boldsymbol{x} - \boldsymbol{S}^{\dagger}\mathrm{vec}(\boldsymbol{U}\boldsymbol{Y}_0)\right)\right) + \\ \tfrac{1}{mn}\left((\boldsymbol{\Lambda}_0 - \Pi_S(\boldsymbol{\Lambda}_0)) + \rho(\boldsymbol{U}\boldsymbol{Y}_0 - \Pi_S(\boldsymbol{U}\boldsymbol{Y}_0))\right)]\,\boldsymbol{Y}_0^{\top}$$

In a similar manner, the gradient for the cost function

$$f_Y(\boldsymbol{Y}) = h_{\mu}\left(\boldsymbol{x} - \boldsymbol{S}^{\dagger}(\boldsymbol{I}_n \otimes \boldsymbol{U}_0)\,\mathrm{vec}(\boldsymbol{Y})\right) + \tfrac{1}{mn}\mathrm{vec}(\boldsymbol{\Lambda}_0)^{\top}\left(\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^{\dagger}\right)\mathrm{vec}(\boldsymbol{U}_0\boldsymbol{Y})\right) \\ + \tfrac{\rho}{2mn}\left\|\left(\boldsymbol{I}_{mn} - \boldsymbol{S}\boldsymbol{S}^{\dagger}\right)\mathrm{vec}(\boldsymbol{U}_0\boldsymbol{Y})\right\|_2^2$$

with respect to $\boldsymbol{Y}$ can be derived as

$$\nabla f_Y(\boldsymbol{Y}) = \boldsymbol{U}_0^{\top}\,[-\mathrm{vec}^{-1}\left((\boldsymbol{S}^{\dagger})^{\top}\nabla h_{\mu}\left(\boldsymbol{x} - \boldsymbol{S}^{\dagger}\mathrm{vec}(\boldsymbol{U}_0\boldsymbol{Y})\right)\right) + \\ \tfrac{1}{mn}\left((\boldsymbol{\Lambda}_0 - \Pi_S(\boldsymbol{\Lambda}_0)) + \rho(\boldsymbol{U}_0\boldsymbol{Y} - \Pi_S(\boldsymbol{U}_0\boldsymbol{Y}))\right)]$$