

Imperial College London
Department of Computing

Multilevel Optimisation for Computer Vision

Vahan Hovhannisyan

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College and
the Diploma of Imperial College, 2017

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

Declaration

I, Vahan Hovhannisyan, declare that the research presented in this thesis is my own work, except where acknowledged. No part of this thesis has been submitted before for any degree or examination at this or any other university.

Abstract

The recent spark in machine learning and computer vision methods requiring increasingly larger datasets has motivated the introduction of optimisation algorithms specifically tailored to solve very large problems within practical time constraints. This demand in algorithms challenges the practicality of state of the art methods requiring new approaches that can take advantage of not only the problem's mathematical structure, but also its data structure. Fortunately, such structure is present in many computer vision applications, where the problems can be modelled with varying degrees of fidelity. This structure suggests using multiscale models and thus multilevel algorithms.

The objective of this thesis is to develop, implement and test provably convergent multilevel optimisation algorithms for convex composite optimisation problems in general and its applications in computer vision in particular. Our first multilevel algorithm solves convex composite optimisation problem and it is most efficient particularly for the robust facial recognition task. The method uses concepts from proximal gradient, mirror descent and multilevel optimisation algorithms, thus we call it multilevel accelerated gradient mirror descent algorithm (MAGMA). We first show that MAGMA has the same theoretical convergence rate as the state of the art first order methods and has much lower per iteration complexity. Then we demonstrate its practical advantage on many facial recognition problems.

The second part of the thesis introduces new multilevel procedure most appropriate for the robust PCA problems requiring iterative SVD computations. We propose to exploit the multiscale structure present in these problems by constructing lower dimensional matrices and use its singular values for each iteration of the optimisation procedure. We implement this approach on three different optimisation algorithms - inexact ALM, Frank-Wolfe Thresholding and non-convex alternating projections. In this case as well we show that these multilevel algorithms converge (to an exact or approximate) solution with the same convergence rate as their standard counterparts and test all three methods on numerous synthetic and real life problems demonstrating that the multilevel algorithms are not only much faster, but also solve problems that often cannot be solved by their standard counterparts.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors Dr. Panos Parpas and Dr. Stefanos Zafeiriou for their enthusiasm, guidance and unrelenting support throughout the course of my PhD. They were exceptionally helpful with all the in depth discussions and advice. Their positive outlook and confidence in my research inspired and kept me motivated. I am also indebted to Dr. Ryan Duy Luong and Dr. Yannis Panagakis for their help, advice and contributions during the first and second projects, accordingly. I would also like to thank Dr. Marco Laumanns for supervising my Master's thesis at ETH Zurich and instilling my love to operations research and optimisation.

I was exceptionally lucky to be a part of the Computational Optimisation group and would like to express my sincere gratitude to all of its members. I would like to thank Prof. Berc Rustem for his continuous support and advice in academic, career and financial matters during these years; to Dr. Wolfram Wiesemann for the multiple teaching opportunities he presented to me; to Prof. Daniel Kuhn for giving me the chance to join Imperial College London; and to Dr. Ruth Misener for all the great academic and sport activities she organised.

PhD life is full of roller coasters, but I was lucky to have shared it with my colleagues, friends and office mates Dr. Vladimir Roich, Dr. Tran Kha, Sei Howe, Miten Mistry, Georgia Kouyialis, Radu Baltean, Simon Olofsson, Dr. Dimitrios Letsios, Francesco Ceccon and Dr. Jeremy Cohen. I would like to personally thank Dr. Clint Ho and Juan Campos for the in depth mathematical discussions on our whiteboard (also for their positive attitude to life), and to Dr. Grani Hanasusanto for going so easy and letting me win all those football matches. Special thanks goes to my oidas Stephan Kollmann and Dr. Stefan Czimek for making my life in Europe so much better.

My sincere appreciation goes to Dr. Amani El-Kholy for her outstanding effort with all administrative matters.

My special gratitude goes to Luys Foundation and its staff for partly financing my studies since 2010 and for providing the platform to meet so many bright people.

And finally, I thank my family for their unconditional love, support, trust, tolerance and encouragement, especially during the most difficult times of my life. I warmly dedicate this thesis to them.

To my father, mother and sister

"Fake it 'til you become it."

Amy Cuddy

Contents

Copyright	3
Declaration	5
Abstract	7
Acknowledgements	9
1 Introduction	24
1.1 Motivation and Objectives	24
1.2 Contributions and Thesis Outline	25
2 Background	27
2.1 Convex Composite Optimisation	27
2.1.1 Gradient Descent Methods	28
2.1.2 Mirror Descent Methods	30
2.1.3 Accelerated First Order Methods	31
2.1.4 Smoothing and First Order Methods	33
2.1.5 Frank-Wolfe Method	34
2.2 Sparse Decompositions	35

2.2.1	Sparse Solutions for Underdetermined Linear Systems	36
2.2.2	Robust Face Recognition	37
2.2.3	Optimisation Methods for ℓ_1 -min Problems	38
2.3	Robust Principal Component Analysis	44
2.3.1	The CPCP and RPCA models	46
2.3.2	Inexact ALM	47
2.3.3	Frank-Wolfe Thresholding	48
2.3.4	Non-convex Alternating Projections	50
2.4	Multilevel Optimisation Algorithms	52
2.4.1	Introduction to Multilevel Optimisation Algorithms	52
2.4.2	Coarse Model Construction	53
3	MAGMA	55
3.1	Introduction	55
3.2	Multilevel Accelerated Proximal Gradient Algorithm	56
3.2.1	Model Construction	57
3.2.2	MAGMA	59
3.3	Numerical Experiments	67
3.3.1	MAGMA for Solving Robust Face Recognition	68
3.3.2	Numerical Results	69
3.3.3	Stochastic Algorithms for Robust Face Recognition	73
3.4	Conclusion and Discussion	78

4	Multilevel Robust PCA	79
4.1	Introduction	79
4.2	The Coarse Model	81
4.3	Multilevel IALM	83
4.4	Multilevel Frank-Wolfe	91
4.4.1	Multilevel Frank-Wolfe Thresholding for CPCP	93
4.5	Multilevel Non-Convex Alternating Projections	95
4.6	Experiments	96
4.6.1	Synthetic Data	97
4.6.2	Video Background Extraction	98
4.6.3	Shadow removal from facial images	101
4.6.4	Multilevel Non-convex Alternating Projection	104
4.7	Conclusion	109
5	Discussion	110
5.1	Summary	110
5.2	Future Work	111
	Appendices	113
A	List of acronyms	114
	Bibliography	115

List of Tables

3.1	Number of coarse levels used for MISTA and MAGMA for each dictionary	70
3.2	Average CPU running times (seconds) of MISTA, FISTA and MAGMA	71
3.3	Running FISTA, APCG, SVRG and MAGMA on the largest dictionary (with 8824 images) until convergence with $\epsilon = 10^{-6}$ accuracy or maximum time of 10,000 CPU seconds returning $\mathbf{x}^*$ as a solution.	78
4.1	Achieved objective function values and relative errors from ground truth after running IALM and ML-IALM on RPCA problems with synthetic data for a fixed time.	98
4.2	CPU times (in seconds), achieved objective function values and relative errors from ground truth after running FWT and ML-FWT on CPCP problems with synthetic data until 10^{-3} convergence error.	99
4.3	CPU time (in seconds), achieved objective value, rank and sparsity after solving the resulting PCP problem for noisy video background extraction up to tolerance 10^{-3} using the standard Frank-Wolfe Thresholding (FWT) and its multilevel variant ML-FWT.	102
4.4	CPU times (in seconds) after solving shadow removal problems up to a fixed tolerance using the standard Frank-Wolfe Thresholding algorithm and its multilevel variant. For all experiments we used 2 levels for the multilevel algorithm.	105
4.5	CPU times (in seconds), achieved objective function values and relative errors from ground truth after running AltProj and ML-AltProj on non-convex robust PCA problems with synthetic data until 10^{-7} convergence error.	106

List of Figures

3.1	Comparing MAGMA and MISTA with FISTA. Relative CPU times of MAGMA and MISTA compared to FISTA on face recognition experiments with dictionaries of varying number of images (440, 944 and 8824). For each dictionary 20 random input images $\mathbf{b}$ were chosen, which indicate the numbers on horizontal axis. Additionally, the results with 4 random starting points for each experiment are reflected as horizontal error bars.	72
3.2	MAGMA and FISTA convergences over time. Comparisons of gradient mappings and function values of MAGMA, MISTA and FISTA over time.	73
3.3	Comparing FISTA, APCG and SVRG after running them for a fixed amount of time with ((c), (d), (g) and (h)) and without ((a), (b), (e) and (f)) the bucket model on randomly generated data. The first row contains sequences of the objective values and the bottom row contains optimality criteria, i.e gradient mappings.	75
3.4	Result of FISTA, APCG, SVRG and MAGMA in an image decomposition example without occlusion. Top row ((a) to (e)) contains the original input image and reconstructed images from each algorithm. The bottom row ((f) to (i)) contains the solutions $\mathbf{x}^*$ from each corresponding algorithm.	76
3.5	Result of FISTA, APCG, SVRG and MAGMA in an image decomposition example with corruption. Top row ((a)-(e)) contains the original occluded image and reconstructed images from each algorithm. The middle row ((f)-(i)) contains the reconstructed noise ($\mathbf{e}^*$ reshaped as an image) as determined by corresponding algorithms. The bottom row ((j)-(m)) contains the solutions $\mathbf{x}^*$ from each corresponding algorithm.	77

3.6	Comparing the objective values of FISTA, APCG, SVRG and MAGMA after running them for a fixed amount of time on dictionaries of various sizes.	77
3.7	Comparing the first order optimality criteria of FISTA, SVRG, APCG and MAGMA after running them for a fixed amount of time on dictionaries of various sizes.	78
4.1	Examples from solving video background extraction problems via IALM and ML-IALM methods. Both IALM and ML-IALM run for a fixed CPU seconds. Each row corresponds respectively to highway ($48 \times 64 \times 400$), copy machine ($48 \times 72 \times 3, 400$), walk ($240 \times 320 \times 794$) and gates ($240 \times 320 \times 1, 895$) videos from top to bottom. With each frame we also report the respective rank of the low rank component and the feasibility gap (FG): $\ \mathbf{D} - \mathbf{L} - \mathbf{S}\ _F / \ \mathbf{D}\ _F$	100
4.2	Comparing the relative errors during IALM and ML-IALM iterations. The first two columns give relative errors (RE) compared to the ground truth $(\mathbf{L}_0, \mathbf{S}_0)$, and the third column gives feasibility gaps (FG) during iterations. Each row corresponds respectively to highway ($48 \times 64 \times 400$), copy machine ($48 \times 72 \times 3, 400$), walk ($240 \times 320 \times 794$) and gates ($240 \times 320 \times 1, 895$) videos from top to bottom.	102
4.3	Examples from solving facial shadow removal problems via IALM and ML-IALM algorithms on cropped Yale B database ($96 \times 84 \times 2414$). We run both IALM and ML-IALM for fixed five seconds. With each image we also report the respective rank of the low rank component and the feasibility gap (FG): $\ \mathbf{D} - \mathbf{L} - \mathbf{S}\ _F / \ \mathbf{D}\ _F$	103
4.4	Comparing the relative errors during AltProj and ML-AltProj iterations. The first two columns give relative errors compared to the ground truth $(\mathbf{L}_0, \mathbf{S}_0)$, and the third column gives feasibility gaps (FG) during iterations. Each row corresponds to synthetic data of following dimension and rank as given in Table 4.5.	106
4.5	Examples from solving video background extraction problems via non-convex AltProj and ML-AltProj methods. Both AltProj and ML-IALM run until convergence with 10^{-7} error. Each row corresponds respectively to highway ($48 \times 64 \times 400$), copy machine ($48 \times 72 \times 3, 400$), walk ($240 \times 320 \times 794$) and gates ($240 \times 320 \times 1, 895$) videos from top to bottom. With each frame we also report the respective rank of the low rank component and the feasibility gap (FG): $\ \mathbf{D} - \mathbf{L} - \mathbf{S}\ _F / \ \mathbf{D}\ _F$	107

4.6	Examples from solving facial shadow removal problems via the non-convex alternating projections (AltProj) and multilevel alternating projections (ML-AltProj) algorithms on samples from cropped Yale B database ($96 \times 84 \times 65$). We run both methods until convergence with 10^{-7} error. With each experiment we also report the required CPU times in seconds, respective ranks of the low-rank component and the feasibility gaps (FG): $\ \mathbf{D} - \mathbf{L} - \mathbf{S}\ _F / \ \mathbf{D}\ _F$. for each method	108
-----	--	-----

List of Algorithms

1	FISTA	31
2	AGM	32
3	Frank-Wolfe (FW)	34
4	Inexact ALM (IALM)	47
5	Frank-Wolfe Thresholding (FWT)	50
6	Alternating Projections (AltProj)	51
7	MAGMA	62
8	Multilevel Inexact ALM (ML-IALM)	85
9	Multilevel Frank-Wolfe (ML-FW)	91
10	Multilevel Frank-Wolfe Thresholding (ML-FWT)	94
11	Multilevel Alternating Projections (ML-AltProj)	97

Chapter 1

Introduction

1.1 Motivation and Objectives

The recent spark in machine learning and computer vision methods requiring increasingly larger datasets has motivated the introduction of optimisation algorithms specifically tailored to solve very large problems within practical time constraints [SNW12, BCN16]. However, although the state of the art optimisation methods are tuned to operate most efficiently for the problems with special mathematical structure that appear in many machine learning problems, they still lack the ability to exploit the underlying data structure prevailing in most computer vision applications. This lack of most efficient algorithms not only results in often unacceptably long computational times, but also limits the accuracy of the models by enforcing the user to work with impractically small datasets.

Arguably the two most important problems in statistical machine learning are the least absolute shrinkage and selection operator (LASSO) [Tib96] and principle component analysis (PCA) [Hot33] methods. These methods and their extensions have found myriads of applications in computer vision and we will discuss them in more detail in the subsequent chapters. What's more interesting is that they both can be mathematically modelled within the same framework called convex composite optimisation. This is a very generic model and as we will see even the state of the art algorithms are not best suited for solving moderately large problems. Therefore, given the importance of the problems in question and the lack of efficient algorithms that can exploit their underlying data structure, there

is a need for new and specialised methods.

The objective of this thesis is to develop, implement and test provably convergent optimisation algorithms for the above mentioned computer vision problems. We will use ideas and methods from the multilevel optimisation literature [Nas00, WG09, HP16] to significantly accelerate the state of the art methods. The reason is that many computer vision problems can be modelled with varying degrees of fidelity, thus having a multiscale structure. We will exploit this structural knowledge to develop algorithms that perform the most computationally expensive tasks of the optimisation procedures on the model with the lowest possible fidelity and use these as multilevel update steps within the iterative methods.

1.2 Contributions and Thesis Outline

The thesis consists of three main chapters. We begin with Chapter 2, where we give all required background knowledge about the computer vision problems, their mathematical models and state of the art optimisation algorithm. Here we will not dive into the mathematical proofs, but rather present the material in a coherent way citing to appropriate literature for the details.

Chapter 3 introduces our first multilevel algorithm. In general it can solve any convex composite optimisation problem, however it is most efficient and is intended particularly for the robust facial recognition task. Our method uses concepts from proximal gradient, mirror descent and multilevel optimisation algorithms, thus we call it multilevel accelerated gradient mirror descent algorithm (MAGMA). We first show that MAGMA has the same theoretical convergence rate as the state of the art first order methods and has much lower per iteration complexity. Then we demonstrate its practical advantage on many facial recognition problems. The content of this chapter has been published in [HPZ16]. It has also been presented during the 7th British-French-German Conference on Optimization in June 2015 and during the SIAM conference on Imaging Science in May 2016.

Then Chapter 4 introduces a new multilevel procedure most appropriate for robust PCA and related problems. The problems in this category can be separated into two (or more) simple problems and one of them requires performing singular value decomposition on a large, but approximately low rank

matrix. We propose to exploit that structural knowledge by instead constructing lower dimensional matrices and use its singular values. We implement this approach on three different optimisation algorithm. From theoretical point of view, we show that these multilevel algorithms converge (to an exact or approximate) solution with the same convergence rate as their standard counterparts. Moreover, multilevel algorithms have much less per iteration complexity. We then test all three algorithms on numerous synthetic and real life problems and demonstrate that the multilevel algorithms are not only much faster, but also solve problems that often cannot be solved by their standard counterparts. The experimental results of multilevel robust PCA and non-convex alternating projections methods will be presented at ICCV Matrix and Tensor Factorization Methods in Computer Vision workshop and also appear in its proceedings. Furthermore, the multilevel Frank-Wolfe Thresholding method together with all theoretical results has been submitted to the SIAM journal on Imaging Sciences.

Notation. Throughout the thesis, scalars are denoted by lower-case letters, vectors (matrices) are denoted by lower-case (upper-case) boldface letters, e.g. $\mathbf{x}$ ($\mathbf{X}$). $\mathbf{I}$ denotes the identity matrix with appropriate dimension. The ℓ_1 and ℓ_2 norms of a vector $\mathbf{x}$ are defined as $\|\mathbf{x}\|_1 = \sum_i |x_i|$ and $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$, respectively. The matrix ℓ_1 norm is defined as $\|\mathbf{X}\|_1 = \sum_i \sum_j |x_{ij}|$, where $|\cdot|$ denotes the absolute value operator. The Frobenius norm is defined as $\|\mathbf{X}\|_F = \sqrt{\sum_i \sum_j x_{ij}^2}$, and the nuclear norm of $\mathbf{X}$ (i.e., the sum of singular values of a matrix) is denoted by $\|\mathbf{X}\|_*$. We use the $\|\cdot\|_*$ notation also to define the dual norm of $\|\cdot\|$ as $\|\mathbf{z}\|_* = \sup\{\langle \mathbf{z}, \mathbf{x} \rangle : \|\mathbf{x}\| \leq 1\}$. The distinction between matrix nuclear norm from the dual norm will be clear from the context. The l -th largest singular value of matrix $\mathbf{X}$ is denoted as $\sigma_l(\mathbf{X})$. In algorithm pseudocodes we use $\mathbf{X}^{(k)}$ ($\mathbf{x}_k$ and u_k) to denote the value of matrix $\mathbf{X}$ (vector $\mathbf{x}$ and scalar u , accordingly) at iteration k .

Chapter 2

Background

2.1 Convex Composite Optimisation

Composite convex optimisation models consist of the optimisation of a convex function with Lipschitz continuous gradient plus a non-smooth term. They arise in several applications, and are especially prevalent in inverse problems with a sparsity inducing norm and in general convex optimisation with simple constraints (e.g. box or linear constraints). Applications include signal/image reconstruction from few linear measurements (also referred to as compressive or compressed sensing) [Can06, CRT06, CW08, Don06], image super-resolution [YWHM10], data classification [WMM⁺10, WYG⁺09], feature selection [SIL07], sparse matrix decomposition [CRT06], trace norm matrix completion [CR09], image denoising and deblurring [EHN96], to name a few.

Let $f : \mathbb{R}^n \rightarrow (-\infty, \infty]$ be a convex, continuously differentiable function with an L_f -Lipschitz continuous gradient:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_* \leq L_f \|\mathbf{x} - \mathbf{y}\|,$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$ and $g : \mathbb{R}^n \rightarrow (-\infty, \infty]$ be a closed, proper, locally Lipschitz continuous convex function, but not necessarily differentiable. Assuming that the minimizer of the

following optimisation problem:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \{F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})\}, \quad (2.1)$$

is attained, we are concerned with solving it.

Given the large number of applications it is not surprising that several specialized algorithms have been proposed for convex composite models. Second order methods outperform all other methods in terms of the number of iterations required. Interior point methods [Ber99], Newton and truncated Newton methods [KKL⁺07], primal and dual Augmented Lagrangian methods [YZB⁺13] have all been proposed. However, unless the underlying application possesses some specific sparsity pattern second order methods are too expensive to use in applications that arise in practice. As a result the most widely used methods are first order methods. Following the development of accelerated first order methods for the smooth case [Nes83] and the adaptation of accelerated first order methods to the non-smooth case ([BT09b, Nes13]) there has been a large amount of research in this area. State of the art methods use only first order information and as a result (even when accelerated) take a large number of iterations to compute an acceptable solution for large-scale problems. Applications in image processing can have millions of variables and therefore new methods that take advantage of problem structure are needed.

In this section we will introduce the classical and state of the art methods for solving generic convex composite optimisation problems, before showing how they are applied for the particular models studied in this thesis. Then in the subsequent chapters we will introduce new methods aiming to solve problems that the state of the art algorithms cannot.

2.1.1 Gradient Descent Methods

Numerous methods have been proposed to solve (2.1) when g has a simple structure. By simple structure we mean that its proximal mapping, defined as

$$\text{prox}_L(\mathbf{x}) = \arg \min_{\mathbf{y} \in \mathbb{R}^n} \left\{ \frac{L}{2} \|\mathbf{y} - \mathbf{x}\|^2 + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + g(\mathbf{y}) \right\},$$

for some $L \in \mathbb{R}$, is given in a closed form for any f . Correspondingly, denote

$$\text{Prog}_L(\mathbf{x}) = - \min_{\mathbf{y} \in \mathbb{R}^n} \left\{ \frac{L}{2} \|\mathbf{y} - \mathbf{x}\|^2 + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + g(\mathbf{y}) - g(\mathbf{x}) \right\}.$$

We often use the prox and Prog operators with the Lipschitz constant L_f of ∇f . In that case we simplify the notation to $\text{prox}(\mathbf{x}) = \text{prox}_{L_f}(\mathbf{x})$ and $\text{Prog}(\mathbf{x}) = \text{Prog}_{L_f}(\mathbf{x})$ ¹.

When the stepsize is fixed to the Lipschitz constant of the problem, ISTA reduces to the following simple iterative scheme,

$$\mathbf{x}_{k+1} = \text{prox}(\mathbf{x}_k).$$

If $g(\cdot) \equiv 0$, the problem is smooth, $\text{prox}(\mathbf{x}) \equiv \mathbf{x}_k - \frac{1}{L_f} \nabla f(\mathbf{x}_k)$ and ISTA becomes the well-known steepest descent algorithm with

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{L_f} \nabla f(\mathbf{x}_k).$$

For composite optimisation problems it is common to use the gradient mapping as an optimality measure [Nes04]:

$$\mathbf{D}(\mathbf{x}) = \mathbf{x} - \text{prox}(\mathbf{x}).$$

It is a generalisation of the gradient for the composite setting and preserves its most important properties used for the convergence analysis.

We will make use of the following fundamental Lemma in our convergence analysis. The proof can be found, for instance, in [Nes04].

Lemma 1 (Gradient Descent Guarantee). *For any $\mathbf{x} \in \mathbb{R}^n$,*

$$F(\text{prox}(\mathbf{x})) \leq F(\mathbf{x}) - \text{Prog}(\mathbf{x}).$$

Using Lemma 1, it can be shown that ISTA computes an ϵ -optimal solution in terms of function values in $\mathcal{O}(1/\epsilon)$ iterations (see [BT09b] and references therein).

¹These definitions without loss of generality also cover the case when the problem is constrained and smooth. Also see [Nes05] for the original definitions for the unconstrained smooth case.

2.1.2 Mirror Descent Methods

Mirror Descent methods solve the general problem of minimizing a proper convex and locally Lipschitz continuous function F over a simple convex set Q by constructing lower bounds to the objective function (see for instance [NYD82], [Nes09]). Central to the definition of mirror descent is the concept of a distance generating function.

Definition 2.1. A function $\omega(\mathbf{x}) : Q \rightarrow \mathbb{R}$ is called a *Distance Generating Function (DFG)*, if ω is 1-strongly convex with respect to a norm $\|\cdot\|$, that is

$$\omega(\mathbf{z}) \geq \omega(\mathbf{x}) + \langle \nabla \omega(\mathbf{x}), \mathbf{z} - \mathbf{x} \rangle + \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|^2 \quad \forall \mathbf{x}, \mathbf{z} \in Q.$$

Accordingly, the Bregman divergence (or prox-term) is given as

$$V_{\mathbf{x}}(\mathbf{z}) = \omega(\mathbf{z}) - \langle \nabla \omega(\mathbf{x}), \mathbf{z} - \mathbf{x} \rangle - \omega(\mathbf{x}) \quad \forall \mathbf{x}, \mathbf{z} \in Q.$$

The property of DFG ensures that $V_{\mathbf{x}}(\mathbf{x}) = 0$ and $V_{\mathbf{x}}(\mathbf{z}) \geq \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|^2 \geq 0$.

The main step of mirror descent algorithm is given by,

$$\mathbf{x}_{k+1} = \text{Mirr}_{\mathbf{x}_k}(\nabla f(\mathbf{x}_k), \alpha), \quad \text{where} \quad \text{Mirr}_{\mathbf{x}}(\xi, \alpha) = \arg \min_{\mathbf{z} \in \mathbb{R}^n} \{V_{\mathbf{x}}(\mathbf{z}) + \langle \alpha \xi, \mathbf{z} - \mathbf{x} \rangle + \alpha g(\mathbf{z})\}.$$

Here again, we assume that Mirr can be evaluated in closed form. Note that choosing $\omega(\mathbf{z}) = 1/2 \|\mathbf{z}\|_2^2$ as the DFG, which is strongly convex w.r.t. to the ℓ_2 -norm over any convex set and correspondingly $V_{\mathbf{x}}(\mathbf{z}) = \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2$, the mirror step becomes exactly the proximal step of ISTA. However, regardless of this similarity, the gradient and mirror descent algorithms are conceptually different and use different techniques to prove convergence. The core lemma for mirror descent convergence is the so called Mirror Descent Guarantee. For the unconstrained² composite setting it is given below, a proof can be found in [Nes04].

²Simple constraints can be included by defining g as an indicator function.

Lemma 2 (Mirror Descent Guarantee). *Let $\mathbf{x}_{k+1} = \text{Mirr}_{\mathbf{x}_k}(\nabla f(\mathbf{x}_k), \alpha)$, then $\forall \mathbf{u} \in \mathbb{R}^n$*

$$\begin{aligned} \alpha(F(\mathbf{x}_k) - F(\mathbf{u})) &\leq \alpha \langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{u} \rangle + \alpha(g(\mathbf{x}_k) - g(\mathbf{u})) \\ &\leq \alpha^2 L_f \text{Prog}(\mathbf{x}_k) + V_{\mathbf{x}_k}(\mathbf{u}) - V_{\mathbf{x}_{k+1}}(\mathbf{u}). \end{aligned}$$

Using the Mirror Descent Guarantee, it can be shown that the Mirror Descent algorithm converges to the minimizer $\mathbf{x}^*$ in $\mathcal{O}(L_F/\epsilon)$, where ϵ is the convergence precision [BTN13].

2.1.3 Accelerated First Order Methods

A faster method for minimizing smooth convex functions with asymptotically tight $\mathcal{O}(1/\sqrt{\epsilon})$ convergence rate [NYD82] was proposed by Nesterov in [Nes83]. This method and its variants were later extended to solve the more general problem (2.1) (e.g. [BT09b], [Nes13], [AZO14]), see [Tse08] for full review and analysis of accelerated first order methods. The main idea behind the accelerated methods is to update the iterate based on a linear combination of previous iterates rather than only using the current one as in gradient or mirror descent algorithms. The first and most popular method that achieved an optimal convergence rate for composite problems is FISTA [BT09b]. At each iteration it performs one proximal operation, then uses a linear combination of its result and the previous iterator for the next iteration.

Algorithm 1 FISTA

Input: FISTA($f(\cdot)$, $g(\cdot)$, $\mathbf{x}_0$)

```

1: Choose  $\epsilon > 0$ 
2:  $\mathbf{y}_1 \leftarrow \mathbf{x}_0$ 
3:  $t_1 \leftarrow 1$ 
4: for  $k = 0, 1, 2, \dots$  do
5:    $\mathbf{x}_{k+1} \leftarrow \text{prox}(\mathbf{x}_k)$ 
6:    $t_{k+1} \leftarrow \frac{1 + \sqrt{1 + 4t_k^2}}{2}$ 
7:    $\mathbf{y}_{k+1} \leftarrow \mathbf{x}_k + \frac{t_k - 1}{t_{k+1}} (\mathbf{x}_k - \mathbf{x}_{k-1})$ 
8:   if  $\|D(\mathbf{x}_k)\|_* < \epsilon$  then return  $\mathbf{x}_k$ 
9:   end if
10: end for
11:
```

Alternatively, Nesterov's accelerated scheme can be modified to use two proximal operations at each

iteration and linearly combine their results as the current iteration point [Nes05]. In a recent work this approach was reinterpreted as a combination of gradient descent and mirror descent algorithms [AZO14]. The algorithm is given next.

Algorithm 2 AGM

Input: $\text{AGM}(f(\cdot), g(\cdot), \mathbf{x}_0)$

```

1: Choose  $\epsilon > 0$ 
2: Choose DGF  $\omega$  and set  $V_{\mathbf{x}}(\mathbf{y}) = \omega(\mathbf{y}) - \langle \nabla \omega(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle - \omega(\mathbf{x})$ 
3:  $\mathbf{y}_0 \leftarrow \mathbf{z}_0 = \mathbf{x}_0$ 
4: for  $k = 0, 1, 2, \dots$  do
5:    $\alpha_{k+1} \leftarrow \frac{k+2}{2L}$ 
6:    $t_k \leftarrow \frac{2}{k+2}$ 
7:    $\mathbf{x}_k \leftarrow t_k \mathbf{z}_k + (1 - t_k) \mathbf{y}_k$ 
8:   if  $\|D(\mathbf{x}_k)\|_* < \epsilon$  then return  $\mathbf{x}_k$ 
9:   end if
10:   $\mathbf{y}_{k+1} \leftarrow \text{prox}(\mathbf{x}_k)$ 
11:   $\mathbf{z}_{k+1} \leftarrow \text{Mirr}_{\mathbf{z}_k}(\nabla f(\mathbf{x}_k), \alpha_{k+1})$ 
12: end for
13: return  $\mathbf{y}_{k+1}$ 

```

The convergence of Algorithm 2 relies on the following lemmas.

Lemma 3. *If $t_k = \frac{1}{\alpha_{k+1} L_f}$ then it satisfies that for every $\mathbf{u} \in \mathbb{R}^n$,*

$$\begin{aligned}
& \alpha_{k+1} \langle \nabla f(\mathbf{x}_k), \mathbf{z}_k - \mathbf{u} \rangle + \alpha_{k+1} (g(\mathbf{z}_k) - g(\mathbf{u})) \\
& \leq \alpha_{k+1}^2 L_f \text{Prog}(\mathbf{x}_k) + V_{\mathbf{z}_k}(\mathbf{u}) - V_{\mathbf{z}_{k+1}}(\mathbf{u}) \\
& \leq \alpha_{k+1}^2 L_f (F(\mathbf{x}_k) - F(\mathbf{y}_{k+1})) + V_{\mathbf{z}_k}(\mathbf{u}) - V_{\mathbf{z}_{k+1}}(\mathbf{u}).
\end{aligned}$$

Lemma 4 (Coupling). *For any $\mathbf{u} \in \mathbb{R}^n$*

$$\alpha_{k+1}^2 L_f F(\mathbf{y}_{k+1}) - (\alpha_{k+1}^2 L_f - \alpha_{k+1}) F(\mathbf{y}_k) + (V_{\mathbf{z}_{k+1}}(\mathbf{u}) - V_{\mathbf{z}_k}(\mathbf{u})) \leq \alpha_{k+1} F(\mathbf{u}).$$

Theorem 2.1 (Convergence). *Algorithm 2 ensures that*

$$F(\mathbf{y}_T) - F(\mathbf{x}^*) \leq \frac{4\Theta L_f}{T^2},$$

where Θ is any upper bound on $V_{\mathbf{x}_0}(\mathbf{x}^*)$.

Remark. In [AZO14] the analysis of the algorithm is done for smooth constrained problems, however it can be set to work for the unconstrained composite setting as well.

Remark. Whenever L_f is not known or is expensive to calculate, we can use backtracking line search with the following stopping condition

$$F(\mathbf{y}_{k+1}) \leq F(\mathbf{x}_k) - \text{Prog}_L(\mathbf{x}_k), \quad (2.2)$$

where $L > 0$ is the smallest number for which the condition (2.2) holds. The convergence analysis of the algorithm can be extended to cover this case in a straightforward way (see [BT09b]).

2.1.4 Smoothing and First Order Methods

A more general approach of minimizing non-smooth problems is to replace the original problem by a sequence of smooth problems. The smooth problems can be solved more efficiently than by using subgradient type methods on the original problem. In [Nes05] Nesterov proposed a first order smoothing method, where the objective function is of special max-type. Then Beck and Teboulle extended this method to a more general framework for the class of so-called *smoothable* functions [BT12]. Both methods are proven to converge to an ϵ -optimal solution in $\mathcal{O}(1/\epsilon)$ iterations.

Definition 2.2. Let $g : \mathbb{E} \rightarrow (-\infty, \infty]$ be a closed and proper convex function and let $X \subseteq \text{dom } g$ be a closed convex set. The function g is called " (α, β, K) -smoothable" over X , if there exist β_1, β_2 satisfying $\beta_1 + \beta_2 = \beta > 0$ such that for every $\mu > 0$ there exists a continuously differentiable convex function $g_\mu : \mathbb{E} \rightarrow (-\infty, \infty)$ such that the following hold:

1. $g(\mathbf{x}) - \beta_1\mu \leq g_\mu(\mathbf{x}) \leq g(\mathbf{x}) + \beta_2\mu$ for every $\mathbf{x} \in X$.
2. The function g_μ has a Lipschitz gradient over X with a Lipschitz constant such that there exists $K > 0, \alpha > 0$ such that

$$\|\nabla g_\mu(\mathbf{x}) - \nabla g_\mu(\mathbf{y})\|_* \leq \left(K + \frac{\alpha}{\mu}\right) \|\mathbf{x} - \mathbf{y}\| \text{ for every } \mathbf{x}, \mathbf{y} \in X.$$

We often use $L_\mu = K + \frac{\alpha}{\mu}$.

It was shown in [BT12] that with an appropriate choice of the smoothing parameter μ a solution obtained by smoothing the original non-smooth problem and solving it by a method with $\mathcal{O}(1/\sqrt{\epsilon})$ convergence rate finds an ϵ -optimal solution in $\mathcal{O}(1/\epsilon)$ iterations.

2.1.5 Frank-Wolfe Method

The increasingly larger problems have triggered studying and improving optimisation algorithms best fit for the largest problems. One very popular such method is the well studied Frank-Wolfe (FW) method [FW56], also known as the conditional gradient method [LP66]. The method minimises any convex smooth function f over a bounded convex set $\mathcal{D} \subseteq \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space endowed with an inner product $\langle \cdot, \cdot \rangle$, thus in this subsection we study the following optimisation problem:

$$\min f(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{D}, \quad (2.3)$$

where f has a L -Lipschitz continuous gradient:

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{D}, \quad \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|. \quad (2.4)$$

Throughout, we let $D = \max_{\mathbf{x}, \mathbf{y} \in \mathcal{D}} \|\mathbf{x} - \mathbf{y}\|$ denote the diameter of the feasible set $\mathcal{D}$. The Frank-Wolfe method, as well as all its variants and extensions studied in this thesis will assume that the feasibility set $\mathcal{D}$ is bounded, i.e. $D < \infty$. The classical Frank-Wolfe method has many variants with different update rules, but in the most general form it can be written as in Algorithm 3 [MZWG16].

Algorithm 3 Frank-Wolfe (FW)

Input: $\mathbf{x}^{(0)} \in \mathcal{D}$

1: **for** $k \leftarrow 1$ to ... **do**

2: $\mathbf{v}^{(k)} \in \arg \min_{\mathbf{v} \in \mathcal{D}} \langle \mathbf{v}, \nabla f(\mathbf{x}^{(k)}) \rangle$;

3: $\gamma = \frac{2}{k+2}$

4: Update $\mathbf{x}^{(k+1)}$ to a point in $\mathcal{D}$ so that $f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}) + \gamma \langle \mathbf{v}^{(k)} - \mathbf{x}^{(k)}, \nabla f(\mathbf{x}^{(k)}) \rangle$;

5: **end for**

6: **return** $\mathbf{x}^{(k+1)}$

The two most common updating rules for $\mathbf{x}^{(k+1)}$ are the simple

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \gamma(\mathbf{v}^{(k)} - \mathbf{x}^{(k)}), \quad (2.5)$$

and the following slightly more sophisticated one

$$\mathbf{x}^{(k+1)} \in \arg \min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in \text{conv}\{\mathbf{x}^{(k)}, \mathbf{v}^{(v)}\}. \quad (2.6)$$

In this thesis we will use the more advanced update rule (2.6) for its better practical performance. Using standard techniques it can be shown that the FW method converges at a rate of $\mathcal{O}(1/k)$ in function values.

Theorem 2.2. *Let $\mathbf{x}^*$ be an optimal solution of (2.3). For $\{\mathbf{x}^{(k)}\}$ generated by Algorithm 3, we have for $k = 0, 1, 2, \dots$*

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq \frac{2LD^2}{k+2}. \quad (2.7)$$

Proof. The proof can be found, for example, in [MZWG16]. □

2.2 Sparse Decompositions

Finding a sparse (or even the sparsest) decomposition of a given signal/image from an arbitrary dictionary received a lot of attention in recent years. Sparse decompositions have countless applications, including inverse problems, such as signal/image reconstruction from few linear measurements (also referred to as compressive or compressed sensing) [CW08], image super-resolution [YWHM10], data classification [WYG⁺09] and feature selection [SIL07], to name a few.

2.2.1 Sparse Solutions for Underdetermined Linear Systems

An important special case of (2.1) is when $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|_2^2$ and $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a full rank matrix with $m < n$, $\mathbf{b} \in \mathbb{R}^m$ is a vector and $\lambda > 0$ is a parameter. Then (2.1) becomes

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1, \quad (P_1)$$

The problem (P_1) arises from solving underdetermined linear system of equations $\mathbf{Ax} = \mathbf{b}$. This system has more unknowns than equations and thus has infinitely many solutions. A common way of avoiding that obstacle and narrowing the set of solutions to a well-defined one is regularisation via sparsity inducing norms [Ela10]. In general, the problem of finding the sparsest decomposition of a matrix $\mathbf{A}$ with regard to data sample $\mathbf{b}$ can be written as the following non-convex minimisation problem:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{b}, \quad (P_0)$$

where $\|\cdot\|_0$ denotes the ℓ_0 -pseudo-norm, i.e. counts the number of non-zero elements of $\mathbf{x}$. Thus the aim is to recover the sparsest $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{b}$. However, solving the above non-convex problem is NP-hard, even difficult to approximate [AK98]. Moreover, less sparse, but more stable to noise solutions are often more preferred than the sparsest one. These issues can be addressed by minimizing the more forgiving ℓ_1 -norm instead of the ℓ_0 -pseudo-norm:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{b}. \quad (\text{BP})$$

It can be shown that problem (BP) (often referred to as Basis Pursuit (BP)) is convex, its solutions are gathered in a bounded convex set and at least one of them has at most m non-zeros [Ela10]. In order to handle random noise in real world applications, it is often beneficial to allow some constraint violation. Allowing a certain error $\epsilon > 0$ to appear in the reconstruction we obtain the so-called Basis Pursuit Denoising (BPD) problem:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{Ax} - \mathbf{b}\|_2^2 \leq \epsilon, \quad (\text{BPD})$$

or using the Lagrangian dual we can equivalently rewrite it as an unconstrained, but non-smooth problem (P_1). Note that problem (P_1) is equivalent to the well known LASSO regression [Tib96]. Often BP and BPD (in both constrained and unconstrained forms) are referred to as ℓ_1 -min problems.

A relatively new, but very popular example of the BPD problem is the so-called dense error correction (DEC) [WM10], which studies the problem of recovering a sparse signal $\mathbf{x}$ from highly corrupted linear measurements $\mathbf{A}$. It was shown that if $\mathbf{A}$ has highly correlated columns, with an overwhelming probability the solution of (BP) is also a solution for (P_0) [WM10]. One example of DEC is the face recognition (FR) problem, where $\mathbf{A}$ is a dictionary of facial images, with each column being one image, $\mathbf{b}$ is a new incoming image and the non-zero elements of the sparse solution $\mathbf{x}$ identify the person in the dictionary whose image $\mathbf{b}$ is.

2.2.2 Robust Face Recognition

In [WM10] it was shown that a sparse signal $\mathbf{x} \in \mathbb{R}^n$ from highly corrupted linear measurements $\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e} \in \mathbb{R}^n$, where $\mathbf{e}$ is an unknown error and $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a highly correlated dictionary, can be recovered by solving the following ℓ_1 -minimisation problem

$$\min_{\mathbf{x} \in \mathbb{R}^n, \mathbf{e} \in \mathbb{R}^m} \|\mathbf{x}\|_1 + \|\mathbf{e}\|_1 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{b}.$$

It was shown that accurate recovery of sparse signals is possible and computationally feasible even for images with nearly 100% of the observations corrupted.

We introduce the new variable $\mathbf{w} = [\mathbf{x}^T, \mathbf{e}^T]^T \in \mathbb{R}^{n+m}$ and matrix $\mathbf{B} = [\mathbf{A}, \mathbf{I}] \in \mathbb{R}^{m \times (m+n)}$, where $\mathbf{I} \in \mathbb{R}^{m \times m}$ is the identity matrix. The updated (P_1) problem can be written as the following optimisation problem:

$$\min_{\mathbf{w} \in \mathbb{R}^{m+n}} \frac{1}{2} \|\mathbf{B}\mathbf{w} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{w}\|_1. \quad (2.8)$$

Note that this new model generates significantly larger optimisation problems, therefore requires carefully modified algorithms capable of taking into account the problem structure.

A popular application of dense error correction in computer vision, is the face recognition problem. There $\mathbf{A}$ is a dictionary of facial images stacked as column vectors³, $\mathbf{b}$ is an incoming image of a person who we aim to identify in the dictionary and the non-zero entries of the sparse solution $\mathbf{x}^*$ (obtained from the solution of (2.8) $\mathbf{w}^* = [\mathbf{x}^{*T}, \mathbf{e}^{*T}]^T$) indicate the images that belong to the same person as $\mathbf{b}$. As stated above, $\mathbf{b}$ can be highly corrupted, e.g. with noise and/or occlusion.

In a recent survey Yang et. al. [YZB⁺13] compared a number of algorithms solving the face recognition problem with regard to their efficiency and recognition rates. Their experiments showed that the best algorithms were the Dual Proximal Augmented Lagrangian Method (DALM), primal dual interior point method (PDIPA) and L1LS [KKL⁺07], however, the images in their experiments were of relatively small dimension - 40×30 pixels. Consequently the problems were solved within a few seconds. It is important to notice that both DALM and PDIPA are unable to solve large-scale (both large number of images in $\mathbf{A}$ and large image dimensions) problems, as the former requires inverting $\mathbf{A}\mathbf{A}^T$ and the later uses Newton update steps. On the other hand L1LS is designed to take advantage of sparse problems structures, however it performs significantly worse whenever the problem is not very sparse.

2.2.3 Optimisation Methods for ℓ_1 -min Problems

Many optimisation algorithms have been proposed to solve the ℓ_1 -min problems. In this section a brief review of the most visible ones in literature is given. For more details see the appropriate references.

³facial images are, indeed, highly correlated.

Primal-Dual Interior Point Algorithm

For the sake of simplicity we assume that $\mathbf{x} \leq 0$. Under this assumption BP can be rewritten as a LP and solved via the *primal-dual interior-point algorithm* (PDIPA) [Sch98], [BV04]:

$$\begin{array}{ll}
 \textbf{Primal (P)} & \textbf{Dual (D)} \\
 \min_{\mathbf{x}} & \mathbf{1}^T \mathbf{x} \qquad \max_{\mathbf{y}, \mathbf{z}} \quad \mathbf{b}^T \mathbf{y} \\
 \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{subject to} \quad \mathbf{A}^T \mathbf{y} + \mathbf{z} = \mathbf{1} \\
 & \mathbf{x} \geq 0 \qquad \mathbf{z} \geq 0
 \end{array} \tag{2.9}$$

PDIPA uses Newton update steps at each iteration with complexity $O(n^3)$, thus is very sensitive to the problem size and is not suitable for large-scale ℓ_1 -min problems. Therefore fast approximate methods have been proposed in literature. Namely [KKL⁺07] uses pre-conditioned conjugate gradient (PCG) [WN99] to approximately solve the Newton system and develops an interior point method that solves (BPD). PDIPA with this approximation is called *truncated Newton interior-point method* (TNIPM) and it was implemented within L1LS framework [KKL⁺07]. Based on the choice of the pre-conditioner, it can be comparable to the first-order methods in solving large problems with modest accuracy, while is able to solve them with high accuracy at relatively small additional cost.

Homotopy

Homotopy methods were first studied in the context of LASSO [OPT00], then eventually led to Homotopy algorithms for (BP) in [MCW05], [DT08]. The basic idea of Homotopy is as follows: While solving (BPD), it takes advantage of the fact that the objective function $F(\mathbf{x})$ undergoes a homotopy from the ℓ_2 constraint to the ℓ_1 objective as λ decreases, i.e. when $\lambda \rightarrow \infty$, the solution $\mathbf{x}^* = 0$ and when $\lambda \rightarrow 0$, $\mathbf{x}^*$ converges to the solution of (BP). Homotopy also uses Newton update steps, but only on the coefficients in sparse support set $\mathcal{I}$, which could be very small when $\mathbf{x}$ is sparse. The computational cost is then bounded by $O(dm^2 + dmn)$, if it correctly recovers a d -sparse signal in d steps [DT08]. However when the sparsity of $\mathbf{x}$ and the observation dimension m grow proportionally with the signal dimension n , the worst-case complexity is still $O(n^3)$.

Proximal Gradient Methods

There are several first order algorithms aiming to solve (BPD) in particular and (2.1) in general, amongst which the ones most related to our research are the ISTA and FISTA (see [BT09b] and references therein).

The main step of ISTA is of form

$$\mathbf{x}_{k+1} = \text{prox}(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)), \quad (2.10)$$

where

$$\text{prox}(\mathbf{y}) = \arg \min_{\mathbf{x} \in \mathbb{R}} \left\{ g(\mathbf{x}) + \frac{1}{2\alpha_k} \|\mathbf{x} - \mathbf{y}\|^2 \right\}, \quad (2.11)$$

where α_k is the step-size at iteration k and can be chosen by a backtracking procedure.

In case of (BPD), the proximal operator $\text{prox}(\mathbf{y} - \alpha \nabla f(\mathbf{y}))$ becomes the so-called shrinkage-thresholding operator $\mathcal{T}_\alpha(\mathbf{x})_i = (|\mathbf{x}_i| - \alpha)_+ \text{sgn}(\mathbf{x}_i)$ and the iteration step is $\mathbf{x}_{k+1} = \mathcal{T}_{t\lambda}(\mathbf{x}_k t \mathbf{A}^T (\mathbf{A} \mathbf{x}_k - \mathbf{b}))$.

In a recent study [BT09b] presents a new *fast iterative shrinkage-thresholding algorithm* (FISTA) for solving (2.1), which is both computationally simple and have proven better global rate of convergence on function values than the original ISTA. FISTA enjoys improved rate of convergence of $O(1/k^2)$ instead of $O(1/k)$ of ISTA. This improvement is achieved by adding one well chosen extra point in each iteration (similar to the "optimal" method of Nemirovski and Yudin [NYD82]). FISTA is discussed in more detail in the next section.

The complexity of FISTA at each iteration is bound by the complexity of the matrix-vector multiplication $(\mathbf{A}^T \mathbf{A})\mathbf{x}$, assuming that $\mathbf{A}^T \mathbf{A}$ can be calculated and stored before the iterations start. Thus it has $O(n^2)$ worst case complexity.

Augmented Lagrangian Methods

In [BCG11] the proximal operator is used to solve convex conic programs of form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathcal{A}(\mathbf{x}) + \mathbf{b} \in \mathcal{K}, \end{aligned} \tag{2.12}$$

where $f(\cdot)$ is a convex function, $\mathcal{A}$ is a linear operator, $\mathbf{b}$ is a fixed point and $\mathcal{K}$ is a closed convex cone. Note that both BP and P_1 fall into this category. The idea behind TFOCS is to first formulate the dual problem of (2.12), then smooth its objective function if it's not smooth, then use the proximal operator to solve the arising dual problem while obtaining the primal solutions during each iteration.

It is proven in [BCG11] that for sufficiently small smoothing parameter TFOCS converges to the optimal solution of (2.12), however its convergence speed too depends on that parameter.

In [YZB⁺13] two other Augmented Lagrangian Methods (ALM) methods were proposed: primal and dual. Both methods are used to solve the constrained (BP) problem.

The Augmented Lagrangian of (BP) is given by

$$\mathcal{L}_\xi(\mathbf{x}, \theta) = g(\mathbf{x}) + \frac{\xi}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \theta^T(\mathbf{b} - \mathbf{A}\mathbf{x}), \tag{2.13}$$

where $\theta \in \mathbb{R}^m$ is a vector of Lagrange multipliers and $\xi > 0$ is a smoothing penalty parameter. It is proven that the optimal solution of (BP) can be found by minimizing the augmented Lagrangian function $\mathcal{L}_\xi(\mathbf{x}, \theta)$ [Ber99], [YZB⁺13]. It can be done by using the method of multipliers [Ber99] as follows:

$$\begin{cases} \mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} \mathcal{L}_{\xi_k}(\mathbf{x}, \theta_k) \\ \theta_{k+1} = \theta_k + \xi_k(\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}) \end{cases}, \tag{2.14}$$

where ξ_k is a predefined positive sequence. In [YZB⁺13] FISTA was used to solve the $\mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} \mathcal{L}_{\xi_k}(\mathbf{x}, \theta_k)$ problem, thus has worst case complexity $O(n^2)$. This method is called Primal ALM and was shown to perform better when the dimension n of $\mathbf{A}$ is small, but is noticeably slow for large n .

In [YZB⁺13] ALM was also applied to the dual problem of BP:

$$\begin{aligned} \max_{\mathbf{y} \in \mathbb{R}^m} \quad & \mathbf{b}^T \mathbf{y} \\ \text{subject to} \quad & \mathbf{A}^T \mathbf{y} \in \mathbf{B}_1^\infty, \end{aligned} \quad (2.15)$$

where $\mathbf{B}_1^\infty = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_\infty \leq 1\}$. Then the associated augmented Lagrangian function is given by

$$\begin{aligned} \max_{\mathbf{y}, \mathbf{z}} \quad & -\mathbf{b}^T \mathbf{y} - \mathbf{x}^T (\mathbf{z} - \mathbf{A}^T \mathbf{y}) + \frac{\beta}{2} \|\mathbf{z} - \mathbf{A}^T \mathbf{y}\|_2^2, \\ \text{subject to} \quad & \mathbf{z} \in \mathbf{B}_1^\infty \end{aligned} \quad (2.16)$$

here $\mathbf{x}$ is the Lagrangian multiplier for the dual problem. Here as well the authors propose to alternatively minimize the objective function w.r.t. to one of the variables while holding the rest constant in order to avoid the difficulties of minimizing w.r.t. $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{z}$ simultaneously.

Given that $\mathbf{A}\mathbf{A}^T$ is invertible the main step of the proposed Dual ALM becomes

$$\begin{aligned} \mathbf{z}_{k+1} &= \mathcal{P}_{\mathbf{B}_1^\infty}(\mathbf{A}^T \mathbf{y}_k + \mathbf{x}_k / \beta); \\ \mathbf{y}_{k+1} &= (\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{z}_{k+1} - (\mathbf{A}\mathbf{x}_k - \mathbf{b}) / \beta); \\ \mathbf{x}_{k+1} &= \mathbf{x}_k - \beta(\mathbf{z}_{k+1} - \mathbf{A}^T \mathbf{y}_{k+1}). \end{aligned} \quad (2.17)$$

Where $\mathcal{P}_{\mathbf{B}_1^\infty}$ is the projection operator onto $\mathbf{B}_1^\infty$. Note that the inverse $(\mathbf{A}\mathbf{A}^T)^{-1}$ can be calculated before starting the iterations and in case of moderate number of images in the dictionary DALM is a very fast, with complexity $O(m^2 + mn)$. However, it lacks the ability to handle the special structure of $\mathbf{B} = [\mathbf{A}\mathbf{I}]$ when the images are occluded. This means that $\mathbf{A}$ must be replaced by $\mathbf{B}$ and $\mathbf{x}$ by $\mathbf{w} = [\mathbf{x}_e]$, and when the image dimensions are relatively large inverting $\mathbf{B}^T \mathbf{B}$ can be extremely expensive.

Parallel Coordinate Descent

In [EMZ07] the proximal operator is used in a coordinate descent (CD) manner, i.e. at each iteration k the current estimate $\mathbf{x}_k$ is updated one entry at a time. This way the sequence $\mathbf{x}_k$ is proven to converge, however it requires access to each column of dictionary $\mathbf{A}$, which is often computationally inefficient, because $\mathbf{A}\mathbf{2}$ can contain transforms instead of the actual images. This problem was solved

in [EMZ07] by changing to a so-called parallel update rule resulting to a PCD algorithm.

In [EMZ07] PCD is further accelerated by using a *sequential subspace optimisation* SESOP technique. The idea behind SESOP is that instead of searching along the current direction as in PCD algorithm, directions from the last M steps is also considered, where M is user defined.

Stochastic Gradient Descent

In recent years a very popular methodology for solving ℓ_1 -regularised separable optimisation problems is the stochastic gradient descent method and its extensions. In this subsection we will give a brief review of these methods and discuss in what setting they are appropriate for our problems in hand. For an excellent review of recent developments in stochastic optimisation algorithms applied to machine learning problems we refer the reader to [BCN16]. More specifically, in this subsection we briefly introduce the empirical risk minimisation (ERM) problem, see for instance [HTF02], and some algorithms for solving it. Here the minimisation problem is a special case of the convex composite optimisation model defined in (2.1):

$$\min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x}) = f(\mathbf{x}) + \Psi(\mathbf{x}), \quad (2.18)$$

where f and Ψ are proper convex lower semicontinuous convex functions and f is smooth. Moreover, for this particular setting we also assume that f can be written as an average of many smooth functions as follows:

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x}). \quad (2.19)$$

Clearly, regularised least squares problems can be modelled as ERM, with each f_i corresponding to $\mathbf{A}_{i,:}\mathbf{x} - \mathbf{b}_i$, where $\mathbf{A}_{i,:}$ is the i -th row of matrix $\mathbf{A}$. Many statistical machine learning problems have been modelled as ERM, thus provoking a development of many very efficient algorithms for solving large-scale ERM problems. Particularly, most popular methods are derivations of the stochastic gradient (SG) method. Namely, we implemented one recently very popular algorithm called proximal stochastic variance reduced gradient (SVRG) [XZ14] and tested it on robust facial recognition

problems.

The main idea behind SG methods is to use one (or some) of the coordinates of the components f_i so that instead of gradient updates at each iteration we choose one subset of f_i and use it as an estimate for the true gradient. This results in a convergent method with extremely low per iteration complexity, however the continuous gradient approximation results in increasing variance, so the variance reduction technique [XZ14] has been introduced so that at each iteration the gradient is corrected so that the variance remains zero.

Another very popular randomised method is the accelerated randomised proximal coordinate gradient method (APCG) [LLX15, FR15]. This is a accelerated randomised version of the coordinate descent method, so that at each iteration only one (or some) of the coordinates of $\mathbf{x}$ are used. In this case the gradient will be estimated exactly and thus there is no need for variance reduction. However, this method should be applied on the dual of ERM.

While the robust facial recognition problem can be modelled as ERM and thus both SVRG and APCG can be applied to solve it, as we will see in Chapter 3 their are not appropriate for this particular problem.

2.3 Robust Principal Component Analysis

Low-rank matrix recovery is a cornerstone in data analysis and dimensionality reduction, with the principal component analysis (PCA) [Hot33] being the most widely employed method for this task. Even though the PCA is easy to do by means of eigen-decomposition, it is fragile to the presence of gross, non-Gaussian noise and outliers and the estimated low-rank subspace may be arbitrarily away from the true one; even when a small fraction of the data is corrupted [Hub11]. To alleviate this, robust PCA (RPCA) models have been proposed [CLMW11]. The RPCA aims to recover a low-rank matrix from sparse corruptions that are of unknown value and support by decomposing the observation matrix $\mathbf{D}$ into two terms: a low-rank matrix $\mathbf{L}$ and a sparse one $\mathbf{S}$, accounting for sparse noise and outliers, namely $\mathbf{D} = \mathbf{L} + \mathbf{S}$. In [WGMM13] the RPCA model has been extended to the

case where the low-rank and sparse components can be recovered from a small set of linear measurements. Mathematically, observations have the form $\mathbf{D} \doteq \mathcal{P}_{\mathcal{Q}}[\mathbf{M}] = \mathcal{P}_{\mathcal{Q}}[\mathbf{L} + \mathbf{S}]$, where $\mathcal{Q} \subseteq \mathbb{R}^{m \times n}$ is a linear subspace and $\mathcal{P}_{\mathcal{Q}}$ denotes the projection operator onto that subspace. This model was called *compressive principle component pursuit* (CPCP). These models have profound impact in visual data analysis and computer vision applications such as image denoising [CLMW11], background subtraction, image alignment [PGW⁺12], texture recovery [ZGLM12], deformable models [SPZP14], face finalisation [SPZP16], structure from motion [AZP11], to mention but a few examples.

A natural approach to estimate the low-rank plus sparse decomposition of the RPCA is to minimise the rank of $\mathbf{L}$ and the number of non-zero entries of $\mathbf{S}$, measured by ℓ_0 quasi norm [CLMW11]. Unfortunately, both rank and ℓ_0 -norm minimisation are NP-hard [VB96, Nat95]. The nuclear- and the ℓ_1 - norms are typically adopted as convex surrogates to rank and ℓ_0 - norm, respectively yielding a convex relaxation, which can be provably solved under some natural conditions on the low rank and sparse components.

Common solvers for the convex RPCA and CPCP models include: Iterative Thresholding (IT) [DDDM04], Accelerated Proximal Gradient (APG) [PB13], Augmented Lagrange Multipliers (ALM) [LCM10] and Linearized Augmented Lagrangian method [YY13]. However, all these solvers exhibit significant computational drawbacks. In particular, their each iteration requires computing several (not necessarily all) singular values and vectors of a large matrix, which can be computationally expensive even when only a few singular values are required.

There have been several attempts to reduce the computational time of large nuclear norm regularised optimisation problems. Namely, [LY12] proposed to reduce the problem dimension by writing the large solution matrix as a product of a small orthogonal and another matrix. They solved the resulting non-convex problem via an augmented Lagrangian alternating direction method. Another very popular approach for reducing large problem dimensions is to create smaller sub-problems using various randomised techniques [AZL16, DKM06, LLSG14, MM15, OMTSK15].

The increasingly large problem dimensions have encourage new approaches for solving problems that require SVDs. Most notably, in recent years Frank-Wolfe (FW) algorithms have regained popularity for their extremely low per iteration complexity. Particularly, if algorithms based on proximal

gradient updates require computing a few largest singular values of a large matrix at each iteration, FW-based methods require only the largest singular value. This makes the FW methods very attractive. Specifically, the Frank-Wolfe Thresholding (FWT) method proposed in [MZWG16] is arguably the most efficient method for solving large CPCP problems. However, these methods need significantly more iterations to converge and even computing only the largest singular value of a large matrix for hundreds of iterations can be impractically time consuming.

2.3.1 The CPCP and RPCA models

In this subsection we give a formal presentation of the *compressive principal component pursuit* (CPCP) [WGMM13] model. Let $\mathcal{Q} \subseteq \mathbb{R}^{m \times n}$ be a linear subspace spanned by the set of sensing matrices, and $\mathcal{P}_{\mathcal{Q}}$ denotes the projection operator onto that subspace. In many applications $\mathcal{Q}$ is the subset of observed values of data $\mathbf{D}$. Then the problem is to find a low rank matrix $\mathbf{L}^*$ and a sparse matrix $\mathbf{S}^*$ such that the $\mathcal{P}_{\mathcal{Q}}[\mathbf{L}^* + \mathbf{S}^*] = \mathcal{P}_{\mathcal{Q}}[\mathbf{D}]$. The problem can be written as a convex unconstrained minimisation problem as follows:

$$\min_{\mathbf{L}, \mathbf{S}} \frac{1}{2} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{L} + \mathbf{S} - \mathbf{D}]\|_F^2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{S}\|_1, \quad (\text{CPCP})$$

where λ_L and λ_S are positive penalising coefficients.

An important special case of (CPCP) is the well known robust principle component analysis (RPCA) problem, where $\mathcal{Q}$ is the entire space $\mathbb{R}^{m \times n}$ and $\mathcal{P}_{\mathcal{Q}}$ is the identity, i.e. all values of $\mathbf{D}$ have been observed. The problem in this case can be formulated as to represent the input data matrix $\mathbf{D} \in \mathbb{R}^{m \times n}$ as a sum of a low rank matrix $\mathbf{L}^*$ and a sparse matrix $\mathbf{S}^*$. This can be exactly solved via the following convex constrained optimisation problem:

$$\min_{\mathbf{L}, \mathbf{S}} \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1, \quad \text{subject to } \mathbf{D} = \mathbf{L} + \mathbf{S}, \quad (\text{RPCA})$$

where $\lambda > 0$ is a weighting parameter.

In the subsequent subsections we will give state of the art methods for solving the RPCA and CPCP

problems.

2.3.2 Inexact ALM

We start with the simpler RPCA problem. A classical approach for solving (RPCA) is by minimising its augmented Lagrangian defined as

$$\mathcal{L}(\mathbf{L}, \mathbf{S}, \mathbf{Y}, \mu) = \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 + \langle \mathbf{Y}, \mathbf{D} - \mathbf{L} - \mathbf{S} \rangle + \frac{\mu}{2} \|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_F^2, \quad (2.20)$$

where $\mathbf{Y} \in \mathbb{R}^{m \times n}$ is the Lagrangian variable and $\mu > 0$ is a penalty parameter. (2.20) can be solved via alternating directions method, aka solve the problem for each variable separately at each iteration, while μ_k is increasing at each iteration [LCM10]. In this case each resulting subproblem has a closed form solution.

Solving $\mathbf{S}^{(k+1)} = \arg \min_{\mathbf{S}} \mathcal{L}(\mathbf{L}^{(k+1)}, \mathbf{S}, \mathbf{Y}^{(k)}, \mu_k)$ is equivalent to stacking $\mathbf{S}$ as a vector and computing its proximal operator, which can be done using the soft thresholding operator (see e.g. [BT09b]). The solution of $\mathbf{L}^{(k+1)} = \arg \min_{\mathbf{L}} \mathcal{L}(\mathbf{L}, \mathbf{S}^{(k)}, \mathbf{Y}^{(k)}, \mu_k)$ is given by thresholding the singular values of $\mathbf{D} - \mathbf{S}^{(k)} + \mu_k^{-1} \mathbf{Y}^{(k)}$ [CCS10]. The resulting procedure was dubbed Inexact ALM (IALM) in [LCM10] and is formally given here in Algorithm 4.

Algorithm 4 Inexact ALM (IALM)

Input: $\mathbf{D}, \mathbf{S}^{(0)}, \mathbf{Y}^{(0)} \in \mathbb{R}^{m \times n}; \mu_0 > 0$

- 1: **for** $k \leftarrow 1$ to ... **do**
- 2: // Solve $\mathbf{L}^{(k+1)} = \arg \min_{\mathbf{L}} \mathcal{L}(\mathbf{L}, \mathbf{S}^{(k)}, \mathbf{Y}^{(k)}, \mu_k)$
- 3: $\mathbf{M}^{(k)} \leftarrow \mathbf{D} - \mathbf{S}^{(k)} + \mu_k^{-1} \mathbf{Y}^{(k)}$
- 4: $(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) \leftarrow \text{SVD}(\mathbf{M}^{(k)})$
- 5: $\mathbf{L}^{(k+1)} \leftarrow \mathbf{U} \mathcal{S}_{\mu_k^{-1}}[\mathbf{\Sigma}] \mathbf{V}^\top$
- 6: // Solve $\mathbf{S}^{(k+1)} = \arg \min_{\mathbf{S}} \mathcal{L}(\mathbf{L}^{(k+1)}, \mathbf{S}, \mathbf{Y}^{(k)}, \mu_k)$
- 7: $\mathbf{S}^{(k+1)} \leftarrow \mathcal{S}_{\lambda \mu_k^{-1}}[\mathbf{D} - \mathbf{L}^{(k+1)} + \mu_k^{-1} \mathbf{Y}^{(k)}]$
- 8: // Update the Lagrangian variable
- 9: $\mathbf{Y}^{(k+1)} \leftarrow \mathbf{Y}^{(k)} + \mu_k (\mathbf{D} - \mathbf{L}^{(k+1)} - \mathbf{S}^{(k+1)})$
- 10: Update $\mu_k \leftarrow \mu_{k+1}$
- 11: **end for**
- 12: **return** $(\mathbf{L}^{(k+1)}, \mathbf{S}^{(k+1)})$

Minimising (2.20) over $\mathbf{L}$ requires computing singular values of a large $m \times n$ matrix. It is well known

that computing the k largest singular values has a computational complexity of $\mathcal{O}(kmn)$. Thus for practical efficiency it is important to compute only a few singular values [LCM10]. However, the SVD of step 4 remains the computational bottleneck of Algorithm 4. Theorem 2.3 [LCM10] gives an asymptotic convergence result for Algorithm 4.

Theorem 2.3. *For Algorithm 4, if μ_k is non-decreasing and $\sum_{k=1}^{+\infty} \mu_k^{-1} = +\infty$, then $(\mathbf{L}^{(k)}, \mathbf{S}^{(k)})$ asymptotically converges to an optimal solution of the RPCA problem.*

2.3.3 Frank-Wolfe Thresholding

In this section we will present how the Frank-Wolfe method can be applied to problem (CPCP). Since the FW algorithm can be applied only for smooth and constrained convex optimisation problems with a bounded feasible set, we first need to reformulate (CPCP) into a such problem. In [MZWG16] it was done by first performing an epigraph reformulation on (CPCP) obtaining

$$\begin{aligned} \min \quad & f(\mathbf{L}, \mathbf{S}, t_L, t_S) := \frac{1}{2} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{L} + \mathbf{S} - \mathbf{D}]\|_F^2 + \lambda_L t_L + \lambda_S t_S \\ \text{s.t.} \quad & \|\mathbf{L}\|_* \leq t, \|\mathbf{S}\|_1 = t_S. \end{aligned} \quad (2.21)$$

Now the objective function has a 2-Lipschitz gradient ∇f with partial derivatives given as follows:

$$\nabla_L f(\mathbf{L}, \mathbf{S}, t_L, t_S) = \nabla_S f(\mathbf{L}, \mathbf{S}, t_L, t_S) = \mathcal{P}_{\mathcal{Q}}[\mathbf{L} + \mathbf{S} - \mathbf{D}], \quad (2.22)$$

$$\nabla_{t_L} f(\mathbf{L}, \mathbf{S}, t_L, t_S) = \lambda_L, \nabla_{t_S} f(\mathbf{L}, \mathbf{S}, t_L, t_S) \leq \lambda_S. \quad (2.23)$$

Then to make the feasible region bounded we introduce upper bounds U_L and U_S for t_L and t_S , respectively. It can be shown [MZWG16] that we can choose

$$U_L = \frac{1}{2\lambda_L} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{D}]\|_F^2, \quad U_S = \frac{1}{2\lambda_S} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{D}]\|_F^2, \quad (2.24)$$

and the resulting feasible set has a bounded diameter: $D \leq \sqrt{5} \sqrt{U_L^2 + U_S^2}$.

Now we can apply the FW algorithm on

$$\begin{aligned} \min \quad & f(\mathbf{L}, \mathbf{S}, t_L, t_S) := \frac{1}{2} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{L} + \mathbf{S} - \mathbf{D}]\|_F^2 + \lambda_L t_L + \lambda_S t_S \\ \text{s.t.} \quad & \|\mathbf{L}\|_* \leq t \leq U_L, \|\mathbf{S}\|_1 = t_S \leq U_S. \end{aligned} \quad (2.25)$$

Setting $\mathbf{x} = (\mathbf{L}, \mathbf{S}, \lambda_L, \lambda_S)$ and using the gradient expressions (2.22)-(2.23) we can derive the linear optimisation oracle in step 2 of Algorithm 3 as two independent linear optimisation problems:

$$(\mathbf{V}_L^{(k)}, V_{t_L}^{(k)}) \in \arg \min_{\|\mathbf{V}_L\|_* \leq V_{t_L} \leq U_L} \langle \mathcal{P}_{\mathcal{Q}}[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}], \mathbf{V}_L \rangle + \lambda_L V_{t_L}, \quad (2.26)$$

$$(\mathbf{V}_S^{(k)}, V_{t_S}^{(k)}) \in \arg \min_{\|\mathbf{V}_S\|_* \leq V_{t_S} \leq U_S} \langle \mathcal{P}_{\mathcal{Q}}[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}], \mathbf{V}_S \rangle + \lambda_S V_{t_S}. \quad (2.27)$$

Both (2.26) and (2.27) are seperable and can be solved in closed form using the leading singular values and the largest in magnitude elements of $\mathcal{P}_{\mathcal{Q}}[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}]$ [MZWG16]. This is given in steps 3 – 16 in Algorithm 5. Finally, step 17 uses the update rule (2.6) for problem (2.25). This can be formulated as an quadratic problem over the set $[0, 1]^2$ with two variables by writing the convex hulls of the two points as their convex combinations and solved using any standard method.

A major drawback of the FW method is that for the ℓ_1 -norm, at each iteration it projects a linear function on a ℓ_1 ball, thus updating only one entry of a very large matrix. To resolve this problem [MZWG16] suggested to add a thresholding operation to the FW algorithm, calling it Frank-Wolfe Thresholding (FWT). This is performed in steps 18 – 19 of Algorithm 5. Finally, as suggested in [MZWG16], in steps 20 – 21 we update the bounds U_L and U_S tightening the feasibility set.

We present the complete Frank-Wolfe Thresholding method in Algorithm 5. Both primal and dual convergence of the FWT algorithm was established in [MZWG16]. Since the thresholding (and more generally proximal) operator decreases the function value more than the Frank-Wolfe update, The Frank-Wolfe Thresholding algorithm can be seen as a special case of Algorithm 3 [MZWG16]. Therefore FWT converges to the solution $(\mathbf{L}^*, \mathbf{S}^*)$ in function value with the same $\mathcal{O}(1/k)$ rate. Although the FWT method requires only computing the largest singular value of a $m \times n$ matrix at each iteration,

SVD computations still remain the computational bottleneck.

Algorithm 5 Frank-Wolfe Thresholding (FWT)

Input: $\mathbf{D} \in \mathbb{R}^{m \times n}$; $\lambda_L, \lambda_S > 0$

- 1: Set $\mathbf{L}^{(0)} = \mathbf{S}^{(0)} = \mathbf{0}$; $t_L^{(0)} = t_S^{(0)} = 0$; $U_L^{(0)} = f(\mathbf{L}^{(0)}, \mathbf{S}^{(0)}, t_L^{(0)}, t_S^{(0)})/\lambda_L$; $U_S^{(0)} = f(\mathbf{L}^{(0)}, \mathbf{S}^{(0)}, t_L^{(0)}, t_S^{(0)})/\lambda_S$.
- 2: **for** $k \leftarrow 1$ to ... **do**
- 3: $\mathbf{M}_L^{(k)} \in \arg \min_{\|\mathbf{M}_L^{(k)}\|_* \leq 1} \langle \mathcal{P}_Q[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}], \mathbf{M}_L^{(k)} \rangle$;
- 4:
- 5: $\mathbf{M}_S^{(k)} \in \arg \min_{\|\mathbf{M}_S^{(k)}\|_1 \leq 1} \langle \mathcal{P}_Q[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}], \mathbf{M}_S^{(k)} \rangle$;
- 6:
- 7: **if** $\lambda_L \geq -\langle \mathcal{P}_Q[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}], \mathbf{M}_L^{(k)} \rangle$ **then**
- 8: $\mathbf{V}_L^{(k)} = \mathbf{0}$; $V_{t_L}^{(k)} = 0$
- 9: **else**
- 10: $\mathbf{V}_L^{(k)} = U_L \mathbf{M}_L^{(k)}$; $V_{t_L}^{(k)} = U_L$
- 11: **end if**
- 12: **if** $\lambda_S \geq -\langle \mathcal{P}_Q[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}], \mathbf{M}_S^{(k)} \rangle$ **then**
- 13: $\mathbf{V}_S^{(k)} = \mathbf{0}$; $V_{t_S}^{(k)} = 0$
- 14: **else**
- 15: $\mathbf{V}_S^{(k)} = U_S \mathbf{M}_S^{(k)}$; $V_{t_S}^{(k)} = U_S$
- 16: **end if**
- 17: Compute $(\mathbf{L}^{(k+1)}, \mathbf{S}^{(k+\frac{1}{2})}, t_L^{(k+1)}, t_S^{(k+\frac{1}{2})})$ as a minimiser of

$$\begin{aligned}
& \min_{\mathbf{L}, \mathbf{S}, t_L, t_S} \quad \frac{1}{2} \|\mathcal{P}_Q[\mathbf{L} + \mathbf{S} - \mathbf{D}]\|_F^2 + \lambda_L t_L + \lambda_S t_S \\
& s.t. \quad \begin{pmatrix} \mathbf{L} \\ t_L \end{pmatrix} \in \text{conv} \left\{ \begin{pmatrix} \mathbf{L}^{(k)} \\ t_L^{(k)} \end{pmatrix}, \begin{pmatrix} \mathbf{V}_L^{(k)} \\ V_{t_L}^{(k)} \end{pmatrix} \right\}, \\
& \quad \begin{pmatrix} \mathbf{S} \\ t_S \end{pmatrix} \in \text{conv} \left\{ \begin{pmatrix} \mathbf{S}^{(k)} \\ t_S^{(k)} \end{pmatrix}, \begin{pmatrix} \mathbf{V}_S^{(k)} \\ V_{t_S}^{(k)} \end{pmatrix} \right\};
\end{aligned} \tag{2.28}$$

- 18: $\mathbf{S}^{(k+1)} \leftarrow \mathcal{S}_{\lambda_S}[\mathbf{S}^{(k+\frac{1}{2})} - \mathcal{P}_Q[\mathbf{L}^{(k+\frac{1}{2})} + \mathbf{S}^{(k+\frac{1}{2})} - \mathbf{D}]]$
 - 19: $t_S^{(k+1)} \leftarrow \|\mathbf{S}^{(k+1)}\|_1$
 - 20: $U_L^{(k+1)} \leftarrow g(\mathbf{L}^{(K+1)}, \mathbf{S}^{(k+1)}, t_L^{(k+1)}, t_S^{(k+1)})/\lambda_L$
 - 21: $U_S^{(k+1)} \leftarrow g(\mathbf{L}^{(K+1)}, \mathbf{S}^{(k+1)}, t_L^{(k+1)}, t_S^{(k+1)})/\lambda_S$
 - 22: **end for**
 - 23: **return** $(\mathbf{L}^{(k+1)}, \mathbf{S}^{(k+1)})$
-

2.3.4 Non-convex Alternating Projections

In a recent paper Netrapalli et al proposed a new method for recovering a low-rank matrix from sparse corruptions [NNS⁺14]. Its main idea is to perform alternating projects onto low rank and sparse

matrix spaces. Although these sets are not convex, projections onto them can be done efficiently using the hard thresholding operator $\mathcal{H}_\zeta[x]$, which is applied on vectors and matrices element-wise, i.e. $\mathcal{H}_\zeta[\mathbf{X}]_{i,j} = \mathbf{X}_{i,j}$ if $|\mathbf{X}_{i,j}| \geq \zeta$ and 0 otherwise. Specifically, solve the following non-convex problem of finding a low rank plus sparse matrix decomposition:

$$\min_{\mathbf{L}, \mathbf{S}} \|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_2, \quad \text{subject to } \mathbf{L} \text{ is low rank, and } \mathbf{S} \text{ is sparse} \quad (2.29)$$

where r is a given upper bound on the rank of low rank component $\mathbf{L}^*$. Although the problem in (2.29) is not convex, it can provably be solved in linear time under mild conditions. The main steps of the method are given in Algorithm 6, which alternatively solves two sub-problems at each iteration by fixing one variable and solving for the other. The main difference here is that the arising sub-problems are constrained with the corresponding non-convex sets. The first problem is a projection on the space of low rank matrices and is given by hard thresholding its singular values (Eckart-Young-Mirsky theorem, see for instance [GVL12]). The second problem is a projection on the space of sparse matrices and is given by simply hard thresholding its elements.

Algorithm 6 Alternating Projections (AltProj)

Input: $\mathbf{D} \in \mathbb{R}^{m \times n}$, target rank r

- 1: Initialize $\mathbf{L}^{(0)} = \mathbf{0}$ and $\mathbf{S}^{(0)} = \mathcal{H}_{\zeta_0}(\mathbf{D} - \mathbf{L}^{(0)})$
- 2: **for** Stage $l \leftarrow 1$ to r **do**
- 3: **for** Iteration $k \leftarrow 0$ to T **do**
- 4: // Solve $\arg \min_{\mathbf{L}: \text{rank}(\mathbf{L}) \leq l} \|\mathbf{D} - \mathbf{L} - \mathbf{S}^{(k)}\|_2^2$
- 5: $\mathbf{M}^{(k)} \leftarrow \mathbf{D} - \mathbf{S}^{(k)}$
- 6: $(\mathbf{U}, \Sigma, \mathbf{V}) \leftarrow \text{SVD}(\mathbf{M}^{(k)}, l)$
- 7: $\mathbf{L}^{(k+1)} \leftarrow \mathbf{U} \mathcal{H}_l[\Sigma] \mathbf{V}^\top$
- 8: // Solve $\arg \min_{\mathbf{S}: \|\mathbf{S}\|_0 \leq \zeta} \|\mathbf{D} - \mathbf{L}^{(k+1)} - \mathbf{S}\|_2^2$
- 9: Update threshold ζ as in [NNS⁺14]
- 10: $\mathbf{S}^{(k+1)} \leftarrow \mathcal{H}_\zeta[\mathbf{D} - \mathbf{L}^{(k+1)}]$
- 11: **if** $\sigma_{l+1}(\mathbf{L}^{(k+1)}) < \epsilon$ **then**
- 12: **return** $(\mathbf{L}^{(T)}, \mathbf{S}^{(T)})$
- 13: **else**
- 14: $\mathbf{S}^{(0)} \leftarrow \mathbf{S}^{(T)}$
- 15: **end if**
- 16: **end for**
- 17: **end for**
- 18: **return** $(\mathbf{L}^{(T)}, \mathbf{S}^{(T)})$

Here $\text{SVD}(\mathbf{M}, l)$ returns the first l singular values of $\mathbf{M}$ with corresponding singular vectors. Although Algorithm 6 requires only l singular values at stage $l = 1, \dots, r$, SVD operations are still the computational bottleneck.

2.4 Multilevel Optimisation Algorithms

The algorithms proposed in this thesis are based on multilevel optimisation methods. They take advantage of the structure present in many applications and in image processing in particular. The proposed methodology uses the fact that many applications that give rise to large scale models can be modelled using varying degrees of fidelity. The typical applications of convex composite optimisation include dictionaries for learning problems, images/videos for computer vision applications, or discretisation of infinite dimensional problems appearing in image processing [AK06]. We propose to use the solution of a low dimensional model to generate the next step of various iterative schemes.

2.4.1 Introduction to Multilevel Optimisation Algorithms

A multilevel method for solving convex infinite-dimensional optimisation problems was introduced in [BDH99] and later further developed by Nash in [Nas00]. Although [Nas00] is one of the pioneering works that uses a multilevel algorithm in an optimisation context, it is rather abstract and only gives a general idea of how an optimisation algorithm can be used in a multilevel way. The author proves that the algorithm converges under certain mild conditions. Extending the key ideas of Nash's multilevel method in [WG09], Wen and Goldfarb used it in combination with a line search algorithm for solving discretized versions of unconstrained infinite-dimensional smooth optimisation problems. The main idea is to use the solution obtained from a coarse level for calculating the search direction in the fine level. On each iteration the algorithm uses either the negative gradient direction on the current level or a direction obtained from coarser level models. They prove that either search direction is a descent direction using Wolfe-Armijo and Goldstein-like stopping rules in their backtracking line search procedure. Later a multilevel algorithm using the trust region framework was proposed in [GST08]. In all those works the objective function is assumed to be smooth, whereas the problems

addressed in this thesis are not. We also note that multilevel optimisation methods have been applied to PDE optimisation for many years. A comprehensive review of the latter literature appeared in [BS09].

The first multilevel optimisation algorithm covering the non-smooth convex composite case was introduced in [PLRR14]. It is a multilevel version of the well-known Iterative Shrinkage-Thresholding Algorithm (ISTA) ([BT09b], [CDVLL98], [DDDM04]), called MISTA. In [PLRR14] the authors prove R-linear convergence rate of the algorithm and demonstrate in several image deblurring examples that MISTA can be up to 3-4 times faster than state of the art methods. However, MISTA's R-linear convergence rate is not optimal for first order methods and thus it has the potential for even better performance.

Motivated by this possible improvement, we applied Nesterov's acceleration techniques on MISTA, and show that it is possible to achieve $\mathcal{O}(1/\sqrt{\epsilon})$ convergence rate, where ϵ denotes the desired accuracy. To the best of our knowledge this is the first multilevel method that has an optimal rate. In addition to the proof that our method is optimal, we also show in numerical experiments that, for large-scale problems it can be up to 10 times faster than the current state of the art. These results in turn motivated us to apply the multilevel approach on other optimisation algorithms, namely we developed multilevel algorithms for the inexact augmented Lagrange multipliers [LCM10], Frank-Wolfe [MZWG16] and non-convex alternating projections [MZWG16] methods. For these methods as well we showed (approximate) convergence results, as well as their practical superiority over the standard methods.

2.4.2 Coarse Model Construction

We use the index H to indicate the coarse level of a multilevel method: $\mathbf{x}_H \in \mathbb{R}^{n_H}$ is the coarse level variable, $f_H(\cdot) : \mathbb{R}^{n_H} \rightarrow (-\infty, \infty]$ and $g_H(\cdot) : \mathbb{R}^{n_H} \rightarrow (-\infty, \infty]$ are the corresponding coarse surrogates of $f(\cdot)$ and $g(\cdot)$, respectively. In theory, these functions only need to satisfy a few very general assumptions, but of course, in practice they should reflect the structure of the problem, so that the coarse model is a meaningful smaller sized representation of the original problem. In all examples

studied in this thesis we used fairly straightforward coarse models: we take linear combinations of the columns of the corresponding matrix.

Throughout this thesis we use linear restriction and prolongation operators $\mathbf{R} : \mathbb{R}^{n \times n_H}$ and $\mathbf{P} : \mathbb{R}^{n_H \times n}$ respectively, where $n_H < n$ is the size of the coarse level variable. The restriction operator $\mathbf{R}$ transfers the current fine level point to the coarse level and the prolongation operator $\mathbf{P}$ constructs a point in for the fine level from the coarse level solution. The techniques we use are standard (see [BM⁺00]) so we keep this section brief.

There are many ways to design $\mathbf{R}$ and $\mathbf{P}$ operators, but they must always satisfy the following condition to guarantee the convergence of the algorithm,

$$\sigma \mathbf{P} = \mathbf{R}^T,$$

with some scalar $\sigma > 0$. Without loss of generality we assume $\sigma = 1$, which is a standard assumption in multilevel literature [BM⁺00], [Nas00].

In all applications we use a simple interpolation operator given in the form of a standard full weighting operator [BM⁺00] as a restriction operator:

$$\mathbf{R}_x = \frac{1}{4} \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & \dots & 0 \\ & & & \dots & & & & 0 \\ 0 & \dots & & 0 & 0 & 1 & 2 & 1 \end{bmatrix} \in \mathbb{R}^{\frac{n_i}{2} \times n_i} \quad (2.30)$$

and its transpose for the prolongation operator. Since multilevel methods often use more than one level constructed by applying the corresponding restriction and prolongation operators multiple times. Indeed, assume that $\mathbf{I}_h^H \in \mathbb{R}^{H \times h}$ is a linear operator that transfers $\mathbf{x}$ from $\mathbf{R}^h$ to $\mathbf{R}^H$. Now, we can define our restriction operator as $\mathbf{R} = \prod_{i=1}^l \mathbf{I}_{h_i}^{H_i}$, where $h_1 = n$, $h_{i+1} = H_i$ and $H_l = n_H$. Accordingly, the prolongation operator will be $\mathbf{P} = \prod_{i=l}^1 \mathbf{I}_{H_i}^{h_i}$, where $\sigma_i \mathbf{I}_{H_i}^{h_i} = (\mathbf{I}_{h_i}^{H_i})^\top$. Note that this construction satisfies the assumption that $\sigma \mathbf{P} = \mathbf{R}^\top$, with $\sigma = \prod_{i=1}^l \sigma_i$.

Chapter 3

MAGMA

3.1 Introduction

In this chapter we propose to speed up first order methods by taking advantage of the structure present in many applications and in image processing in particular. The proposed methodology uses the fact that many applications that give rise to large scale models can be modelled using varying degrees of fidelity. The typical applications of convex composite optimization include dictionaries for learning problems, images for computer vision applications, or discretization of infinite dimensional problems appearing in image processing [AK06]. First order methods use a quadratic model with first order information and the Lipschitz constant to construct a search direction. We propose to use the solution of a low dimensional (not necessarily quadratic) model to compute a search direction. The low dimensional model is not just a low fidelity model of the original model but it is constructed in a way so that is consistent with the high fidelity model.

One very popular recent approach for handling very large scale problems are stochastic coordinate descent methods [FR15, LLX15]. However, while being very effective for ℓ_1 -regularized least squares problems in general, stochastic gradient methods are less applicable for problems with highly correlated data, such as the face recognition problem and other pattern recognition problems, as well as in dense error correction problems with highly correlated dictionaries [WM10]. We compare our method both with accelerated gradient methods (FISTA) as well as stochastic block-coordinate meth-

ods (APCG [FR15]).

Our convergence proof is based on the proof of Nesterov’s method given in [AZO14], where the authors showed that optimal gradient methods can be viewed as a linear combination of primal gradient and mirror descent steps. We show that for some iterations (especially far away from the solution) it is beneficial to replace gradient steps with coarse correction steps. The coarse correction steps are computed by iterating on a carefully designed coarse model and a different step-size strategy. Since our algorithm combines concepts from multilevel optimization, gradient and mirror descent steps we call it Multilevel Accelerated Gradient Mirror Descent Algorithm (MAGMA). Furthermore, for the purposes of establishing a step-size strategy for MAGMA it is convenient to work with smooth problems. Thus we combine the theoretical superiority of non-smooth methods with the rich mathematical structure of smooth models to derive a step size strategy that guarantees convergence. MAGMA does not use smoothing in order to solve the original problem. Instead, it uses a smoothed version of the original model to compute a step size when the search direction is given by the coarse model (coarse correction step).

The proposed method converges in function value with a faster rate than any other existing multilevel method for convex problems, and in addition has the same rate as accelerated methods which is known to be optimal for first order methods [NYD82]. However, given that we use the additional structure of problems that appear in imaging applications in practice our algorithm is several times faster than the state of the art.

The rest of this chapter is structured as follows: Section 3.2 is devoted to presenting a multilevel model as well as our multilevel accelerated algorithm with its convergence proof. Then, in Section 3.3 we present the numerical results from experiments on large-scale face recognition problems. Mathematical definitions and algorithms most related to MAGMA can be found in Section 2.1.

3.2 Multilevel Accelerated Proximal Gradient Algorithm

In this section we formally present our proposed algorithm within the multilevel optimization setting, together with the proof of its convergence with $\mathcal{O}(1/\sqrt{\epsilon})$ rate, where ϵ is the convergence precision.

3.2.1 Model Construction

First we present the notation and construct a mathematical model, which will later be used in our algorithm for solving (2.1). A well-defined and converging multilevel algorithm requires appropriate information transfer mechanisms in between levels and a well-defined and coherent coarse model. For the information transfer we use the same restriction and prolongation operators as described in Section 2.4.2.

For the algorithm proposed in this section we do not use the direct prolongation of the coarse level solution to obtain a solution for the fine level problem. Instead, we use these coarse solutions to construct search directions for the fine level iterations. Therefore, in order to ensure the convergence of the algorithm it is essential to ensure that the fine and coarse level problems are coherent in terms of their gradients [Nas00, WG09].

However, in our setting the objective function is not smooth. To overcome this in a consistent way, we construct smooth coarse models, as well as use the smoothed version of the fine problem to find appropriate step-sizes¹. Therefore, we make the following natural assumptions on the coarse model.

Assumption 1. *For each coarse model constructed from a fine level problem (2.1) it holds that*

1. *The coarse level problem is solvable, i.e. bounded from below and the minimum is attained, and*
2. *$f_H(\mathbf{x})$ and $g_H(\mathbf{x})$ are both continuous, closed, convex and differentiable with Lipschitz continuous gradients.*

On the fine level, we say that iteration k is a coarse step, if the search direction is obtained by coarse correction. To denote the j -th iteration of the coarse model, we use $\mathbf{x}_{H,j}$.

A key property of the coarse model is that at its initial point $\mathbf{x}_{H,0}$ the optimality conditions of the two models match. This is achieved by adding a linear term to the coarse objective function:

$$F_H(\mathbf{x}_H) = f_H(\mathbf{x}_H) + g_H(\mathbf{x}_H) + \langle \mathbf{v}_H, \mathbf{x}_H \rangle, \quad (3.1)$$

¹We give more details about smoothing non-smooth functions in general, and $\|\cdot\|_1$ in particular, at the end of this section.

where the vector $\mathbf{v}_H \in \mathbb{R}^{n_H}$ is defined so that the fine and coarse level problems are first order coherent, that is, their first order optimality conditions coincide.

In this setting, however, we have a non-smooth objective function and assume the coarse model is smooth with L_H -Lipschitz continuous gradient, which can be done by using the smoothing technique described in subsection 2.1.4. For the face recognition problem (and (P_1) in general), where $g(\mathbf{x}) = \lambda \|\mathbf{x}\|_1$, it can be shown [BT12] that

$$g_\mu(\mathbf{x}) = \lambda \sum_{j=1}^n \sqrt{\mu^2 + \mathbf{x}_j^2} \quad (3.2)$$

is a μ -smooth approximation of $g(\mathbf{x}) = \lambda \|\mathbf{x}\|_1$ with parameters $(\lambda, \lambda n, 0)$ [BT12].

Furthermore, we construct $\mathbf{v}_H$ such that the gradient of the coarse model is equal to the gradient of the smoothed fine model's gradient:

$$\nabla F_H(\mathbf{R}\mathbf{x}) = \mathbf{R} \nabla F_\mu(\mathbf{x}), \quad (3.3)$$

where $F_\mu(\mathbf{x})$ is a μ -smooth approximation of $F(\mathbf{x})$ with parameters (a, β, K) . Note that for the composite problem (2.1) $F_\mu(\mathbf{x}) = f(\mathbf{x}) + g_\mu(\mathbf{x})$, where g_μ is a μ -smooth approximation of $g(\mathbf{x})$. In our experiments for (P_1) we use $g_\mu(\mathbf{x})$ as given in (3.2). The next lemma gives a choice for $\mathbf{v}_H$, such that (3.3) is satisfied.

Lemma 5 (Lemma 3.1 of [WG09]). *Let F_H be a Lipschitz continuous function with L_H Lipschitz constant, then for*

$$\mathbf{v}_H = \mathbf{R} \nabla F_\mu(\mathbf{x}) - (\nabla f_H(\mathbf{R}\mathbf{x}) + \nabla g_H(\mathbf{R}\mathbf{x})) \quad (3.4)$$

we have

$$\nabla F_H(\mathbf{R}\mathbf{x}) = \mathbf{R} \nabla F_\mu(\mathbf{x}).$$

The condition (3.3) is referred to as *first order coherence*. It ensures that if $\mathbf{x}$ is optimal for the smoothed fine level problem, then $\mathbf{R}\mathbf{x}$ is also optimal in the coarse level.

While in practice it is often beneficial to use more than one levels, for the theoretical analysis of our

algorithm without loss of generality (see Section 2.4.2) we assume that there is only one coarse level.

3.2.2 MAGMA

In this subsection we describe our proposed multilevel accelerated proximal algorithm for solving (2.1). We call it MAGMA for Multilevel Accelerated Gradient Mirror descent Algorithm. As in [Nes05] and [AZO14], at each iteration MAGMA performs both gradient and mirror descent steps, then uses a convex combination of their results to compute the next iterate. The crucial difference here is that whenever a coarse iteration is performed, our algorithm uses the coarse direction

$$\mathbf{y}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k(\mathbf{x}_k),$$

instead of the gradient, where $\mathbf{d}_k(\cdot)$ is the search direction and s_k is an appropriately chosen step-size. Next we describe how each of these terms is constructed.

At each iteration k the algorithm first decides whether to use the gradient or the coarse model to construct a search direction for the gradient descent step. This decision is based on the optimality condition at the current iterate $\mathbf{x}_k$: we do not want to use the coarse direction when it is not helpful, i.e. when

- the first order optimality conditions are almost satisfied, or
- the current point $\mathbf{x}_k$ is very close to the point $\tilde{\mathbf{x}}$, where a coarse iteration was last performed, as long as the algorithm has not performed too many gradient correction steps.

More formally, we choose the coarse direction, whenever both of the following conditions are satisfied:

$$\begin{aligned} \|\mathbf{R}\nabla F_{\mu_k}(\mathbf{x}_k)\|_* &> \kappa \|\nabla F_{\mu_k}(\mathbf{x}_k)\|_*, \\ \|\mathbf{x}_k - \tilde{\mathbf{x}}\| &> \vartheta \|\tilde{\mathbf{x}}\| \quad \text{or} \quad q < K_d, \end{aligned} \tag{3.5}$$

where $\kappa \in (0, \min(1, \min \|\mathbf{R}\|))$, $\vartheta > 0$ and K_d are predefined constants, and q is the number of consecutive gradient correction steps [WG09], [PLRR14].

If a coarse direction is chosen, a coarse model is constructed as described in (3.1). In order to satisfy the coherence property (3.3), we start the coarse iterations with $\mathbf{x}_{H,0} = \mathbf{R}\mathbf{x}_k$, then solve the coarse model by a first order method and construct a search direction:

$$\mathbf{d}_k(\mathbf{x}_k) = \mathbf{P}(\mathbf{x}_{H,N_H} - \mathbf{R}\mathbf{x}_k), \quad (3.6)$$

where $\mathbf{x}_{H,N_H} \in \mathbb{R}^{n_H}$ is an approximate solution of the coarse model after N_H iterations. Note that in practice we do not find the exact solution, but rather run the algorithm for N_H iterations, until we achieve a certain acceptable $\mathbf{x}_{H,N_H}$. In our experiments we used the monotone version of FISTA [BT09a], however in theory any monotone algorithm will ensure convergence.

We next show that any coarse direction defined in (3.6) is a descent direction for the smoothed fine problem.

Lemma 6 (Descent Direction). *If at iteration k a coarse step is performed and suppose that $F_H(\mathbf{x}_{H,N_H}) < F_H(\mathbf{x}_{H,1})$, then for any $\mathbf{x}_k$ it holds that*

$$\langle \mathbf{d}_k(\mathbf{x}_k), \nabla F_{\mu_k}(\mathbf{x}_k) \rangle < -\frac{\kappa^2}{2L_H} \|\nabla F_{\mu_k}(\mathbf{x}_k)\|_*^2 \leq 0.$$

Proof. Using the definition of coarse direction $\mathbf{d}(\mathbf{x}_k)$, linearity of $\mathbf{R}$ and Lemma 5 we obtain

$$\begin{aligned} \langle \mathbf{d}_k(\mathbf{x}_k), \nabla F_{\mu_k}(\mathbf{x}_k) \rangle &= \langle \mathbf{R}^\top (\mathbf{x}_{H,N_H} - \mathbf{x}_{H,0}), \nabla F_{\mu_k}(\mathbf{x}_k) \rangle \\ &= \langle \mathbf{x}_{H,N_H} - \mathbf{x}_{H,0}, \mathbf{R} \nabla F_{\mu_k}(\mathbf{x}_k) \rangle \\ &= \langle \mathbf{x}_{H,N_H} - \mathbf{x}_{H,0}, \nabla F_H(\mathbf{x}_{H,0}) \rangle \\ &\leq F_H(\mathbf{x}_{H,N_H}) - F_H(\mathbf{x}_{H,0}), \end{aligned} \quad (3.7)$$

where for the last inequality we used the convexity of F_H . On the other hand using the monotonicity assumption on the coarse level algorithm and lemma (1) we derive:

$$F_H(\mathbf{x}_{H,N_H}) - F_H(\mathbf{x}_{H,0}) < F_H(\mathbf{x}_{H,1}) - F_H(\mathbf{x}_{H,0}) \leq -\frac{1}{2L_H} \|\nabla F_H(\mathbf{x}_{H,0})\|_*^2. \quad (3.8)$$

Now using the choice $\mathbf{x}_{H,0} = \mathbf{R}\mathbf{x}_k$ and (3.3) we obtain:

$$\nabla F_H(\mathbf{x}_{H,0}) = \nabla F_H(\mathbf{R}\mathbf{x}_k) = \mathbf{R}\nabla F_{\mu_k}(\mathbf{x}_k). \quad (3.9)$$

Then combining (3.8), (3.9) and condition (3.5) we obtain

$$F_H(\mathbf{x}_{H,N_H}) - F_H(\mathbf{x}_{H,0}) < -\frac{1}{2L_H}\|\nabla \mathbf{R}F_{\mu_k}(\mathbf{x}_k)\|^2 \leq -\frac{\kappa^2}{2L_H}\|\nabla F_{\mu_k}(\mathbf{x}_k)\|^2. \quad (3.10)$$

Finally, combining (3.7) and (3.10) we derive the desired result. $\square$

After constructing a descent search direction, we find suitable step sizes for both gradient and mirror steps at each iteration. In [AZO14] a fixed step size of $\alpha_{k+1} = \frac{1}{L_f}$ is used for gradient steps. However, in our algorithm we do not always use the steepest descent direction for gradient steps, and therefore another step size strategy is required. We show that step sizes obtained from Armijo-type backtracking line search on the smoothed (fine) problem ensures that the algorithm converges with optimal rate. Starting from a predefined $s_k = s^0$, we reduce it by a factor of constant $\tau \in (0, 1)$ until

$$F_{\mu_k}(\mathbf{y}_{k+1}) \leq F_{\mu_k}(\mathbf{x}_k) + cs_k \langle \mathbf{d}_k(\mathbf{x}_k), \nabla F_{\mu_k}(\mathbf{x}_k) \rangle \quad (3.11)$$

is satisfied for some predefined constant $c \in (0, 1)$.

For mirror steps we always use the current gradient as search direction, as they have significant contribution to the convergence only when the gradient is small enough, meaning we are close to the minimizer, and in those cases we do not perform coarse iterations. However, we need to adjust the step size α_k for mirror steps in order to ensure convergence. Next we formally present the proposed algorithm.

Now we show the convergence of Algorithm 7 with $\mathcal{O}(1/\sqrt{\epsilon})$ rate. First we prove an analogue of Lemma 3.

Lemma 7. *For every $\mathbf{u} \in \mathbb{R}^n$ and $\eta_k > 0$ and*

Algorithm 7 MAGMA

Input: MAGMA($f(\cdot), g(\cdot), \mathbf{x}_0, T$)

```
1:  $\mathbf{y}_0 \leftarrow \mathbf{z}_0 = \mathbf{x}_0$ 
2:  $\alpha_0 \leftarrow 0$ 
3: for  $k = 0, 1, 2, \dots, T$  do
4:   Choose  $\eta_{k+1}$  and  $\alpha_{k+1}$  according to (3.19) and (3.20)
5:    $t_k \leftarrow \frac{1}{\alpha_{k+1}\eta_{k+1}}$ 
6:    $\mathbf{x}_k \leftarrow t_k \mathbf{z}_k + (1 - t_k) \mathbf{y}_k$ 
7:   if Conditions (3.5) are satisfied then
8:     Perform  $N_H$  coarse iterations
9:      $\mathbf{d}_k \leftarrow \mathbf{P}(\mathbf{x}_{H,N_H} - \mathbf{R}\mathbf{x}_k)$ 
10:     $\mathbf{y}_{k+1} \leftarrow \mathbf{x}_k + s_k \mathbf{d}(\mathbf{x}_k)$ 
11:    with  $s_k$  satisfying (3.11)
12:   else
13:      $\mathbf{y}_{k+1} \leftarrow \text{prox}(\mathbf{x}_k)$ 
14:   end if
15:    $\mathbf{z}_{k+1} \leftarrow \text{Mirr}_{\mathbf{z}_k}(\nabla f(\mathbf{x}_k), \alpha_{k+1})$ 
16: end for
```

$$\eta_{k+1} = \begin{cases} L_f, & \text{when } k+1 \text{ is a gradient correction step} \\ \max \left\{ \frac{1}{4\alpha_k^2\eta_k}, \frac{L_H}{cs_k\kappa^2} \right\}, & \text{otherwise} \end{cases}$$

it holds that

$$\begin{aligned} & \alpha_{k+1} \langle \nabla f(\mathbf{x}_k), \mathbf{z}_k - \mathbf{u} \rangle + \alpha_{k+1} (g(\mathbf{z}_k) - g(\mathbf{u})) \\ & \leq \alpha_{k+1}^2 L_f \text{Prog}(\mathbf{x}_k) + V_{\mathbf{z}_k}(\mathbf{u}) - V_{\mathbf{z}_{k+1}}(\mathbf{u}) \\ & \leq \alpha_{k+1}^2 [\eta_{k+1} (F_{\mu_k}(\mathbf{x}_k) - F_{\mu_k}(\mathbf{y}_{k+1})) + L_f \beta \mu_k] + V_{\mathbf{z}_k}(\mathbf{u}) - V_{\mathbf{z}_{k+1}}(\mathbf{u}) \end{aligned}$$

Proof. First note that the proof of the first inequality follows directly from Lemma 4.2 of [AZO14].

Now assume k is a coarse step, then the first inequality follows from Lemma 2. To show the second

one we first note that for $\tilde{\mathbf{x}} = \text{prox}(\mathbf{x}_k)$,

$$\begin{aligned}
& \text{Prog}(\mathbf{x}_k) - \frac{1}{2L_f} \|\nabla F_{\mu_k}(\mathbf{x}_k)\|^2 \\
&= -\frac{L_f}{2} \|\tilde{\mathbf{x}} - \mathbf{x}_k\|^2 - \langle \nabla f(\mathbf{x}_k), \tilde{\mathbf{x}} - \mathbf{x}_k \rangle - g(\tilde{\mathbf{x}}) + g(\mathbf{x}_k) - \frac{1}{2L_f} \|\nabla F_{\mu_k}(\mathbf{x}_k)\|^2 \\
&\leq -\frac{L_f}{2} \|\tilde{\mathbf{x}} - \mathbf{x}_k\|^2 - \langle \nabla f(\mathbf{x}_k), \tilde{\mathbf{x}} - \mathbf{x}_k \rangle - \langle \tilde{\mathbf{x}} - \mathbf{x}_k, \nabla g_{\mu_k}(\mathbf{x}_k) \rangle + \beta\mu_k - \frac{1}{2L_f} \|\nabla F_{\mu_k}(\mathbf{x}_k)\|^2 \\
&= -\frac{L_f}{2} (\|\tilde{\mathbf{x}} - \mathbf{x}_k\|^2 + 2\langle \tilde{\mathbf{x}} - \mathbf{x}_k, \frac{1}{L_f} \nabla F_{\mu_k}(\mathbf{x}_k) \rangle + \|\frac{1}{L_f} \nabla F_{\mu_k}(\mathbf{x}_k)\|^2) + \beta\mu_k \\
&\leq \beta\mu_k.
\end{aligned}$$

Here we used the definitions of Prog, prox and g_μ , as well as the convexity of g_μ . Therefore from Lemma 6 and backtracking condition (3.11) we obtain that if k is a coarse correction step, then

$$\begin{aligned}
L_f \text{Prog}(\mathbf{x}_k) &\leq \frac{1}{2} \|\nabla F_{\mu_k}(\mathbf{x}_k)\|^2 + L_f \beta\mu_k \\
&\leq -\frac{L_H}{\kappa^2} \langle \mathbf{d}_k(\mathbf{x}_k), \nabla F_{\mu_k}(\mathbf{x}_k) \rangle + L_f \beta\mu_k \\
&\leq \frac{L_H}{c s_k \kappa^2} (F_{\mu_k}(\mathbf{x}_k) - F_{\mu_k}(\mathbf{y}_{k+1})) + L_f \beta\mu_k,
\end{aligned}$$

Otherwise, if k is a gradient correction step, then from Lemma 1

$$L_f \text{Prog}(\mathbf{x}_k) \leq L_f (F(\mathbf{x}_k) - F(\mathbf{y}_{k+1})) \leq L_f (F_{\mu_k}(\mathbf{x}_k) - F_{\mu_k}(\mathbf{y}_{k+1})) + L_f \beta\mu_k.$$

Now choosing

$$\eta_{k+1} = \begin{cases} L_f, & \text{when } k+1 \text{ is a gradient correction step} \\ \max \left\{ \frac{1}{4\alpha_k^2 \eta_k}, \frac{L_H}{c s_k \kappa^2} \right\}, & \text{otherwise} \end{cases}$$

we obtain the desired result. $\square$

Remark. The recurrent choice of η_{k+1} may seem strange at this point, as we could simply set it to $\max\{\frac{L_H}{c s_k \kappa^2}, L_f\}$, however forcing $\eta_{k+1} \geq \frac{1}{4\alpha_k^2 \eta_k}$ helps us ensure that $t_k \in (0, 1]$ later.

Lemma 8 (Coupling). For any $\mathbf{u} \in \mathbb{R}^n$ and $t_k = 1/\alpha_{k+1}\eta_{k+1}$, where η_{k+1} is defined as in Lemma 7,

it holds that

$$\begin{aligned} \alpha_{k+1}^2 \eta_{k+1} F(\mathbf{y}_{k+1}) - (\alpha_{k+1}^2 \eta_{k+1} - \alpha_{k+1}) F(\mathbf{y}_k) + (V_{\mathbf{z}_{k+1}}(\mathbf{u}) - V_{\mathbf{z}_k}(\mathbf{u})) \\ \leq \alpha_{k+1} F(\mathbf{u}) + (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k. \end{aligned} \quad (3.12)$$

Proof. First note that, if k is a gradient correction step, then the proof follows from Lemma 4. Now assume k is a coarse correction step, then using the convexity of f we obtain

$$\begin{aligned} F(\mathbf{x}_k) - F(\mathbf{u}) &= f(\mathbf{x}_k) - f(\mathbf{u}) + g(\mathbf{x}_k) - g(\mathbf{u}) \\ &\leq \langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{u} \rangle + g(\mathbf{x}_k) - g(\mathbf{u}). \end{aligned} \quad (3.13)$$

On the other hand, from the definition of g_{μ_k} and its convexity we have

$$\begin{aligned} g(\mathbf{x}_k) - g(\mathbf{z}_k) &\leq g_{\mu_k}(\mathbf{x}_k) + \beta_1 \mu_k - g_{\mu_k}(\mathbf{z}_k) + \beta_2 \mu_k \\ &\leq \langle \nabla g_{\mu_k}(\mathbf{x}_k), \mathbf{x}_k - \mathbf{z}_k \rangle + \beta \mu_k. \end{aligned} \quad (3.14)$$

Then using (3.14), we can rewrite $g(\mathbf{x}_k) - g(\mathbf{u})$ as

$$\begin{aligned} g(\mathbf{x}_k) - g(\mathbf{u}) &= (g(\mathbf{x}_k) - g(\mathbf{z}_k)) + (g(\mathbf{z}_k) - g(\mathbf{u})) \\ &\leq \langle \nabla g_{\mu_k}(\mathbf{x}_k), \mathbf{x}_k - \mathbf{z}_k \rangle + \beta \mu_k + (g(\mathbf{z}_k) - g(\mathbf{u})). \end{aligned} \quad (3.15)$$

Now rewriting $\langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{u} \rangle = \langle \nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{z}_k \rangle + \langle \nabla f(\mathbf{x}_k), \mathbf{z}_k - \mathbf{u} \rangle$ in (3.13) and using (3.15) we obtain

$$F(\mathbf{x}_k) - F(\mathbf{u}) \leq \langle \nabla F_{\mu_k}(\mathbf{x}_k), \mathbf{x}_k - \mathbf{z}_k \rangle + \beta \mu_k + \langle \nabla f(\mathbf{x}_k), \mathbf{z}_k - \mathbf{u} \rangle + (g(\mathbf{z}_k) - g(\mathbf{u})), \quad (3.16)$$

where we used that $\nabla F_{\mu_k}(\mathbf{x}_k) = \nabla f(\mathbf{x}_k) + \nabla g_{\mu_k}(\mathbf{x}_k)$.

From the choice $t_k(\mathbf{x}_k - \mathbf{z}_k) = (1 - t_k)(\mathbf{y}_k - \mathbf{x}_k)$ and convexity of F_{μ_k} we have

$$\begin{aligned}\langle \nabla F_{\mu_k}(\mathbf{x}_k), \mathbf{x}_k - \mathbf{z}_k \rangle &= \frac{(1 - t_k)}{t_k} \langle \nabla F_{\mu_k}(\mathbf{x}_k), \mathbf{y}_k - \mathbf{x}_k \rangle \\ &\leq \frac{(1 - t_k)}{t_k} (F_{\mu_k}(\mathbf{y}_k) - F_{\mu_k}(\mathbf{x}_k)) \\ &\leq \frac{(1 - t_k)}{t_k} (F(\mathbf{y}_k) - F(\mathbf{x}_k)) + \frac{(1 - t_k)}{t_k} \beta \mu_k,\end{aligned}$$

where for the last inequality we used that F_{μ_k} is a μ_k -smooth approximation of F . Then choosing $t_k = 1/(\alpha_{k+1}\eta_{k+1})$, where α_{k+1} is a step-size for the Mirror Descent step and η_{k+1} is defined in Lemma 7, we have

$$\langle \nabla F_{\mu_k}(\mathbf{x}_k), \mathbf{x}_k - \mathbf{z}_k \rangle \leq (\alpha_{k+1}\eta_{k+1} - 1)(F(\mathbf{y}_k) - F(\mathbf{x}_k)) + (\alpha_{k+1}\eta_{k+1} - 1)\beta\mu_k. \quad (3.17)$$

Plugging (3.17) into (3.16) we obtain

$$\begin{aligned}\alpha_{k+1}(F(\mathbf{x}_k) - F(\mathbf{u})) &\leq (\alpha_{k+1}^2\eta_{k+1} - \alpha_{k+1})(F(\mathbf{y}_k) - F(\mathbf{x}_k)) + \alpha_{k+1}^2\eta_{k+1}\beta\mu_k \\ &\quad + \alpha_{k+1}\langle \nabla f(\mathbf{x}_k), \mathbf{z}_k - \mathbf{u} \rangle + \alpha_{k+1}(g(\mathbf{z}_k) - g(\mathbf{u})).\end{aligned} \quad (3.18)$$

Finally, we apply Lemma 7 on (3.18) and obtain the desired result after simplifications:

$$\begin{aligned}\alpha_{k+1}(F(\mathbf{x}_k) - F(\mathbf{u})) &\leq (\alpha_{k+1}^2\eta_{k+1} - \alpha_{k+1})(F(\mathbf{y}_k) - F(\mathbf{x}_k)) \\ &\quad + \alpha_{k+1}^2[\eta_{k+1}(F(\mathbf{x}_k) - F(\mathbf{y}_{k+1})) + (L_f + \eta_{k+1})\beta\mu_k] + V_{\mathbf{z}_k}(\mathbf{u}) - V_{\mathbf{z}_{k+1}}(\mathbf{u}).\end{aligned}$$

□

Theorem 3.1 (Convergence). *After T iterations of Algorithm 7, without loss of generality assuming that the last iteration is a gradient correction step for*

$$\eta_{k+1} = \begin{cases} L_f, & \text{when } k+1 \text{ is a gradient correction step} \\ \max\left\{\frac{1}{4\alpha_k^2\eta_k}, \frac{L_H}{c s_k \kappa^2}\right\}, & \text{otherwise} \end{cases} \quad (3.19)$$

and

$$\alpha_{k+1} = \begin{cases} \frac{k+2}{2L_f}, & \text{when } k+1 \text{ is a gradient correction step} \\ \frac{1}{2\eta_{k+1}} + \alpha_k \sqrt{\frac{\eta_k}{\eta_{k+1}}}, & \text{otherwise} \end{cases} \quad (3.20)$$

it holds that

$$F(\mathbf{y}_T) - F(\mathbf{x}^*) \leq \mathcal{O}\left(\frac{L_f}{T^2}\right),$$

Proof. By choosing η_{k+1} and α_{k+1} according to (3.19) and (3.20), we ensure that $\alpha_k^2 \eta_k = \alpha_{k+1}^2 \eta_{k+1} - \alpha_{k+1} + \frac{1}{4\eta_{k+1}}$ and $t_k = 1/(\alpha_{k+1} \eta_{k+1}) \in (0, 1]$. Then telescoping Lemma 8 with $k = 0, 1, \dots, T-1$ we obtain

$$\begin{aligned} \sum_{k=0}^{T-1} [\alpha_{k+1}^2 \eta_{k+1} F(\mathbf{y}_{k+1}) - \alpha_k^2 \eta_k F(\mathbf{y}_k) + \frac{1}{4\eta_{k+1}} F(\mathbf{y}_k)] + V_{\mathbf{z}_T}(\mathbf{u}) - V_{\mathbf{z}_0}(\mathbf{u}) \\ \leq \sum_{k=0}^{T-1} [\alpha_{k+1} F(\mathbf{u}) + (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k]. \end{aligned}$$

Or equivalently,

$$\begin{aligned} \alpha_T^2 \eta_T F(\mathbf{y}_T) - \alpha_0^2 \eta_0 F(\mathbf{y}_0) + \sum_{k=0}^{T-1} \frac{1}{4\eta_{k+1}} F(\mathbf{y}_k) + V_{\mathbf{z}_T}(\mathbf{u}) - V_{\mathbf{z}_0}(\mathbf{u}) \\ \leq \sum_{k=0}^{T-1} [\alpha_{k+1} F(\mathbf{u}) + (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k], \end{aligned}$$

Then choosing $\mathbf{u} = \mathbf{x}^*$ and noticing that $F(\mathbf{y}_k) \geq F(\mathbf{x}^*)$ and $V_{\mathbf{z}_T}(\mathbf{u}) \geq 0$, for an upper bound $\Theta \geq V_{\mathbf{z}_0}(\mathbf{x}^*)$, we obtain

$$\alpha_T^2 \eta_T F(\mathbf{y}_T) - \alpha_0^2 \eta_0 F(\mathbf{y}_0) \leq \Theta + \sum_{k=0}^{T-1} \left[\left(\alpha_{k+1} - \frac{1}{4\eta_{k+1}} \right) F(\mathbf{x}^*) + (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k \right].$$

Now using the fact that $\alpha_{k+1} - \frac{1}{4\eta_{k+1}} = \alpha_{k+1}^2 \eta_{k+1} - \alpha_k^2 \eta_k$, for $k = 0, 1, \dots, T-1$ we simplify to

$$\alpha_T^2 \eta_T F(\mathbf{y}_T) - \alpha_0^2 \eta_0 F(\mathbf{y}_0) \leq \Theta + \alpha_T^2 \eta_T F(\mathbf{x}^*) - \alpha_0^2 \eta_0 F(\mathbf{x}^*) + \sum_{k=0}^{T-1} (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k.$$

Then defining $\alpha_0 = 0$ and $\eta_0 = L_f$ and using the property of Bregman distances we obtain

$$\begin{aligned}\alpha_T^2 \eta_T (F(\mathbf{y}_T) - F(\mathbf{x}^*)) &\leq \Theta + \alpha_0^2 \eta_0 (F(\mathbf{y}_0) - F(\mathbf{x}^*)) + \sum_{k=0}^{T-1} (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k \\ &\leq \Theta + \frac{1}{L_f} + \sum_{k=0}^{T-1} (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k.\end{aligned}$$

Then assuming that the last iteration T is a gradient correction step² and using the definitions of α_T and η_T for the left hand side, we can further simplify to

$$\frac{(T+1)^2}{4L_f} (F(\mathbf{y}_T) - F(\mathbf{x}^*)) \leq \Theta + \sum_{k=0}^{T-1} (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k,$$

or equivalently

$$F(\mathbf{y}_T) - F(\mathbf{x}^*) \leq 4L_f \frac{\Theta + \sum_{k=0}^{T-1} (L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta \mu_k}{(T+1)^2}.$$

Finally, choosing $\mu_k \in \left(0, \frac{\zeta}{(L_f + \eta_{k+1}) \alpha_{k+1}^2 \beta T}\right]$ for a small predefined $\zeta \in (0, 1]$, we obtain:

$$F(\mathbf{y}_T) - F(\mathbf{x}^*) \leq \frac{4L_f(\Theta + \zeta)}{(T+1)^2}.$$

□

Remark. Clearly, the constant factor of our Algorithm's worst case convergence rate is not better than that of AGM, however, as we show in the next section, in practice MAGMA can be several times faster than AGM.

3.3 Numerical Experiments

In this section we demonstrate the empirical performance of our algorithm compared to FISTA [BT09b] and a variant of MISTA [PLRR14] on several large-scale face recognition problems. We chose those two particular algorithms, as MISTA is the only multilevel algorithm that can solve

²This assumption does not lose the generality, as we can always perform one extra gradient correction iteration

non-smooth composite problems and FISTA is the only other algorithm that was able to solve our large-scale problems in reasonable times. In Section 3.3.3 we also compare FISTA and MAGMA to two recent proximal stochastic algorithms: (APCG [LLX15]) and SVRG [XZ14]. We chose Face Recognition (FR) as a demonstrative application since (a) FR using large scale dictionaries is a relatively unexplored problem ³ and (b) the performance of large scale face recognition depends on the face resolution.

3.3.1 MAGMA for Solving Robust Face Recognition

In order to be able to use the multilevel algorithm, we need to define a coarse model and restriction and prolongation operators appropriate for the given problem. In this subsection we describe those constructions for the face recognition problem used in our numerical experiments.

Creating a coarse model requires decreasing the size of the original fine level problem. In our experiments we only reduce n , as we are trying to improve over the performance of first order methods, whose complexity per iteration is $\mathcal{O}(n^2)$. Also the columns of $\mathbf{A}$ are highly correlated, which means that reducing n loses little information about the original problem, whereas it is not necessarily true for the rows of $\mathbf{A}$. Therefore, we introduce the dictionary $\mathbf{A}_H \in \mathbb{R}^{m \times n_H}$ with $n_H < n$. More precisely, we set $\mathbf{A}_H = \mathbf{A} \mathbf{R}_x^\top$, with $\mathbf{R}_x$ defined in (2.30). With the given restriction operator we are ready to construct a coarse model for the problem (2.8):

$$\min_{\mathbf{w}_H \in \mathbb{R}^{m+n_H}} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{A}_H & \mathbf{I} \end{bmatrix} \mathbf{w}_H - \mathbf{b} \right\|_2^2 + \lambda \sum_{j=1}^{m+n_H} \sqrt{\mu_H^2 + \mathbf{w}_{H,j}^2} + \langle \mathbf{v}_H, \mathbf{w}_H \rangle, \quad (3.21)$$

where $\mathbf{w}_H = [\mathbf{x}_H^\top, \mathbf{e}^\top]^\top$ and $\mathbf{v}_H$ is defined in (3.4). It is easy to check that the coarse objective function and gradient can be evaluated using the following equations:

$$F_H(\mathbf{w}_H) = \frac{1}{2} \left\| \mathbf{A}_H \mathbf{x}_H + \mathbf{e} - \mathbf{b} \right\|_2^2 + \lambda \sum_{j=1}^{m+n_H} \sqrt{\mu_H^2 + \mathbf{w}_{H,j}^2} + \langle \mathbf{v}_H, \mathbf{w}_H \rangle,$$

³FR using large scale dictionaries is an unexplored problem in ℓ_1 optimization literature due to the complexity of the (P_1) problem.

$$\nabla F_H(\mathbf{w}_H) = \begin{bmatrix} \mathbf{A}_H^\top \mathbf{A}_H & \mathbf{A}_H^\top \\ \mathbf{A}_H & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_H \\ \mathbf{e} \end{bmatrix} - \begin{bmatrix} \mathbf{A}_H^\top \mathbf{b} \\ \mathbf{b} \end{bmatrix} + \nabla g_H(\mathbf{w}_H) + \mathbf{v}_H,$$

where $\nabla g_H(\mathbf{w}_H)$ is the gradient of g_H defined in (3.2) and is given elementwise as follows:

$$\frac{\lambda \mathbf{w}_{H,j}}{\sqrt{\mu_H^2 + \mathbf{w}_{H,j}^2}}, \forall j = 1, 2, \dots, n_H.$$

Hence we do not multiply matrices of size $m \times (m + n_H)$, but only $m \times n_H$.

We use a standard full weighting operator 2.30, which constructs linear combinations of the columns of dictionary $\mathbf{A}$ and its transpose for the prolongation operator. However, we do not apply those operators on the whole variable $\mathbf{w}$ of the model (2.8), as this would imply also reducing m . We rather apply the operators only on part $\mathbf{x}$ of $\mathbf{w} = [\mathbf{x}^\top, \mathbf{e}^\top]^\top$, therefore our restriction and prolongation operators are of the following forms: $\mathbf{R} = [\mathbf{R}_x, \mathbf{I}]$ and $\mathbf{P} = \mathbf{R}^\top = [\mathbf{R}_x^\top, \mathbf{I}]^\top$.

In all our experiments, for the Mirr operator we chose the standard Euclidean norm $\|\cdot\|_2$ and accordingly $\frac{1}{2}\|\cdot\|_2^2$ as a Bregman divergence.

3.3.2 Numerical Results

The proposed algorithm has been implemented in MatLab® and tested on a PC with Intel® Core™ i7 CPU (3.4GHz×8) and 15.6GB memory.

In order to create a large scale FR setting we created a dictionary $\mathbf{A}$ of up to 8,824 images from more than 4,000 different people ⁴ by merging images from several facial databases captured in controlled and in uncontrolled conditions. In particular, we used images from MultiPIE [GMC⁺10], XM2VTS [LM98] and many in-the-wild databases such as LFW [HRBLM07], LFPW [BJKK11] and HELEN [LBL⁺12]. In [YZB⁺13] the dictionary was small hence only very low-resolution images were considered. However, in large scale FR having higher resolution images is very beneficial (e.g., from a random cohort of 300 selected images, low resolution images of 40×30 achieved around 85%

⁴Some people had up to 5 facial images in the database some only one.

	440 images	944 images	8824 images
MISTA	7	2	2
MAGMA	7	7	13

Table 3.1: Number of coarse levels used for MISTA and MAGMA for each dictionary

recognition rate, while using images of 200×200 we went up to 100%). Hence, in the remaining of our experiments the dictionary used is $\begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix}$ of $40,000 \times 48,824$ and some subsets of this dictionary.

In order to show the improvements of our method with regards to FISTA as well as to MISTA, we have designed the following experimental settings

- Using 440 facial images, then $\begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix}$ is of $40,000 \times 40,440$
- Using 944 facial images, then $\begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix}$ is of $40,000 \times 40,944$
- Using 8,824 facial images, then $\begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix}$ is of $40,000 \times 48,824$

For each dictionary we randomly chose 20 different input images $\mathbf{b}$ that are not in the dictionary and ran the algorithms with 4 different random starting points. For all of those experiments we set the parameters of the algorithms as follows: $\kappa = 0.9, 0.8$ and 0.7 respectively for experiment settings with 440, 944 and 8,824 images in the database, $K_d = 30$ for the two smaller settings and $K_d = 50$ for the largest one, the convergence tolerance was set to $\epsilon = 10^{-6}$ for the two smaller experiments and $\epsilon = 10^{-7}$ for the largest one, $\lambda = 10^{-6}$, $\tau = 0.95$ and $s^0 = 10$. For larger dictionaries we slightly adjust κ and K_d in order to adjust for the larger problem size, giving the algorithm more freedom to perform coarse iterations. In all experiments we solve the coarse models with lower tolerance than the original problem, namely with tolerance 10^{-3} .

For all experiments we used as small coarse models as possible so that the corresponding algorithm could produce sparse solutions correctly identifying the sought faces in the dictionaries. Specifically, for experiments on the largest database with 8824 images we used 13 levels for MAGMA (so that in the coarse model $\mathbf{A}_H$ has only 2 columns) and only two levels for MISTA, since with more than two levels MISTA was unable to produce sparse solutions. The number of levels used in MAGMA and MISTA are tabulated in Table 3.1.

	440 images	944 images	8824 images
FISTA	98.69	175.77	1753
MISTA	68.77	70.4	1302
MAGMA	25.73	29.62	481

Table 3.2: Average CPU running times (seconds) of MISTA, FISTA and MAGMA

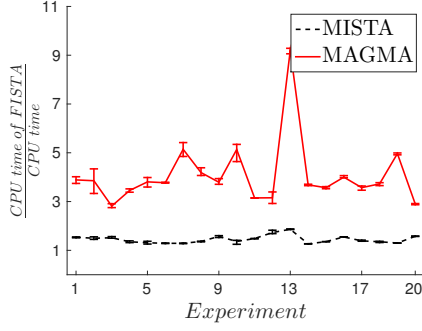
In all experiments we measure and compare the CPU running times of each tested algorithm, because comparing the number of iterations of a multilevel method against a standard first order algorithm would not be fair. This is justified as the multilevel algorithm may need fewer iterations on the fine level, but with a higher computational cost for obtaining each coarse direction. Comparing all iterations including the ones in coarser levels would not be fair either, as those are noticeably cheaper than the ones on the fine level. Even comparing the number of objective function values and gradient evaluations would not be a good solution, as on the coarse level those are also noticeably cheaper than on the fine level. Furthermore, the multilevel algorithm requires additional computations for constructing the coarse model, as well as for vector restrictions and prolongations. Therefore, we compare the algorithms with respect to CPU time.

Figures 3.1a, 3.1b and 3.1c show the relative CPU running times until convergence of MAGMA and MISTA compared to FISTA⁵. The horizontal axis indicate different input images b , whereas the vertical lines on the plots show the respective highest and lowest values obtained from different starting points. Furthermore, in Table 3.2 we present the average CPU times required by each algorithm until convergence for each problem setting, namely with databases of 440, 944 and 8824 images. As the experimental results show, MAGMA is 2 – 10 times faster than FISTA. On the other hand, MISTA is more problem dependant. On some instances it can be even faster than MAGMA, but on most cases its performance is somewhere in between FISTA and MAGMA.

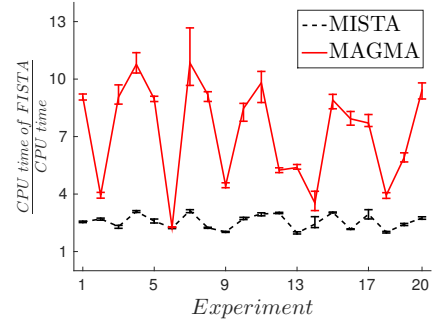
In order to better understand the experimental convergence speed of MAGMA, we measured the error of stopping criteria⁶ and objective function values generated by MAGMA and FISTA over time from an experiment with 440 images in the dictionary. The stopping criteria are log-log-plotted in Figure 3.2a and the objective function values - in Figure 3.2b. In all figures the horizontal axis is the CPU

⁵We only report the performance of FISTA, as AGM and FISTA demonstrate very similar performances on all experiments.

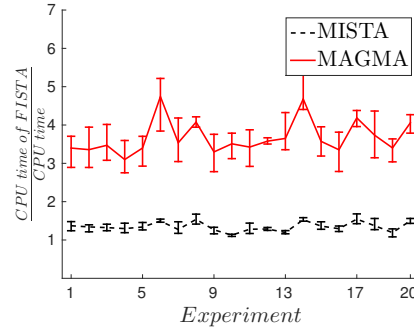
⁶We use the norm of the gradient mapping as a stopping criterion



(a) CPU times with 440 images



(b) CPU times with 944 images



(c) CPU times with 8824 images

Figure 3.1: Comparing MAGMA and MISTA with FISTA. Relative CPU times of MAGMA and MISTA compared to FISTA on face recognition experiments with dictionaries of varying number of images (440, 944 and 8824). For each dictionary 20 random input images b were chosen, which indicate the numbers on horizontal axis. Additionally, the results with 4 random starting points for each experiment are reflected as horizontal error bars.

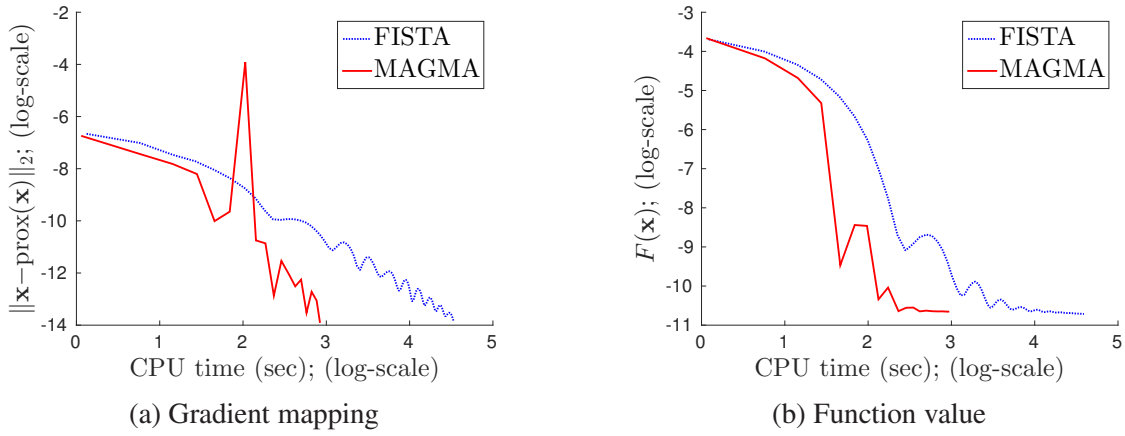


Figure 3.2: **MAGMA and FISTA convergences over time.** Comparisons of gradient mappings and function values of MAGMA, MISTA and FISTA over time.

time in seconds (in log scale). Note that the coarse correction steps of MAGMA take place at the steep dropping points giving large decrease, but then at gradient correction steps first large increase and then relatively constant behaviour is recorded for both plots. Overall, MAGMA reduces both objective value and the norm of the gradient mapping significantly faster.

3.3.3 Stochastic Algorithms for Robust Face Recognition

In this section we test the state of the art stochastic gradient and randomised coordinate descent methods against FISTA and MAGMA for solving robust face recognition problems.

In our numerical experiments we found that CD and SG algorithms are not suitable for robust face recognition problems. There are two reasons for this. Firstly, the data matrix contains highly correlated data. The correlation is due to the fact that we are dealing with a fine-grained object recognition problem. That is, the samples of the dictionary are all facial images of different people and the problem is to identify the particular individual. The second reason is the need to extend the standard ℓ_1 regularized least squared model so that it can handle gross errors, such as occlusions. It can be achieved by using the dense error correction formulation [WM10] in (2.8) (also referred to as the “bucket” model in [WM10]).

To demonstrate our argument we implemented APCG and SVRG and compared them with FISTA [BT09b]. We run all three algorithms for a fixed time and compare the achieved function values.

For APCG we tried three different block size strategies, namely 1, 10 and 50. While 10 and 50 usually performed similarly, 1 was exceptionally slow, so we report the best achieved results for all experiments. For SVRG one has to choose a fixed step size as well as a number of inner iterations. We tuned both parameters to achieve the best possible results for each particular problem.

We performed the experiments on 5 databases: the 3 reported in the previous subsection (with 440, 944 and 8824 images of 200×200 dimension) and two “databases” with data generated uniformly at random with dimensions $40,000 \times 100$ and $40,000 \times 8,000$. The later experiments are an extreme case where full gradient methods (FISTA) are outperformed by both APCG and SVRG when on standard ℓ_1 -regularized least squared problems, while MAGMA is not applicable. The results are given in Figure 3.3. As we can see from Figures 3.3a and 3.3b all three methods can achieve low objective values very quickly for standard ℓ_1 -regularized least squared problems. However, after adding an identity matrix to the right of database (dense error correction model) the performance of all algorithms changes: they all become significantly slower due to larger problem dimensions (Figures 3.3c and 3.3d). Most noticeably SVRG stagnates after first one-two iterations. For the smaller problem (Figure 3.3c) FISTA is the fastest to converge to a minimizer, but for the larger problem (Figure 3.3d) APCG performs best. The picture is quite similar when looking at optimality conditions instead of objective function values in Figures 3.3e to 3.3h: all three algorithms, especially SVRG are slower on the dense error correction model and for the largest problem APCG performs best.

This demonstrates that for ℓ_1 -regularized least squares problems partial gradient methods with a random dictionary can often be faster than accelerated full gradient methods. However, for problems with highly correlated dictionaries and noisy data, such as the robust face recognition problem, the picture is quite different.

Having established that APCG and SVRG are superior than FISTA for random data with no gross errors, we turn our attention to the problem at hand, i.e. the robust face recognition problem. First we run FISTA, APCG, SVRG and MAGMA on problems with no noise and thus the bucket model (2.8) is not used. We run all algorithms for 100 seconds on a problem with $n = 440$ images of $m = 200 \times 200 = 40,000$ dimension in the database. As the results in Figure 3.4 show all algorithms

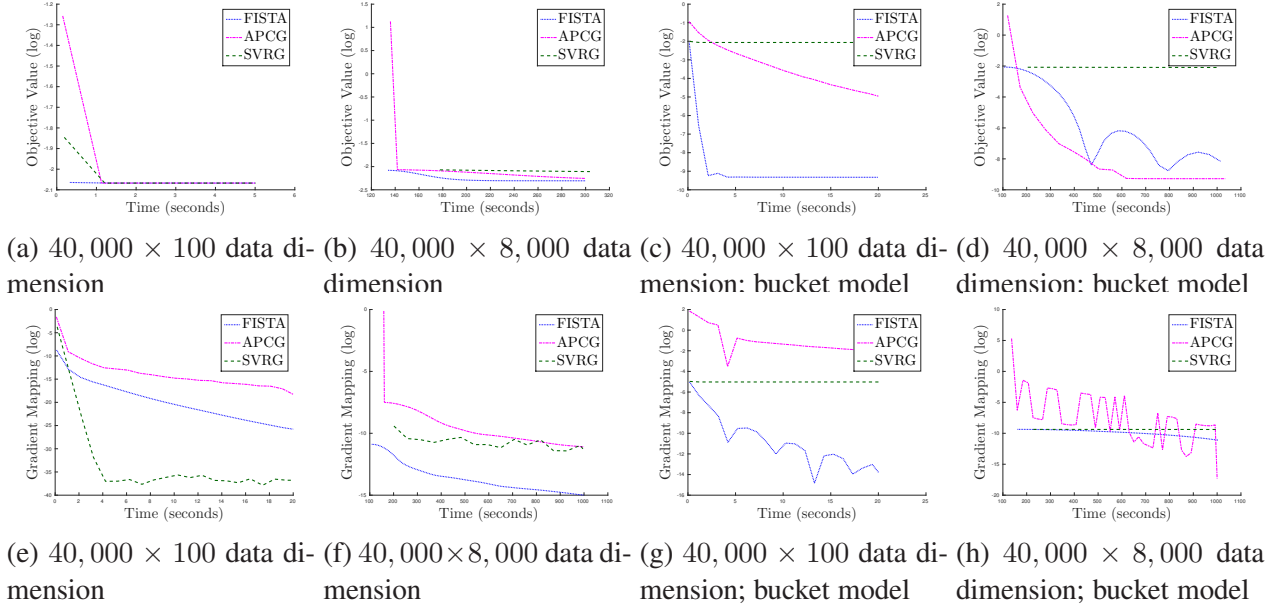


Figure 3.3: Comparing FISTA, APCG and SVRG after running them for a fixed amount of time with ((c), (d), (g) and (h)) and without ((a), (b), (e) and (f)) the bucket model on randomly generated data. The first row contains sequences of the objective values and the bottom row contains optimality criteria, i.e gradient mappings.

achieve very similar solutions of good quality with APCG resulting in obviously less sparse solution (Figure 3.4g).

Then we run all the tested algorithms on the same problem, but this time we simulated occlusion by adding a black square on the incoming image. Thus, we need to solve the dense error correction optimisation problem (2.8). The results are shown in Figure 3.5. The top row contains the original occluded image and reconstructed images by each algorithm and the middle row contains reconstructed noises as determined by each corresponding algorithm. FISTA (Figure 3.5b) and MAGMA (Figure 3.5e) both correctly reconstruct the underlying image while putting noise (illumination changes and gross corruptions) into the error part (Figures 3.5f and 3.5i). APCG (Figure 3.5c), on the other hand, reconstructs the true image together with the noise into the error part (Figure 3.5g), as if the sought person does not have images in the database. SVRG seems to find the correct image in the database, however it fails to separate noise from the true image (Figures 3.5d and 3.5h). We also report the reconstruction vectors $\mathbf{x}^*$ as return by each algorithm in the bottom row. FISTA (Figure 3.5j) and MAGMA (3.5m) both are fairly sparse with a clear spark indicating to the correct image in the database. SVRG (Figure 3.5l) also has a minor spark indicating to the correct person in the database, however it is not sparse, since it could not separate the noise. The result from APCG (Figure 3.5k) is

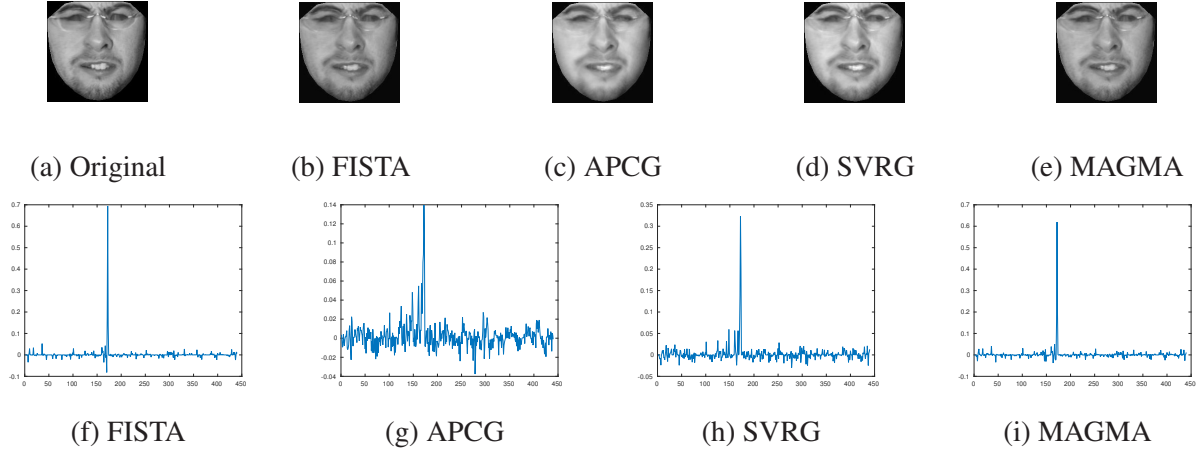


Figure 3.4: Result of FISTA, APCG, SVRG and MAGMA in an image decomposition example without occlusion. Top row ((a) to (e)) contains the original input image and reconstructed images from each algorithm. The bottom row ((f) to (i)) contains the solutions $\mathbf{x}^*$ from each corresponding algorithm.

the worst, since it is not sparse and indicates to multiple different images in the database, thus failing in the face recognition task.

Note that this is not a specifically designed case, in fact, SVRG, APCG and other algorithms that use partial information per iteration might suffer from this problem. Indeed, APCG uses one or a few columns of $\mathbf{A}$ at a time, thus effectively transforming the original large database into many tiny ones, which cannot result in good face recognition. SVRG, on the other hand, uses one row of $\mathbf{A}$ and all of variable $\mathbf{x}$ at a time for all iterations, which allows it to solve the recognition task correctly. However, when the dense error correction optimisation problem in (2.8) is used the minibatch approach results in using only one row of the identity matrix with each corresponding row of $\mathbf{A}$ and one coordinate from variable $\mathbf{e}$ together with $\mathbf{x}$, therefore resulting in poor performance when there is noise present.

To further investigate the convergence properties of the discussed algorithms, we plot the objective function values achieved by APCG, SVRG, FISTA and MAGMA on problems with 440, 944 and 8824 images in the database in Figure 3.6⁷. As we can see from Figures 3.6a and 3.6b APCG is slower than FISTA for smaller problems and slightly outperforms it only for the largest dictionary (Figure 3.6c). SVRG on the other hand quickly drops in function after one-two iterations, but then stagnates. MAGMA is significantly faster than all three algorithms in all experiments.

⁷For all experiments we use the same standard parameters.

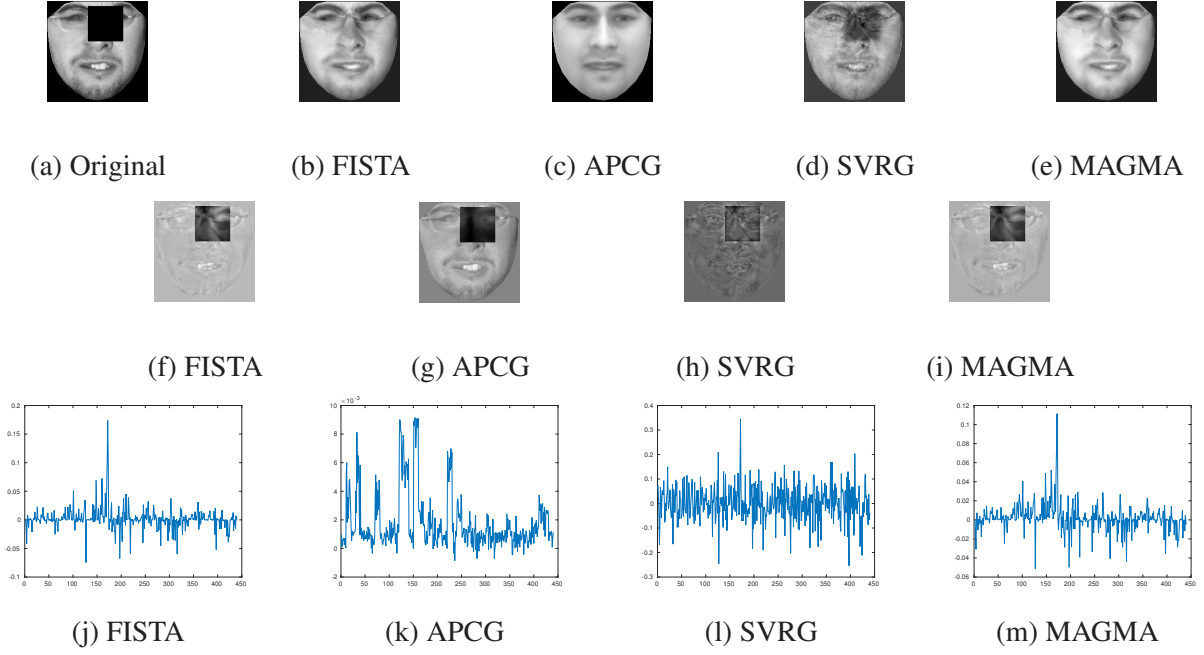


Figure 3.5: Result of FISTA, APCG, SVRG and MAGMA in an image decomposition example with corruption. Top row ((a)-(e)) contains the original occluded image and reconstructed images from each algorithm. The middle row ((f)-(i)) contains the reconstructed noise ($\mathbf{e}^*$ reshaped as an image) as determined by corresponding algorithms. The bottom row ((j)-(m)) contains the solutions $\mathbf{x}^*$ from each corresponding algorithm.

However, looking at the objective function value alone can be deceiving. In order to obtain a full picture of the performance of the algorithms for the face recognition task, one has to look at the optimality conditions and the sparsity of the obtained solution. In order to test it, we run FISTA, APCG, SVRG and MAGMA until the first order condition is satisfied with $\epsilon = 10^{-6}$ error. As the plots in Figure 3.7 indicate APCG and SVRG are slower to achieve low convergence criteria. Furthermore, as Table 3.3 shows they also produce denser solutions with higher ℓ_1 -norm.

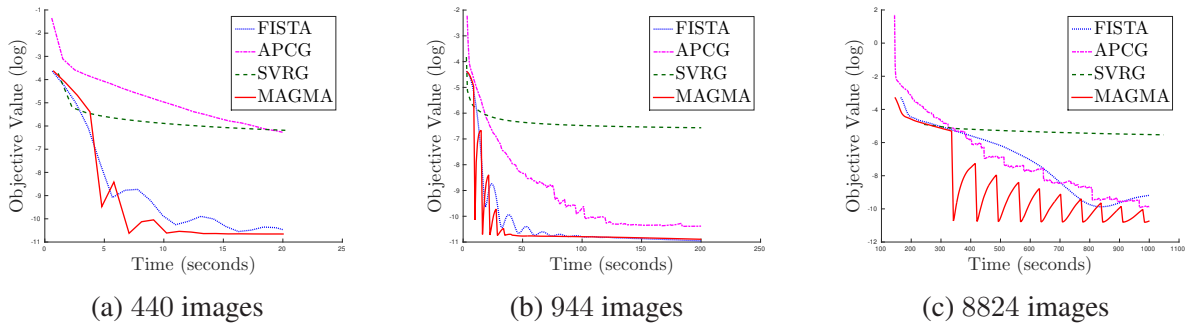


Figure 3.6: Comparing the objective values of FISTA, APCG, SVRG and MAGMA after running them for a fixed amount of time on dictionaries of various sizes.

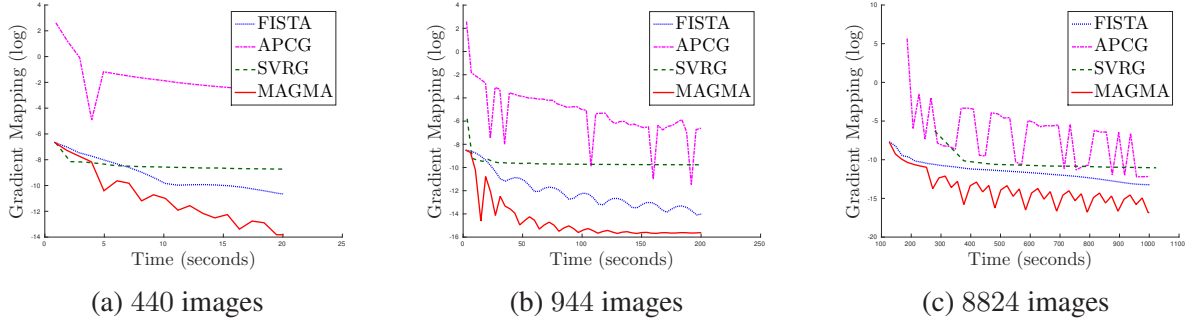


Figure 3.7: Comparing the first order optimality criteria of FISTA, SVRG, APCG and MAGMA after running them for a fixed amount of time on dictionaries of various sizes.

Method	CPU Time (seconds)	Number of iterations	$\ \mathbf{x}^*\ _1$
FISTA	1690	554	3.98
APCG	680	68,363 (block size 10)	6.67
SVRG	$\geq 10,000$	≥ 93 (epoch length 17,648)	5.19
MAGMA	526	144 fine and 120 coarse	3.09

Table 3.3: Running FISTA, APCG, SVRG and MAGMA on the largest dictionary (with 8824 images) until convergence with $\epsilon = 10^{-6}$ accuracy or maximum time of 10,000 CPU seconds returning $\mathbf{x}^*$ as a solution.

3.4 Conclusion and Discussion

In this chapter we presented a novel accelerated multilevel algorithm - MAGMA, for solving convex composite problems. We showed that in theory our algorithm has an optimal convergence rate of $\mathcal{O}(1/\sqrt{\epsilon})$, where ϵ is the accuracy. To the best of our knowledge this is the first multilevel algorithm with optimal convergence rate. Furthermore, we demonstrated on several large-scale face recognition problems that in practice MAGMA can be up to 10 times faster than the state of the art methods.

In terms of theory, we note that even though we prove that MAGMA has the same worst case asymptotic convergence rate as Nesterov's accelerated methods, it would be interesting to see, what conditions (we expect it to be high correlation of the columns of $\mathbf{A}$) imposed on the problem ensure that MAGMA has a better convergence rate or at least a strictly better constant factor. This could also suggest an automatic way for choosing parameters κ and K_d optimally and a closed form solution for the step size s_k .

Chapter 4

Multilevel Robust PCA

4.1 Introduction

In this chapter, motivated by the recent advances in multilevel optimisation algorithms [HP16, HPZ16, JH17, Nas00, SP16, Par17], we propose a simple, yet quite generic and very effective approach for significantly reducing computational costs for many problems that require solving nuclear norm based oracles, including (RPCA) and (CPCP) models. We exploit the fact that many problems arising from computer vision and machine learning applications can be modelled using various degrees of fidelity. For instance, video frames from a fixed camera or facial images taken with varying illuminations are highly correlated, and therefore, their linear combinations maintain the model’s underlying information.

The basic idea is to construct and solve lower dimensional (coarse) models for each optimisation oracle and then lift its solution to the original problem dimension. We show that the multilevel variants always have significantly cheaper per iteration complexity compared to the standard version. In all our experiments we observed at least two times per iteration speed-up and the average per iteration speed was 3.2 times. More importantly, we show that using special restriction and prolongation operators result in algorithms converging to a (approximate) solution of the original problem.

We apply the multilevel approach on three state of the art algorithms - the inexact augmented La-

grange multiplier method (IALM) for the robust PCA problem [LCM10], the Frank-Wolfe Thresholding (FW-T) method for the more general CPCP model [MZWG16] and the alternating projections (AltProj) method for non-convex robust PCA model [NNS⁺14].

For the IALM we are proposing an approximate multilevel algorithm and show that i) it converges to an approximate solution and the approximation error is very small for the type of problems we are concerned with, and ii) in practice it is several times faster than the standard IALM. To the best of our knowledge this is the first provably convergent variant of IALM (and alternating directions methods in general) with approximate updates.

For the FW-T method we prove that its multilevel variant converges in function value with the same worst case iteration complexity, however, each iteration of the multilevel algorithm is much cheaper. We then show on various synthetic and real life problems that for both methods the multilevel variant solves large-scale problems several times faster than its standard counterpart. Moreover, not only the speed up achieved via the multilevel methods allows us to solve large problems that would otherwise take impractically long computational time, but also on some problems the standard method fails to solve the problem at all, while its multilevel variant solves it very efficiently.

We also present a multilevel variant for the non-convex alternating projections method. In this case we do not give a formal convergence proof, but we do show that each multilevel iteration achieves a very good approximate solution. And more importantly, we demonstrate in many synthetic and real problems that the multilevel AltProj algorithm is much faster and can solve problems that the standard one fails.

Before proceeding to the proposed multilevel algorithms, let us restate the two problems of interest: (RPCA) and (CPCP). RPCA (robust principal component analysis) refers to the following constrained minimisation problem:

$$\min_{\mathbf{L}, \mathbf{S}} \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1, \quad \text{subject to } \mathbf{D} = \mathbf{L} + \mathbf{S}, \quad (\text{RPCA})$$

where $\lambda > 0$ is a weighting parameter. And CPCP (compressive principle component pursuit) refers to the following more general minimisation problem:

$$\min_{\mathbf{L}, \mathbf{S}} \frac{1}{2} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{L} + \mathbf{S} - \mathbf{D}]\|_F^2 + \lambda_L \|\mathbf{L}\|_* + \lambda_S \|\mathbf{S}\|_1, \quad (\text{CPCP})$$

where λ_L and λ_S are positive penalising coefficients.

These models have myriads of applications. For example, consider a surveillance video from a fixed camera. Then by stacking each frame of the video as a column vector we can represent the video as a matrix and then decomposing it as a sum of low rank and sparse matrices we will achieve a separation of background (low rank) and moving objects (sparse). Specifically, if we denote the solution of either (RPCA) or (CPCP) by $\{\mathbf{L}^*, \mathbf{S}^*\}$, then $\mathbf{L}^*$ will be the matrix containing the video background and $\mathbf{S}^*$ - the matrix containing only the moving objects.

4.2 The Coarse Model

In this section we will introduce a generic lower dimensional model for the (CPCP) and (RPCA) problems. It uses the so-called restriction operator $\mathbf{R} \in \mathbb{R}^{n \times n_H}$, for some $n_H \leq n$, where n_H is the dimension of the coarse model. Throughout this chapter we will make the following assumption about the restriction operator $\mathbf{R}$.

Assumption 2. *The restriction operator $\mathbf{R}$ has linearly independent columns. Therefore, $\mathbf{R}$ has a left inverse $\mathbf{R}^\dagger \in \mathbb{R}^{n_H \times n}$ so that $\mathbf{R}^\dagger \mathbf{R} = \mathbf{I}_{n_H}$ (in general, $\mathbf{R} \mathbf{R}^\dagger \neq \mathbf{I}_n$ and $\mathbf{R}$ may not have a right inverse).*

Assumption 2 is a very generic and natural assumption about the restriction operator and it is satisfied for all restriction operators used in this work. Indeed, there is no practical advantage of having redundant columns in $\mathbf{R}$, and we can always remove the redundant columns thus creating lower dimensional coarse models. In fact, the full weighted operator 2.30 and its powers used in this work are full rank operators.

Throughout this chapter we will denote $\mathbf{L}^* \in \mathbb{R}^{m \times n}$ as the low rank component of either the (RPCA) or (CPCP) problems. We also make the following natural assumptions about the coarse model.

Assumption 3.

$$\text{rank}(\mathbf{L}^*) \leq n_H \leq \frac{m+1}{2}. \quad (4.1)$$

The first inequality of (4.1) holds whenever $n < m$, which is the case in all practical problems we consider in this chapter. The second inequality, on the other hand, has to be explicitly enforced using an approximate guess on the rank of $\mathbf{L}^*$, which is well known for most applications. For example, for the video background extraction problem $\text{rank}(\mathbf{L}^*) \approx 1$ and for the facial shadow removal problem it is ≈ 9 . In all experiments we use this a priori knowledge to set the number of levels so that (4.1) is satisfied.

Assumption 4. *The low-rank component $\mathbf{L}^*$ can be represented as $\mathbf{L}^* = \mathbf{L}_H^* \mathbf{R}^\top$ for some coarse $\mathbf{L}_H^*$.*

Let us consider a simple demonstrative example. Let $\mathbf{L}^*$ be a simple white rank one background extracted from a video, and thus it can be written as a matrix of ones. Also assume $\mathbf{R}$ is the normalised full weighted operator (2.30). Then setting $\mathbf{L}_H^*$ as a matrix of ones multiplied with an appropriate normalising constant we get

$$\mathbf{L}_H^* \mathbf{R}^\top = \begin{bmatrix} 1 & 1 & 1 \\ & \dots & \\ 1 & 1 & 1 \end{bmatrix} \cdot \frac{1}{2} \begin{bmatrix} 2 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ & & \dots & & & \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \mathbf{L}^*. \quad (4.2)$$

Assumption 4 is the main assumption of this chapter. It essentially says that the solution can be represented with varying degrees of fidelity, which is what we observe in many applications, including those studied here. It is important to notice, that while we assume the existence of such $\mathbf{L}_H^*$, we do not need to know its value. Nowhere in our multilevel algorithms we use the value of $\mathbf{L}_H^*$.

In multilevel literature the standard choice for restriction operator is the full weighted operator (2.30) and we will also use it in our experiments. Often in practice we use more than 2 levels of coarse models. Specifically, we use a restriction operator $\mathbf{R} = \mathbf{R}_n \mathbf{R}_{\frac{n}{2}} \dots \mathbf{R}_{n_H} \in \mathbb{R}^{n \times n_H}$, where $\mathbf{R}_k \in \mathbb{R}^{k \times \frac{k}{2}}$ is the full weighted operator of appropriate dimensions. For all experiments we use up to the deepest possible levels, so that $n_H > \max\{\text{rank}(\mathbf{L}^*), r\}$, where r is the number of singular values required by the overlying algorithm. Clearly, this $\mathbf{R}$ has linearly independent columns and thus is full rank.

4.3 Multilevel IALM

In this section we present a computationally efficient multilevel variant of the Inexact ALM algorithm presented in Section 2.3.2. First, we make an additional assumption for this subsection:

Assumption 5. $\mathbf{R}$ is normalised so that $\|\mathbf{R}\|_2 \leq \|\mathbf{R}\|_* \leq 1$.

Before we proceed, we shall present an important inequality about singular values that we will use in the proofs of this chapter. It's given below and the proof can be found for instance in [WX97].

Theorem 4.1. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$ are matrices and $m \geq n$. Then for any $k \in [1, n]$

$$\sum_{i=1}^k \sigma_i(\mathbf{A})\sigma_{n-i+1}(\mathbf{B}) \leq \sum_{i=1}^k \sigma_i(\mathbf{AB}) \leq \sum_{i=1}^k \sigma_i(\mathbf{A})\sigma_i(\mathbf{B}). \quad (4.3)$$

Now we show that the restriction operator approximately preserves the nuclear norm.

Theorem 4.2. For any $\mathbf{L}_H \in \mathbb{R}^{m \times n_H}$ and $\mathbf{R} \in \mathbb{R}^{n \times n_H}$ with $n_H \leq n \leq m$, the following inequalities hold:

1.

$$\|\mathbf{L}_H \mathbf{R}^\top\|_* \geq \|\mathbf{L}_H\|_* - \epsilon, \quad (4.4)$$

with $\epsilon = \sum_{k=1}^{r_H} \sigma_k(\mathbf{L}_H)(1 - \sigma_{n_H-k+1}(\mathbf{R}))$, where $r_H = \text{rank}(\mathbf{L}_H)$; and

2. if $\|\mathbf{R}\|_2 \leq 1$, then also

$$\|\mathbf{L}_H\|_* \geq \|\mathbf{L}_H \mathbf{R}^\top\|_*. \quad (4.5)$$

Proof. Since $n \leq n_H$, then using (4.3) we have

$$\begin{aligned} \|\mathbf{L}_H \mathbf{R}^\top\|_* &\geq \sum_{k=1}^{n_H} \sigma_k(\mathbf{L}_H \mathbf{R}^\top) \\ &\geq \sum_{k=1}^{n_H} \sigma_k(\mathbf{L}_H) \sigma_{n_H-k+1}(\mathbf{R}^\top) \\ &= \|\mathbf{L}_H\|_* - \sum_{k=1}^{r_H} \sigma_k(\mathbf{L}_H)(1 - \sigma_{n_H-k+1}(\mathbf{R})). \end{aligned} \quad (4.6)$$

The second part can be shown similarly. From Assumption 2 we have

$$\begin{aligned}
\|\mathbf{L}_H\|_* &= \|\mathbf{L}_H \mathbf{R}^\top \mathbf{R}^{\dagger\top}\|_* \\
&\geq \sum_{k=1}^{n_H} \sigma_k(\mathbf{L}_H \mathbf{R}^\top \mathbf{R}^{\dagger\top}) \\
&\geq \sum_{k=1}^{n_H} \sigma_k(\mathbf{L}_H \mathbf{R}^\top) \sigma_{n_H-k+1}(\mathbf{R}^{\dagger\top}) \\
&= \|\mathbf{L}_H \mathbf{R}^\top\|_* - \sum_{k=1}^{r_H} \sigma_k(\mathbf{L})(1 - \sigma_{n_H-k+1}(\mathbf{R}^\dagger)) \\
&= \|\mathbf{L}_H \mathbf{R}^\top\|_* - \sum_{k=1}^{r_H} \sigma_k(\mathbf{L})(1 - \sigma_{n_H-k+1}^{-1}(\mathbf{R})).
\end{aligned} \tag{4.7}$$

Finally, if $\sigma_k(\mathbf{R}) \leq 1$ for all $k = n_H - r_H + 1, \dots, n_H$, then $\|\mathbf{L}_H \mathbf{R}^\top\|_* \leq \|\mathbf{L}_H\|_*$.

□

As it can be seen from Theorem 4.2, in general having $\epsilon = 0$ requires an orthogonal $\mathbf{R}$, which may not be sparse, thus making each iteration of the algorithm computationally expensive. However, if $\mathbf{L}^*$ is low rank with quickly decreasing singular values and also $\sigma_i(\mathbf{R})$ drop slowly, then ϵ will be small. This is indeed the case for many computer vision applications such as video background extraction and facial shadow removal discussed in this chapter, where $\mathbf{L}^*$ is not only low rank, but also has quadratically decreasing singular values. Moreover, it is easy to check that the singular values of the full weighted restriction operator (2.30) satisfy $\sigma_1(\mathbf{R})/\sigma_{n_H}(\mathbf{R}) \leq 2$.

Now we proceed to introduce our multilevel inexact augmented Lagrangian method of multipliers in Algorithm 8. It is derived from Algorithm 4 by replacing the singular value thresholding of steps 2 – 5 by a multilevel singular value thresholding. Here we first create a coarse model for $\mathbf{M}^{(k)}$ (step 3), then perform singular value thresholding on the lower dimensional coarse matrix (steps 4 – 5) and prolong it to the fine dimension using $\mathbf{R}^\top$ (step 6).

Now we proceed to study the convergence of Algorithm 8. First, we define the sequence $\hat{\mathbf{Y}}^{(k)}$, which

Algorithm 8 Multilevel Inexact ALM (ML-IALM)

Input: $\mathbf{D}, \mathbf{S}^{(0)}, \mathbf{Y}^{(0)} \in \mathbb{R}^{m \times n}; \mu_0 > 0$
1: **for** $k \leftarrow 1$ to ... **do**
2: // Solve $\mathbf{L}_H^{(k+1)} = \arg \min_{\mathbf{L}} \mathcal{L}_H(\mathbf{L}, \mathbf{S}^{(k)}, \mathbf{Y}^{(k)}, \mu_k)$
3: $\mathbf{M}_H^{(k)} \leftarrow (\mathbf{D} - \mathbf{S}^{(k)} + \mu_k^{-1} \mathbf{Y}^{(k)}) \mathbf{R}$
4: $(\mathbf{U}_H, \boldsymbol{\Sigma}_H, \mathbf{V}_H) \leftarrow \text{SVD}(\mathbf{M}_H^{(k)})$
5: $\mathbf{L}_H^{(k+1)} \leftarrow \mathbf{U}_H \mathcal{S}_{\mu^{-1}}[\boldsymbol{\Sigma}_H] \mathbf{V}_H^\top$
6: $\mathbf{L}^{(k+1)} \leftarrow \mathbf{L}_H^{(k+1)} \mathbf{R}^\top$
7: // Continue as in Algorithm 4
8: $\mathbf{S}^{(k+1)} \leftarrow \mathcal{S}_{\lambda \mu_k^{-1}}[\mathbf{D} - \mathbf{L}^{(k+1)} + \mu_k^{-1} \mathbf{Y}^{(k)}]$
9: $\mathbf{Y}^{(k+1)} \leftarrow \mathbf{Y}^{(k)} + \mu_k (\mathbf{D} - \mathbf{L}^{(k+1)} - \mathbf{S}^{(k+1)})$
10: Update $\mu_k \leftarrow \mu_{k+1}$
11: **end for**
12: **return** $(\mathbf{L}^{(k+1)}, \mathbf{S}^{(k+1)})$

we will use throughout this section. Let

$$\widehat{\mathbf{Y}}^{(k+1)} = \mathbf{Y}^{(k)} + \mu_k [\mathbf{D} - \mathbf{S}^{(k)} - \mathbf{L}^{(k+1)}]. \quad (4.8)$$

Now we can proceed to prove multilevel versions of some lemmas of [LCM10], which we will use to show the convergence of Algorithm 8. We begin with Lemma 1 from [LCM10] and show that Multilevel IALM also produces bounded sequences.

Lemma 9. *Let $\widehat{\mathbf{Y}}^{(k+1)}$ be defined as in (4.8) and $\mathbf{Y}^{(k)}$ be defined as in Algorithm 8. Then the sequence $\{\mathbf{Y}^{(k)}\}$ is bounded and the following hold:*

1.

$$\widehat{\mathbf{Y}}^{(k+1)} \mathbf{R} + \mu_k \mathbf{L}_H^{(k+1)} (\mathbf{R}^\top \mathbf{R} - \mathbf{I}) \in \partial \|\mathbf{L}_H^{(k+1)}\|_*, \quad (4.9)$$

for any $\mathbf{L}_H^{(k+1)} \in \mathbb{R}^{m \times n_H}$.

2.

$$\mathbf{Y}^{(k+1)} \in \partial \|\mathbf{S}^{(k+1)}\|_*. \quad (4.10)$$

Proof. To show (4.9) we use the optimality condition of $\mathcal{L}_H$ and the construction $\mathbf{L}^{(k+1)} = \mathbf{L}_H^{(k+1)} \mathbf{R}^\top$.

We have

$$\begin{aligned}
\mathbf{0} &\in \partial_{\mathbf{L}_H} \mathcal{L}(\mathbf{L}_H^{(k+1)}, \mathbf{S}^{(k)}, \mathbf{Y}^{(k)}, \mu_k) \\
&= \partial \|\mathbf{L}_H^{(k+1)}\|_* - \mathbf{Y}^{(k)} \mathbf{R} - \mu_k [(\mathbf{D} - \mathbf{S}^{(k)}) \mathbf{R} - \mathbf{L}_H^{(k+1)}] \\
&= \partial \|\mathbf{L}_H^{(k+1)}\|_* - \widehat{\mathbf{Y}}^{(k+1)} \mathbf{R} - \mu_k (\mathbf{L}_H^{(k+1)} \mathbf{R}^\top \mathbf{R} - \mathbf{L}_H^{(k+1)}).
\end{aligned} \tag{4.11}$$

Similarly, using the optimality condition for updating $\mathbf{S}^{(k+1)}$ we can show (4.10):

$$\mathbf{Y}^{(k+1)} \in \partial \|\mathbf{S}^{(k+1)}\|_*. \tag{4.12}$$

□

Lemma 10 ([LCM10], Lemma 2).

$$\begin{aligned}
&\|\mathbf{S}^{(k+1)} - \mathbf{S}^*\|_F^2 + \mu_k^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^*\|_F^2 \\
&= \|\mathbf{S}^{(k)} - \mathbf{S}^*\|_F^2 + \mu_k^{-2} \|\mathbf{Y}^{(k)} - \mathbf{Y}^*\|_F^2 - \|\mathbf{S}^{(k+1)} - \mathbf{S}^{(k)}\|_F^2 - \mu_k^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}\|_F^2 \\
&\quad - 2\mu_k^{-1} (\langle \mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}, \mathbf{S}^{(k+1)} - \mathbf{S}^{(k)} \rangle + \langle \mathbf{L}^{(k+1)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k+1)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k+1)} - \mathbf{S}^*, \mathbf{Y}^{(k+1)} - \mathbf{Y}^* \rangle).
\end{aligned} \tag{4.13}$$

Proof. Since the proof of Lemma 2 in [LCM10] relies only on the optimality of $\mathbf{L}^*$ and $\mathbf{S}^*$ and the update formula for $\mathbf{Y}^{(k+1)}$, but not on the update rule for $\mathbf{L}^{(k+1)}$ (the only multilevel update part of ML-IALM), then its proof can be exactly repeated for Algorithm 8. □

Lemma 11. *If $\|\mathbf{R}\|_2 \leq 1$, then under Assumption 4*

$$\langle \mathbf{L}^{(k+1)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k+1)} - \mathbf{Y}^* \rangle \geq -\epsilon - \mu_k \Delta_{k+1}, \tag{4.14}$$

for ϵ defined as in Theorem 4.2 and $\Delta_{k+1} = \langle \mathbf{L}_H^{(k+1)} (\mathbf{R}^\top \mathbf{R} - \mathbf{I}), \mathbf{L}_H^{(k+1)} - \mathbf{L}_H^* \rangle$.

Proof. Since $\mathbf{Y}^* \in \partial \|\mathbf{L}^*\|_*$, we have

$$\|\mathbf{L}_H^{(k+1)} \mathbf{R}^\top\|_* - \|\mathbf{L}_H^* \mathbf{R}^\top\|_* \geq \langle \mathbf{Y}^*, \mathbf{L}^{(k+1)} - \mathbf{L}^* \rangle. \tag{4.15}$$

Then applying Theorem 4.2 with $\mathbf{L}_H = \mathbf{L}_H^*$ we derive:

$$\|\mathbf{L}_H^{(k+1)} \mathbf{R}^\top\|_* - \|\mathbf{L}_H^*\|_* \geq \langle \mathbf{Y}^*, \mathbf{L}^{(k+1)} - \mathbf{L}^* \rangle - \epsilon. \quad (4.16)$$

On the other hand, from (4.9) we have

$$\|\mathbf{L}_H^*\|_* - \|\mathbf{L}_H^{(k+1)}\|_* \geq \langle \widehat{\mathbf{Y}}^{(k+1)} \mathbf{R} + \mu_k \mathbf{L}_H^{(k+1)} (\mathbf{R}^\top \mathbf{R} - \mathbf{I}), \mathbf{L}_H^* - \mathbf{L}_H^{(k+1)} \rangle. \quad (4.17)$$

Then adding (4.16) and (4.17) and using Assumption 4 we get

$$\|\mathbf{L}_H^{(k+1)} \mathbf{R}^\top\|_* - \|\mathbf{L}_H^{(k+1)}\|_* \geq \langle \mathbf{Y}^* - \widehat{\mathbf{Y}}^{(k+1)}, \mathbf{L}^{(k+1)} - \mathbf{L}^* \rangle - \epsilon - \mu_k \langle \mathbf{L}_H^{(k+1)} (\mathbf{R}^\top \mathbf{R} - \mathbf{I}), \mathbf{L}_H^{(k+1)} - \mathbf{L}_H^* \rangle. \quad (4.18)$$

We finish the proof applying the construction $\mathbf{L}^{(k+1)} = \mathbf{L}_H^{(k+1)} \mathbf{R}^\top$ and denoting $\Delta_{k+1} := \langle \mathbf{L}_H^{(k+1)} (\mathbf{R}^\top \mathbf{R} - \mathbf{I}), \mathbf{L}_H^{(k+1)} - \mathbf{L}_H^* \rangle$.

□

Lemma 12. *Let Δ_k be defined as in Lemma 11 and define*

$$c_k := \mu_k^{-1} (\langle \mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}, \mathbf{S}^{(k+1)} - \mathbf{S}^{(k)} \rangle + \langle \mathbf{L}^{(k+1)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k+1)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k+1)} - \mathbf{S}^*, \mathbf{Y}^{(k+1)} - \mathbf{Y}^* \rangle). \quad (4.19)$$

Then if μ_k is non-decreasing, then

- $c_k \geq -\epsilon \mu_k^{-1} - \Delta_{k+1}$, and
- $\sum_{k=1}^{k=+\infty} c_k < +\infty$.

Proof. Let $(\mathbf{L}^*, \mathbf{S}^*, \mathbf{Y}^*)$ be a saddle point of the Lagrangian of (RPCA). So we have

$$\mathbf{Y}^* \in \partial \|\mathbf{L}^*\|_*, \quad \mathbf{Y}^* \in \partial \|\lambda \mathbf{S}^*\|_1. \quad (4.20)$$

Then from Lemma 3 of [LCM10] and $\mathbf{Y}^{(k+1)} \in \partial \|\lambda \mathbf{S}^{(k+1)}\|_1$ we have

$$\begin{aligned} \langle \mathbf{S}^{(k+1)} - \mathbf{S}^*, \mathbf{Y}^{(k+1)} - \mathbf{Y}^* \rangle &\geq 0, \\ \langle \mathbf{S}^{(k+1)} - \mathbf{S}^{(k)}, \mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)} \rangle &\geq 0. \end{aligned} \quad (4.21)$$

Adding (4.14) and the two inequalities of (4.21) we get

$$\langle \mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}, \mathbf{S}^{(k+1)} - \mathbf{S}^{(k)} \rangle + \langle \mathbf{L}^{(k+1)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k+1)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k+1)} - \mathbf{S}^*, \mathbf{Y}^{(k+1)} - \mathbf{Y}^* \rangle \geq -\epsilon - \mu_k \Delta_{k+1}. \quad (4.22)$$

Showing that $c_k \geq -\epsilon \mu_k^{-1} - \Delta_{k+1}$.

Show part 2, we apply Lemma 10 and use $\mu_{k+1} \geq \mu_k$ to get

$$\|\mathbf{S}^{(k+1)} - \mathbf{S}^*\|_F^2 + \mu_{k+1}^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^*\|_F^2 \leq \|\mathbf{S}^{(k)} - \mathbf{S}^*\|_F^2 + \mu_k^{-2} \|\mathbf{Y}^{(k)} - \mathbf{Y}^*\|_F^2 + 2\epsilon \mu_k^{-1} + 2\Delta_{k+1}. \quad (4.23)$$

Therefore from Lemma 10 we conclude that

$$\begin{aligned} &2\mu_k^{-1} (\langle \mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}, \mathbf{S}^{(k+1)} - \mathbf{S}^{(k)} \rangle + \langle \mathbf{L}^{(k+1)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k+1)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k+1)} - \mathbf{S}^*, \mathbf{Y}^{(k+1)} - \mathbf{Y}^* \rangle) \\ &\leq (\|\mathbf{S}^{(k)} - \mathbf{S}^*\|_F^2 + \mu_k^{-2} \|\mathbf{Y}^{(k)} - \mathbf{Y}^*\|_F^2) - (\|\mathbf{S}^{(k+1)} - \mathbf{S}^*\|_F^2 + \mu_{k+1}^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^*\|_F^2). \end{aligned}$$

□

Theorem 4.3 (Convergence of Multilevel IALM). *For Algorithm 8, if $\{\mu_k\}$ is non-decreasing, $\sum_{k=1}^{+\infty} \mu_k^{-1} = +\infty$ and $\sum_{k=1}^{+\infty} \mu_k^{-2} < +\infty$, then $(\mathbf{L}^{(k)}, \mathbf{S}^{(k)})$ asymptotically converges to an approximate solution of (RPCA).*

Proof. Similar to the proof of Lemma 12 we have that

$$\begin{aligned}
& \mu_k^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}\|_F^2 - 2\epsilon\mu_k^{-1} - 2[\Delta_{k+1}]_+ \\
& \leq \mu_k^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}\|_F^2 - 2\epsilon\mu_k^{-1} - 2\Delta_{k+1} \\
& \leq (\|\mathbf{S}^{(k)} - \mathbf{S}^*\|_F^2 + \mu_k^{-2} \|\mathbf{Y}^{(k)} - \mathbf{Y}^*\|_F^2) - (\|\mathbf{S}^{(k+1)} - \mathbf{S}^*\|_F^2 + \mu_{k+1}^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^*\|_F^2) - 2\epsilon\mu_k^{-1} - 2\Delta_{k+1} \\
& \quad - 2\mu_k^{-1} (\langle \mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}, \mathbf{S}^{(k+1)} - \mathbf{S}^{(k)} \rangle + \langle \mathbf{L}^{(k+1)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k+1)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k+1)} - \mathbf{S}^*, \mathbf{Y}^{(k+1)} - \mathbf{Y}^* \rangle) \\
& \leq (\|\mathbf{S}^{(k)} - \mathbf{S}^*\|_F^2 + \mu_k^{-2} \|\mathbf{Y}^{(k)} - \mathbf{Y}^*\|_F^2) - (\|\mathbf{S}^{(k+1)} - \mathbf{S}^*\|_F^2 + \mu_{k+1}^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^*\|_F^2).
\end{aligned} \tag{4.24}$$

Therefore,

$$\sum_{k=1}^{+\infty} (\mu_k^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}\|_F^2 - 2\epsilon\mu_k^{-1} - 2[\Delta_{k+1}]_+) < +\infty. \tag{4.25}$$

Thus, $\mu_k^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^{(k)}\|_F^2 - 2\epsilon\mu_k^{-1} - 2[\Delta_{k+1}]_+ \rightarrow 0$, and since $2\epsilon\mu_k^{-1} \rightarrow 0$ we see that

$$\|\mathbf{D} - \mathbf{L}^{(k)} - \mathbf{S}^{(k)}\|_F = \mu_k^{-1} \|\mathbf{Y}^{(k)} - \mathbf{Y}^{(k-1)}\|_F \rightarrow (2[\Delta_{k+1}]_+)^{1/2}, \tag{4.26}$$

Moreover, since we showed in Lemma 9 that $\{\mathbf{Y}^{(k+1)}\}$ is a bounded sequence, it follows that so is $[\Delta_k]_+$. Therefore, denoting $\max\{\Delta_k\} := \delta$ we show that any accumulation point of $(\mathbf{L}^{(k)}, \mathbf{S}^{(k)})$ is a δ -feasible solution.

On the other hand, denote the optimal objective value of problem (RPCA) by f^* . As $\widehat{\mathbf{Y}}^{(k)} \mathbf{R} \in \partial\|\mathbf{L}_H^{(k)}\|_*$ and $\mathbf{Y}^{(k)} \in \partial(\lambda\|\mathbf{S}^{(k)}\|_1)$, under Assumption 5 we have

$$\begin{aligned}
& \|\mathbf{L}^{(k)}\|_* + \lambda\|\mathbf{S}^{(k)}\|_1 \\
& \leq \|\mathbf{L}_H^{(k)}\|_* + \lambda\|\mathbf{S}^{(k)}\|_1 \\
& \leq \|\mathbf{L}_H^*\|_* + \lambda\|\mathbf{S}^*\|_1 - \langle \widehat{\mathbf{Y}}^{(k)} \mathbf{R}, \mathbf{L}_H^* - \mathbf{L}_H^{(k)} \rangle - \langle \mathbf{Y}^{(k)}, \mathbf{S}^* - \mathbf{S}^{(k)} \rangle \\
& \leq \|\mathbf{L}^*\|_* + \epsilon + \lambda\|\mathbf{S}^*\|_1 - \langle \widehat{\mathbf{Y}}^{(k)}, \mathbf{L}^* - \mathbf{L}^{(k)} \rangle - \langle \mathbf{Y}^{(k)}, \mathbf{S}^* - \mathbf{S}^{(k)} \rangle \\
& = f^* + \langle \mathbf{Y}^* - \widehat{\mathbf{Y}}^{(k)}, \mathbf{L}^* - \mathbf{L}^{(k)} \rangle + \langle \mathbf{Y}^* - \mathbf{Y}^{(k)}, \mathbf{S}^* - \mathbf{S}^{(k)} \rangle - \langle \mathbf{Y}^*, \mathbf{L}^* - \mathbf{L}^{(k)} + \mathbf{S}^* - \mathbf{S}^{(k)} \rangle + \epsilon \\
& = f^* + \langle \mathbf{Y}^* - \widehat{\mathbf{Y}}^{(k)}, \mathbf{L}^* - \mathbf{L}^{(k)} \rangle + \langle \mathbf{Y}^* - \mathbf{Y}^{(k)}, \mathbf{S}^* - \mathbf{S}^{(k)} \rangle - \langle \mathbf{Y}^*, \mathbf{D} - \mathbf{L}^{(k)} - \mathbf{S}^{(k)} \rangle + \epsilon.
\end{aligned} \tag{4.27}$$

Similarly to Lemma 12 we notice that from Lemma 10

$$\begin{aligned} & \mu_k^{-1}(\langle \mathbf{L}^{(k)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k)} - \mathbf{S}^*, \mathbf{Y}^{(k)} - \mathbf{Y}^* \rangle) \\ & \leq (\|\mathbf{S}^{(k)} - \mathbf{S}^*\|_F^2 + \mu_k^{-2} \|\mathbf{Y}^{(k)} - \mathbf{Y}^*\|_F^2) - (\|\mathbf{S}^{(k+1)} - \mathbf{S}^*\|_F^2 + \mu_{k+1}^{-2} \|\mathbf{Y}^{(k+1)} - \mathbf{Y}^*\|_F^2). \end{aligned} \quad (4.28)$$

And therefore,

$$\sum_{k=1}^{k=+\infty} \mu_k^{-1}(\langle \mathbf{L}^{(k)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k)} - \mathbf{S}^*, \mathbf{Y}^{(k)} - \mathbf{Y}^* \rangle) < +\infty. \quad (4.29)$$

As $\sum_{k=1}^{+\infty} \mu_k^{-1} = +\infty$, there must exist a subsequence $(\mathbf{L}^{(k_j)}, \mathbf{S}^{(k_j)})$ such that

$$\langle \mathbf{L}^{(k_j)} - \mathbf{L}^*, \widehat{\mathbf{Y}}^{(k_j)} - \mathbf{Y}^* \rangle + \langle \mathbf{S}^{(k_j)} - \mathbf{S}^*, \mathbf{Y}^{(k_j)} - \mathbf{Y}^* \rangle \rightarrow 0. \quad (4.30)$$

Then since $\|\mathbf{D} - \mathbf{L}^{(k)} - \mathbf{S}^{(k)}\|_F \leq \delta$, we have that

$$\lim_{j \rightarrow +\infty} \|\mathbf{L}^{(k_j)}\|_* + \lambda \|\mathbf{S}^{(k_j)}\|_1 \leq f^* + \epsilon + \delta. \quad (4.31)$$

So $(\mathbf{L}^{(k_j)}, \mathbf{S}^{(k_j)})$ approaches to an $(\epsilon + \delta)$ -approximate solution of problem (RPCA).

□

Notice that Theorem 4.3 gives a similar convergence results as Theorem 2.3, meaning that one should expect a similar number of iterations for IALM and ML-IALM methods. This is indeed the case, as observed from empirical studies. However, since ML-IALM performs SVDs on much smaller dimensional matrices, its each iteration is significantly cheaper.

Of course, as opposed to the original IALM algorithm, here we only showed an approximate convergence. However, as several numerical experiments will demonstrate, the approximation error is practically negligible.

4.4 Multilevel Frank-Wolfe

In this section we use operator notation for the linear restriction operator, i.e. $\mathcal{R} : \mathcal{H} \rightarrow \mathcal{H}_H$ is a linear operator from the original space $\mathcal{H}$ to the *coarse space* $\mathcal{H}_H$. $\mathcal{H}$ and $\mathcal{H}_H$ are both Hilbert spaces endowed with inner products. Naturally, $\mathcal{H}_H$ has lower dimension. In the next subsection we will see what $\mathcal{H}$, $\mathcal{H}_H$ and $\mathcal{R}$ are for the CPCP model.

First we create a coarse model for the gradient by applying the restriction operator $\mathcal{R}$, then solve the linear optimisation oracle over the coarse gradient, then lift the solution back to the original dimension applying the transpose of the restriction operator. For the algorithm we use a convex set $\mathcal{D}_H \subseteq \mathcal{H}_H$ such that for every $\mathbf{x}_H \in \mathcal{D}_H$ it holds that $\mathcal{R}^\top(\mathbf{x}_H) \in \mathcal{D}$. The method is given in Algorithm 9.

Algorithm 9 Multilevel Frank-Wolfe (ML-FW)

Input: $\mathbf{x}_H^{(0)} \in \mathcal{D}_H$
1: $\mathbf{x}^{(0)} \leftarrow \mathcal{R}^\top(\mathbf{x}_H^{(0)})$
2: **for** $k \leftarrow 0, 1$, to $\dots$ **do**
3: $\mathbf{v}_H^{(k)} \in \arg \min_{\mathbf{v} \in \mathcal{D}_H} \langle \mathbf{v}, \mathcal{R}(\nabla f(\mathbf{x}^{(k)})) \rangle$
4: $\mathbf{v}^{(k)} \leftarrow \mathcal{R}^\top(\mathbf{v}_H^{(k)})$
5: $\gamma \leftarrow \frac{2}{k+2}$
6: Set $\mathbf{x}^{(k+1)} \in \mathcal{D}$ so that $f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)} + \gamma(\mathbf{v}^{(k)} - \mathbf{x}^{(k)}))$
7: **end for**
8: **return** $\mathbf{x}^{(k+1)}$

Using techniques similar to the original proof [MZWG16], it can be shown that the ML-FW method converges to a point obtained from the coarse level at a $\mathcal{O}(1/k)$ rate in function values.

Theorem 4.4. *For any $\mathbf{x}_H^* \in \mathcal{D}_H$ and for $\{\mathbf{x}^{(k)}\}$ generated by Algorithm 9, we have for any $\mathbf{x}_H^* \in \mathcal{H}_H$ and $k = 0, 1, 2, \dots$*

$$f(\mathbf{x}^{(k)}) - f(\mathcal{R}^\top(\mathbf{x}_H^*)) \leq \frac{2LD^2}{k+2}. \quad (4.32)$$

Proof. For $k = 0, 1, 2, \dots$ we have

$$\begin{aligned}
f(\mathbf{x}^{(k+1)}) &\leq f(\mathbf{x}^{(k)} + \gamma(\mathbf{v}^{(k)} - \mathbf{x}^{(k)})) \\
&\leq f(\mathbf{x}^{(k)}) + \gamma \langle \nabla f(\mathbf{x}^{(k)}), \mathbf{v}^{(k)} - \mathbf{x}^{(k)} \rangle + \frac{L\gamma^2}{2} \|\mathbf{v}^{(k)} - \mathbf{x}^{(k)}\|^2 \\
&\leq f(\mathbf{x}^{(k)}) + \gamma \langle \nabla f(\mathbf{x}^{(k)}), \mathcal{R}^\top(\mathbf{v}_H^{(k)}) - \mathcal{R}^\top(\mathbf{x}_H^{(k)}) \rangle + \gamma \langle \nabla f(\mathbf{x}^{(k)}), \mathcal{R}^\top(\mathbf{x}_H^{(k)}) - \mathbf{x}^{(k)} \rangle + \frac{\gamma^2 LD^2}{2} \\
&= f(\mathbf{x}^{(k)}) + \gamma \langle \mathcal{R}(\nabla f(\mathbf{x}^{(k)})), \mathbf{v}_H^{(k)} - \mathbf{x}_H^{(k)} \rangle + \gamma \langle \nabla f(\mathbf{x}^{(k)}), \mathcal{R}^\top(\mathbf{x}_H^{(k)}) - \mathbf{x}^{(k)} \rangle + \frac{\gamma^2 LD^2}{2} \\
&\leq f(\mathbf{x}^{(k)}) + \gamma \langle \mathcal{R}(\nabla f(\mathbf{x}^{(k)})), \mathbf{x}_H^* - \mathbf{x}_H^{(k)} \rangle + \gamma \langle \nabla f(\mathbf{x}^{(k)}), \mathcal{R}^\top(\mathbf{x}_H^{(k)}) - \mathbf{x}^{(k)} \rangle + \frac{\gamma^2 LD^2}{2} \\
&= f(\mathbf{x}^{(k)}) + \gamma \langle \nabla f(\mathbf{x}^{(k)}), \mathcal{R}^\top(\mathbf{x}_H^*) - \mathbf{x}^{(k)} \rangle + \frac{\gamma^2 LD^2}{2} \\
&\leq f(\mathbf{x}^{(k)}) + \gamma(f(\mathcal{R}^\top(\mathbf{x}_H^*)) - f(\mathbf{x}^{(k)})) + \frac{\gamma^2 LD^2}{2}.
\end{aligned} \tag{4.33}$$

Here for the first line we used the updating rule in Algorithm 9; for the second line we used the Lipschitz continuity of f ; for the third line - the definitions of $\mathbf{v}_H^{(k)}$ and D , and we added and subtracted $\mathcal{R}^\top(\mathbf{x}_H^{(k)})$; for the fourth line - the property of inner product; for the fifth line - the optimality of $\mathbf{v}_H^{(k)}$; for the sixth line - the property of inner product and definition $\mathbf{x}^{(k)}$; and for the last line - the convexity of f .

Now rearranging the terms we get

$$f(\mathbf{x}^{(k+1)}) - f(\mathcal{R}^\top(\mathbf{x}_H^*)) \leq (1 - \gamma)(f(\mathbf{x}^{(k)}) - f(\mathcal{R}^\top(\mathbf{x}_H^*))) + \frac{\gamma^2 LD^2}{2}. \tag{4.34}$$

Therefore, by mathematical induction, it can be verified that

$$f(\mathbf{x}^{(k)}) - f(\mathcal{R}^\top(\mathbf{x}_H^*)) \leq \frac{2LD^2}{k+2} \quad \text{for } k = 1, 2, 3, \dots \tag{4.35}$$

□

Theorem 4.4 tells us, that if the minimiser $\mathbf{x}^*$ can be accurately represented in terms of a course variable, then ML-FWT is a good and efficient method for that particular problem. As we will see in the next subsection, this is indeed the case for the CPCP model.

4.4.1 Multilevel Frank-Wolfe Thresholding for CPCP

In this subsection we will modify the Multilevel Frank-Wolfe method similarly to the Frank-Wolfe Thresholding method introducing the Multilevel Frank-Wolfe Thresholding method and apply it for the CPCP problem (CPCP). In this case as well, we will apply the multilevel update only on nuclear ball projections.

We begin with defining the fine and coarse spaces, and the restriction operator for the CPCP problem. In this setting our variables becomes $\mathbf{x} = (\mathbf{L}, \mathbf{S}, t_L, t_S)$ and the space is $\mathcal{H} = \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \times \mathbb{R} \times \mathbb{R}$.

Since we are applying the multilevel steps only for updating $\mathbf{L}^{(k)}$, we will use the following restriction operator:

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_x & & & \\ & \mathbf{I}_{n \times n} & & \\ & & 1 & \\ & & & 1 \end{pmatrix}, \quad (4.36)$$

so that

$$\begin{aligned} \mathcal{R}(\mathbf{L}, \mathbf{S}, \lambda_L, \lambda_S) &= (\mathbf{L}\mathbf{R}_x, \mathbf{S}, \lambda_L, \lambda_S) \\ \mathcal{R}^\top(\nabla_{\mathbf{L}}f, \nabla_{\mathbf{S}}f, \nabla_{\lambda_L}f, \nabla_{\lambda_S}f) &= (\nabla_{\mathbf{L}}f\mathbf{R}_x, \nabla_{\mathbf{S}}f, \nabla_{\lambda_L}f, \nabla_{\lambda_S}f), \end{aligned} \quad (4.37)$$

and thus $\mathcal{R}(\nabla f(\mathbf{x}))$ only affects $\nabla_{\mathbf{L}}f(\mathbf{L}, \mathbf{S}, t_L, t_S)$ and correspondingly, $\mathcal{R}^\top(\mathbf{v}_H)$ only affects $\nabla f_{\mathbf{L}}$. Here $\mathbf{R}_x = \mathbf{R}_n^d$ with $\mathbf{R}_n$ being the restriction operator and $d \geq 2$ the number of levels used. Therefore, the coarse space becomes $\mathcal{H}_H = \mathbb{R}^{m \times n_H} \times \mathbb{R}^{m \times n} \times \mathbb{R} \times \mathbb{R}$. Now we can define the coarse feasibility set $\mathcal{D}_H$ for the CPCP problem as follows:

$$\|\mathbf{M}_{L,H}\|_* \leq 1/\|\mathcal{R}\|_*, \quad (4.38)$$

so that for each $k = 0, 1, \dots$

$$\|\mathbf{M}_L^{(k)}\|_* = \|\mathcal{R}^\top(\mathbf{M}_{L,H}^{(k)})\|_* \leq \|\mathcal{R}\|_* \|\mathbf{M}_{L,H}^{(k)}\|_* \leq 1, \quad (4.39)$$

is a feasible point of the fine problem, where $\mathbf{M}_{L,H}$ is the coarse variable. We call the new algorithm Multilevel Frank-Wolfe Thresholding (ML-FWT) and give it in Algorithm (10).

Algorithm 10 Multilevel Frank-Wolfe Thresholding (ML-FWT)

Input: $\mathbf{D} \in \mathbb{R}^{m \times n}$; $\lambda_L, \lambda_S > 0$

1: Initialize as in Algorithm 5

2: **for** $k \leftarrow 1$ to ... **do**

3: $\mathbf{M}_{L,H}^{(k)} \in \arg \min_{\|\mathbf{M}_{L,H}\|_* \leq 1/\|\mathcal{R}\|_*} \langle \mathcal{R}(\mathcal{P}_{\mathcal{Q}}[\mathbf{L}^{(k)} + \mathbf{S}^{(k)} - \mathbf{D}]), \mathbf{M}_{L,H} \rangle$

4:

5:

6: $\mathbf{M}_L^{(k)} \leftarrow \mathcal{R}^\top(\mathbf{M}_{L,H}^{(k)})$

7: // Continue as in Algorithm 5

8: **end for**

9: **return** $(\mathbf{L}^{(k+1)}, \mathbf{S}^{(k+1)})$

Thus the multilevel update step for $\mathbf{M}_L^{(k)}$ requires calculating only the largest singular value with the corresponding singular vectors for much lower dimensional matrices. The next theorem gives convergence guarantees for the ML-FWT method.

Theorem 4.5. *Let $f(\mathbf{L}, \mathbf{S}, t_L, t_S)$ be defined as in (2.25) and $\text{rank}(\mathbf{L}^*) \leq n_H$. Then under Assumptions 2 and 4, for $\mathbf{x}^{(k)}$, $k = 1, 2, \dots$ defined as in Algorithm 10 the following holds*

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq \frac{2LD^2}{k+2}. \quad (4.40)$$

Proof. From assumption 4 we derive

$$\begin{aligned} f(\mathcal{R}^\top(\mathbf{L}_H^*, \mathbf{S}^*, t_L^*, t_S^*)) &= \frac{1}{2} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{D} - \mathbf{L}_H^* \mathbf{R}^\top - \mathbf{S}^*]\|_F^2 + \lambda_L t_L^* + \lambda_S t_S^* \\ &= \frac{1}{2} \|\mathcal{P}_{\mathcal{Q}}[\mathbf{D} - \mathbf{L}^* - \mathbf{S}^*]\|_F^2 + \lambda_L t_L^* + \lambda_S t_S^* \\ &= f(\mathbf{L}^*, \mathbf{S}^*, t_L^*, t_S^*). \end{aligned} \quad (4.41)$$

Therefore, the claim follows from Theorem 4.4.

□

Finally, note that since FWT and ML-FWT have the same convergence rate and multiplying by the sparse matrix $\mathbf{R}$ (and its powers) is much cheaper than computing one singular value (although both

have the same $\mathcal{O}(mn)$ rate of complexity), ML-FWT has a lower overall complexity.

4.5 Multilevel Non-Convex Alternating Projections

To demonstrate the extensibility of the multilevel approach, in this section we are proposing to apply it for the non-convex robust PCA model and accordingly the alternating projection algorithm presented with it in section 2.3.4. In this case we create coarse models for subproblems of finding low rank approximations for intermediate matrices $\mathbf{M} = \mathbf{D} - \mathbf{S}^{(k)}$, solve these subproblems and lift their solutions to the original fine dimension.

Each iteration of the Alternating Projections algorithm requires solving

$$\min_{\mathbf{L} \in \mathbb{R}^{m \times n}} \|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_2 \quad s.t. \quad \text{rank}(\mathbf{L}) \leq l, \quad (4.42)$$

which has a closed form solution given by the hard thresholding operator as follows ¹

$$\hat{\mathbf{L}} = \mathbf{U} \mathcal{H}_l[\boldsymbol{\Sigma}] \mathbf{V}^\top, \quad (4.43)$$

where $\mathcal{H}_l[x]$ is the hard thresholding operator and is defined element-wise for vectors and matrices. Therefore, for this setting we use the hard thresholding operator on the coarse model to construct an approximate solution for problem (4.42) as follows:

$$\mathbf{L}^H = \mathbf{U}_H \mathcal{H}_l[\boldsymbol{\Sigma}_H] \mathbf{V}_H^\top \mathbf{R}^\top, \quad (4.44)$$

where $\mathbf{M}\mathbf{R} = \mathbf{M}_H = \mathbf{U}_H \boldsymbol{\Sigma}_H \mathbf{V}^\top$ is a SVD of the coarse model. Notice that the multilevel operator computes SVDs on much smaller problems than the original algorithm in this case as well. Then in Theorem 4.6 we show that the $\mathbf{L}^H$ defined in (4.44) gives a good approximate solution for problem (4.42).

¹this is the Eckhart-Young-Mirsky theorem [GVL12]

Theorem 4.6. *The multilevel low rank approximation procedure of (4.44) gives a $(\sigma_1 + \sigma_{H,1})$ -approximate solution of the problem (4.42), where $\sigma_{H,1}$ is the largest singular value of $\mathbf{M}_H = \mathbf{M}\mathbf{R}$.*

Proof. The proof follows the proof of Eckhard-Young-Mirsky theorem [GVL12] and is given below.

First note that for any $\mathbf{B} = \mathbf{X}\mathbf{Y}^\top$ ($\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m \times k}$) we can find a vector $\omega = \sum_{i=1}^{k+1} \gamma_i \mathbf{v}_i$ so that $\omega^\top \mathbf{X} = \mathbf{0}$ and $\|\omega\|_2 = \sum_{i=1}^{k+1} \gamma_i^2 = 1$, where $\mathbf{v}_i$ are the columns of $\mathbf{V}$. Then

$$\|\mathbf{M} - \mathbf{B}\|_2^2 \geq \|\omega^\top (\mathbf{M} - \mathbf{B})\|_2^2 = \|\omega^\top \mathbf{M}\|_2^2 = \sum_{i=1}^{k+1} \gamma_i^2 \sigma_i^2 \geq \sigma_{k+1}^2. \quad (4.45)$$

On the other hand

$$\begin{aligned} \|\mathbf{M} - \mathbf{L}^H\|_2 &= \|\mathbf{U}_0 \mathbf{\Sigma}_0 \mathbf{V}_0^\top + \mathbf{U}_1 \mathbf{\Sigma}_1 \mathbf{V}_1^\top - \mathbf{U}_{H,0} \mathbf{\Sigma}_{H,0} \mathbf{V}_{H,0}^\top \mathbf{R}^\top\|_2 \\ &\leq \sigma_1 + \sigma_{k+1} + \sigma_{H,1}, \end{aligned} \quad (4.46)$$

where $\mathbf{U}_0, \mathbf{V}_0, \mathbf{U}_{H,0}$ and $\mathbf{V}_{H,0}$ (respectively, $\mathbf{U}_1, \mathbf{V}_1, \mathbf{U}_{H,1}$ and $\mathbf{V}_{H,1}$) are correspondingly the singular vectors associated with the largest k (respectively, smallest $n - k$ and $n_H - k$) singular values of $\mathbf{M}$ and $\mathbf{M}\mathbf{R}$. From these two inequalities we have that for any $\mathbf{B}$

$$\|\mathbf{M} - \mathbf{L}^H\|_2 - \|\mathbf{M} - \mathbf{B}\|_2 \leq \sigma_1 + \sigma_{H,1}. \quad (4.47)$$

□

Then we can use (4.44) inside Algorithm 6 to efficiently solve the corresponding subproblem. The resulting method is presented in Algorithm 11.

4.6 Experiments

To test the practical efficiency of the proposed methods we compare them with the standard Inexact ALM [LCM10] and Frank-Wolfe Thresholding [NNS⁺14] algorithms on several synthetically gener-

Algorithm 11 Multilevel Alternating Projections (ML-AltProj)

Input: $\mathbf{D} \in \mathbb{R}^{m \times n}$, target rank r

```
1: Initialize  $\mathbf{L}^{H(0)} = \mathbf{0}$  and  $\mathbf{S}^{(0)} = \mathcal{H}_{\zeta_0}(\mathbf{D} - \mathbf{L}^{H(0)})$ 
2: for Stage  $l \leftarrow 1$  to  $r$  do
3:   for Iteration  $k \leftarrow 0$  to  $T$  do
4:     // Approx solve  $\arg \min_{\mathbf{L}: \text{rank}(\mathbf{L}) \leq l} \|\mathbf{D} - \mathbf{L} - \mathbf{S}^{(k)}\|_2$ 
5:      $\mathbf{M}_H^{(k)} \leftarrow (\mathbf{D} - \mathbf{S}^{(k)})\mathbf{R}$ 
6:      $(\mathbf{U}_H, \boldsymbol{\Sigma}_H, \mathbf{V}_H) \leftarrow \text{SVD}(\mathbf{M}_H^{(k)}, l)$ 
7:      $\mathbf{L}_H \leftarrow \mathbf{U}_H \mathcal{H}_l[\boldsymbol{\Sigma}_H] \mathbf{V}_H^\top$ 
8:      $\mathbf{L}^{H(k+1)} \leftarrow \mathbf{L}_H \mathbf{R}^\top$ 
9:     // Continue as in Algorithm 6
10:    Update threshold  $\zeta$  as in [NNS+14]
11:     $\mathbf{S}^{(k+1)} \leftarrow \mathcal{H}_\zeta[\mathbf{D} - \mathbf{L}^{H(k+1)}]$ 
12:    if  $\sigma_{l+1}(\mathbf{L}^{H(k+1)}) < \epsilon$  then
13:      return  $(\mathbf{L}^{H(T)}, \mathbf{S}^{(T)})$ 
14:    else
15:       $\mathbf{S}^{(0)} \leftarrow \mathbf{S}^{(T)}$ 
16:    end if
17:  end for
18: end for
19: return  $(\mathbf{L}^{H(T)}, \mathbf{S}^{(T)})$ 
```

ated problems, as well as real life video background extraction and facial shadow removal problems. For the standard Inexact ALM and Frank-Wolfe Thresholding algorithms we used the provided Matlab code. Then for each multilevel variant we replaced the standard singular value thresholding parts of respective algorithms with corresponding multilevel singular value thresholding code, keeping the rest of the algorithms unchanged. Particularly, we used the same optimality criteria, so that the comparisons are fair. All methods were tested in Matlab R2015a on a standard desktop machine with Intel Core i7 processor and 32GB RAM.

4.6.1 Synthetic Data

First we test the multilevel algorithms on synthetically generated data matrix $\mathbf{D} = \hat{\mathbf{L}} + \hat{\mathbf{S}} \in \mathbb{R}^{m \times n}$, where $\hat{\mathbf{L}}$ has a fixed low rank r and $\hat{\mathbf{S}}$ is η -sparse (i.e. has at most $\eta \cdot mn$ non-zero entries). We generate the synthetic data so that the singular values of the low rank component follow $1/k^2$, where k indicates the k -th largest singular value.

problem			IALM			ML-IALM		
dimensions	rank	sec	f^*	$\text{error}(\mathbf{L}^*)$	$\text{error}(\mathbf{S}^*)$	f^*	$\text{error}(\mathbf{L}^*)$	$\text{error}(\mathbf{S}^*)$
5000×100	2	5	19	7	0.1	10	1	0.02
5000×100	5	5	18	6	0.1	7.5	1	0.02
5000×1000	2	10	64	42	0.8	7	1	0.01
5000×1000	5	10	64	43	0.8	8	1	0.01

Table 4.1: Achieved objective function values and relative errors from ground truth after running IALM and ML-IALM on RPCA problems with synthetic data for a fixed time.

We run two sets of experiments. The first one compares the results achieved after running IALM and ML-IALM on RPCA problems for a fixed time. We run two pairs of experiments: with smaller and larger data, each with lower and higher rank of the low-rank component. The experiments are described in Table 4.1, with each row corresponding to one experimental setting. The first three columns describe the particular setting and the number of seconds dedicated to solve the problem. Then each triplet of columns gives the results achieved by IALM and ML-IALM algorithms correspondingly. The reported results are f^* - achieved objective value, $\text{error}(\mathbf{S}^*)$ and $\text{error}(\mathbf{L}^*)$ - achieved relative errors from corresponding ground truths. It is evident that ML-IALM accurately solves all four problems, while both objective values and relative errors from IALM are several times larger than those of ML-IALM. This effectively means that ML-IALM can solve large problems in reasonable time that may require impractically long times for IALM.

Then we synthetically generate similar data, but this time with partial observations. This setting is modelled as a CPCP problem and is then solved using FWT and ML-FWT methods. The experimental settings are given in Table 4.2. Here as well we have two pairs of problems: larger and smaller with larger and smaller ranks of the low-rank component. Here we run both problems until 10^{-3} tolerance as suggested in [MZWG16]. Here both algorithms achieve relatively small objective values and relative errors from the ground truth, with ML-FWT being slightly better, however ML-FWT takes significantly less time to do so. In fact, it is more than twice faster for the smaller rank settings.

4.6.2 Video Background Extraction

Now we test the algorithms on real surveillance videos. Assume we are given a surveillance video from a fixed camera and the task is to separate the constant background from moving objects. This

problem		FWT				ML-FWT			
dimensions	rank	sec	f^*	$\text{err}(\mathbf{L}^*)$	$\text{err}(\mathbf{S}^*)$	sec	f^*	$\text{err}(\mathbf{L}^*)$	$\text{err}(\mathbf{S}^*)$
10000×500	2	14.5	$8 \cdot 10^{-4}$	1.3	88	9	$2 \cdot 10^{-4}$	1.1	89.4
10000×5000	2	209	0.02	1.3	125	91	$7 \cdot 10^{-4}$	1	125
10000×500	5	14.4	$8 \cdot 10^{-4}$	1.3	89	12.5	$2 \cdot 10^{-4}$	1.1	89
10000×5000	5	203	0.002	1.3	125	90	$7 \cdot 10^{-4}$	1.1	125

Table 4.2: CPU times (in seconds), achieved objective function values and relative errors from ground truth after running FWT and ML-FWT on CPCP problems with synthetic data until 10^{-3} convergence error.

problem can be modelled as a RPCA problem [BSJ⁺16]. We first stack each frame of the video as a column vector creating a data matrix $\mathbf{D}$. Then, since the fixed background remains (approximately) constant in each frame and the moving objects take a relatively small portion of each frame, they can respectively represent the low rank and sparse components of the RPCA decomposition. We tested all algorithms on several surveillance videos described below.

- **highway**: $48 \times 64 \times 400$; run 5 seconds
- **copy machine**: $48 \times 72 \times 3400$; run 50 seconds ²
- **walk**: $240 \times 320 \times 794$; run 30 seconds [VCWL12]
- **gates**: $240 \times 320 \times 1895$; run 200 seconds [VCWL12]

First we test the IALM and ML-IALM methods. Here we run both methods for a fixed amount of time until a reasonably small error from ground truth has been achieved. The running times for each problem are indicated above. we then compare the results, which are reported in Figure 4.1. Each row represents a tested video. The first column contains sample frames from each corresponding video, then each of the following column triplets contains corresponding low rank and sparse components as returned from IALM and ML-IALM algorithms. Below each frame we also report the corresponding achieved rank and the feasibility gap (FG) i.e. $\|\mathbf{D} - \mathbf{L}^* - \mathbf{S}^*\|_F / \|\mathbf{D}\|_F$.

As the results indicate, both algorithms produce similar results for all videos, except the larger **copy-machine** and **gates** examples, for which ML-IALM produces significantly clearer separation of background than IALM.

²<http://wordpress-jodoin.dmi.usherb.ca/dataset2012/>

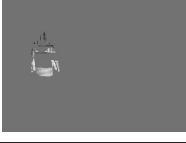
Original	Low Rank		Sparse	
	IALM	ML-IALM	IALM	ML-IALM
				
highway	rank =5	rank=3	FG = 0.0176	FG=0.01
				
copy machine	rank = 7	rank=4	FG = 0.0364	FG=0.0031
				
walk	rank = 2	rank=1	FG = 0.02	FG=0.0231
				
gates	rank = 3	rank=3	FG = 0.05	FG=0.04

Figure 4.1: Examples from solving video background extraction problems via IALM and ML-IALM methods. Both IALM and ML-IALM run for a fixed CPU seconds. Each row corresponds respectively to **highway** ($48 \times 64 \times 400$), **copy machine** ($48 \times 72 \times 3,400$), **walk** ($240 \times 320 \times 794$) and **gates** ($240 \times 320 \times 1,895$) videos from top to bottom. With each frame we also report the respective rank of the low rank component and the feasibility gap (FG): $\|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_F / \|\mathbf{D}\|_F$.

In order to further investigate the convergence properties of the ML-IALM algorithm compared to the standard IALM, we measure the relative error of the current iterates compared to the ground truth $(\mathbf{L}_0, \mathbf{S}_0)$ and FGs during the iterations of both algorithms through the same time interval. We report those relative errors against CPU time (seconds) and iteration numbers in Figure 4.2. The plots suggest that ML-IALM performs only slightly faster than IALM on the smaller **highway** example, however, as expected it is significantly faster on the larger **copy machine** problem. As we could anticipate from the theory, at each iteration ML-IALM achieves a very good approximation as measured by the reconstruction error, and since its iterations are significantly cheaper, it performs more iterations during the same time interval than IALM.

In all experiments we used 4 levels of coarse models for all four problems. In this case as well, the multilevel variant largely outperforms the original algorithm. In fact, the larger the original problem, the bigger relative speed up can be achieved using the multilevel approach, since for larger n we can use deeper levels.

Next we test the performance of our ML-FWT algorithm against the standard FWT. In this case we will also add 75% random noise to the original video. Here we run both algorithms until convergence with 10^{-3} accuracy as suggested in [MZWG16]. Consequently, here we will report the running times and the achieved results (objective value, rank of the low-rank component and sparsity of the sparse component) of each algorithm in Table 4.3. As the numbers indicate both algorithms achieve very similar objective values and sparsity of the sparse component. However that ranks of the low rank components is better for ML-FWT. In fact for the largest problems FWT returns values with very large ranks and thus fails to solve the problem, while ML-FWT performs equally well on all problems. Furthermore, ML-FWT is much faster, especially on larger problems.

4.6.3 Shadow removal from facial images

Here we have a set of facial images from one individual under various illuminations and the task is to remove shadow/light noises from images. This problem can also be modelled as RPCA by stacking the facial images as column vectors and then putting them together to form the data matrix. Then since

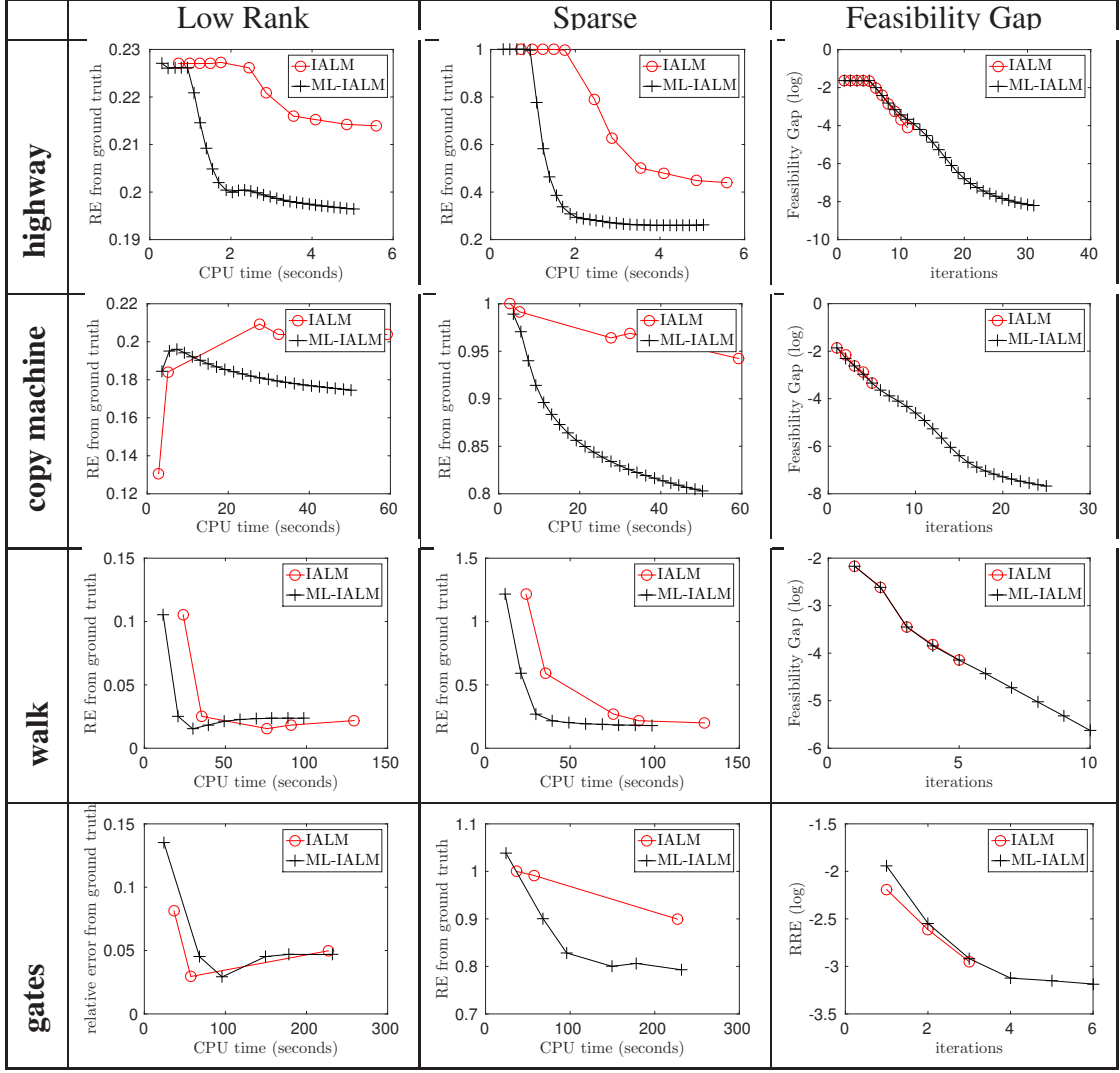


Figure 4.2: Comparing the relative errors during IALM and ML-IALM iterations. The first two columns give relative errors (RE) compared to the ground truth ($\mathbf{L}_0, \mathbf{S}_0$), and the third column gives feasibility gaps (FG) during iterations. Each row corresponds respectively to **highway** ($48 \times 64 \times 400$), **copy machine** ($48 \times 72 \times 3,400$), **walk** ($240 \times 320 \times 794$) and **gates** ($240 \times 320 \times 1,895$) videos from top to bottom.

problem (dimensions)	FWT				ML-FWT			
	sec	f^*	$\text{rank}(\mathbf{L}^*)$	$\text{sp}(\mathbf{S}^*)$	sec	f^*	$\text{rank}(\mathbf{L}^*)$	$\text{sp}(\mathbf{S}^*)$
highway (3072×400)	14	0.001	39	0.22	10	0.001	36	0.22
hall (25344×200)	50	0.001	38	0.47	37	0.001	12	0.47
copym. (3456×3400)	373	0.001	104	0.11	160	0.001	26	0.17
mall (81920×300)	337	0.001	32	0.42	195	0.001	9	0.43
lobby (20480×1000)	487	0.001	111	0.05	385	0.001	31	0.06

Table 4.3: CPU time (in seconds), achieved objective value, rank and sparsity after solving the resulting PCP problem for noisy video background extraction up to tolerance 10^{-3} using the standard Frank-Wolfe Thresholding (FWT) and its multilevel variant ML-FWT.

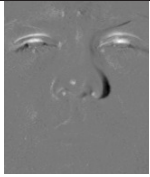
Original	Low Rank		Sparse	
	IALM	ML-IALM	IALM	ML-IALM
				
Yale B01	rank = 5	10	FG = 0.24	0.05
				
Yale B02	rank = 5	10	FG = 0.24	0.05
				
Yale B10	rank = 3	10	FG = 0.24	0.05

Figure 4.3: Examples from solving facial shadow removal problems via IALM and ML-IALM algorithms on cropped **Yale B** database ($96 \times 84 \times 2414$). We run both IALM and ML-IALM for fixed five seconds. With each image we also report the respective rank of the low rank component and the feasibility gap (FG): $\|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_F / \|\mathbf{D}\|_F$.

aligned frontal facial images span a low dimensional subspace, we can represent the clear images as the low-rank component of the data matrix and the shadow will become the sparse component.

We used images of individuals from the Yale B facial extended database [GBK01]. It contains (96×84) dimensional facial images of 39 subjects taken under various poses and illuminations each, with total 2,414 images. For this setting as well we ran the IALM and ML-IALM algorithms for a fixed 5 second and compare the returned results, which are reported in Figure 4.3. Here as well each row represents a particular problem setting (individual). The first column contains sample frames from each corresponding facial database, then each of the following four columns contains correspondingly low rank and sparse components as returned from IALM, and ML-IALM algorithms. With each image we also report the corresponding achieved rank of the low rank component and the feasibility gap.

A brief examination of the Table 4.3 reveals that ML-IALM produces much better separation for each subject, with a more accurate rank 10 of the low-rank component and a five times smaller feasibility gap.

For the shadow removal problem as well, we test the Frank-Wolfe methods on the noisy data with 75% contaminated entries. Both FWT and ML-FWT run until convergence with 10^{-3} tolerance and record CPU times (seconds) and the achieved rank of the low-rank component and sparsity of the sparse component. The results of all 35 subjects are reported in Table 4.4. In all experiments we used up to 4 levels of coarse models. Here in all experiments both methods achieved similar objective values and sparsity values, as expected. However ML-FWT is not only twice faster, but it also achieves a much better rank of the low-rank component. In fact, FWT fails to solve the problem, since the low-rank component is essentially full rank.

4.6.4 Multilevel Non-convex Alternating Projection

We devote this last section to experimentally studying the multilevel alternating projections method (ML-AltProj). We test it on the same problems as ML-IALM and ML-FWT, i.e. synthetic data, video background extraction and shadow removal. We begin the presentation of the results from the synthetically generated data sets as described in Table 4.1.

Table 4.5 shows that the standard alternating projections method fails to solve the problems. The ML-AltProj method, on the other hand, not only converges several times faster, but also solves the problems to a high accuracy. This is a very interesting observation, showing that the AltProj is largely dependant on the model assumptions and may fail in many common scenarios.

To have a better understanding of how quickly either of the algorithm converges, we also measured the distances from ground truth over time at each iteration. The plots are given in Table 4.4. These plots confirm that the AltProj iterations fail to move towards the ground truth, even though it decreased the feasibility gap at each iteration. Interestingly, each iteration of both algorithms achieves a similar decrease in feasibility gap. Thus considering that each iteration of ML-AltProj is much cheaper, ML-AltProj decreases the feasibility gap faster in CPU time.

Next we present the results after running AltProj and ML-AltProj algorithms on video background extraction problems. We use the same videos as for the IALM methods in Section 4.6.2. The experimental results are given in Table 4.5. The first column contains the problem names. Then each next

problem	FWT				ML-FWT			
	sec	f^*	rank($\mathbf{L}^*$)	sp($\mathbf{S}^*$)	sec	f^*	rank($\mathbf{L}^*$)	sp($\mathbf{S}^*$)
yaleB01	83	0.001	65	0.71	41	0.001	8	0.73
yaleB02	71	0.00098	65	0.71	39	0.00098	8	0.74
yaleB03	75	0.00099	65	0.71	40	0.00099	8	0.73
yaleB04	72	0.001	65	0.72	42	0.001	8	0.73
yaleB05	61	0.00096	65	0.7	33	0.00094	8	0.73
yaleB06	77	0.001	65	0.72	48	0.001	8	0.73
yaleB07	78	0.001	65	0.72	42	0.00099	8	0.73
yaleB08	75	0.00099	65	0.71	44	0.00098	8	0.73
yaleB09	84	0.001	65	0.71	48	0.00097	8	0.73
yaleB10	67	0.00099	65	0.72	45	0.001	8	0.73
yaleB11	75	0.00098	60	0.71	41	0.00097	7	0.73
yaleB12	79	0.00099	59	0.71	41	0.00097	7	0.73
yaleB13	60	0.00096	60	0.71	35	0.00097	7	0.74
yaleB15	85	0.001	63	0.71	43	0.00097	8	0.73
yaleB16	77	0.00098	62	0.7	47	0.001	7	0.73
yaleB17	66	0.00098	63	0.71	42	0.00099	8	0.73
yaleB18	85	0.001	63	0.71	45	0.00099	8	0.73
yaleB19	77	0.001	64	0.71	48	0.001	8	0.73
yaleB20	73	0.00099	64	0.7	43	0.001	8	0.73
yaleB21	75	0.00098	64	0.7	43	0.00099	8	0.73
yaleB22	88	0.00098	64	0.69	44	0.00093	8	0.72
yaleB23	63	0.00098	64	0.71	45	0.001	8	0.73
yaleB24	74	0.00099	64	0.71	49	0.001	8	0.73
yaleB25	75	0.00099	64	0.7	48	0.001	8	0.73
yaleB26	77	0.001	64	0.71	45	0.00099	8	0.73
yaleB27	61	0.00097	64	0.69	40	0.00096	8	0.72
yaleB28	61	0.00096	64	0.69	40	0.00097	8	0.73
yaleB29	71	0.00099	64	0.7	44	0.00099	8	0.73
yaleB30	72	0.00098	64	0.71	43	0.00099	8	0.73
yaleB31	78	0.001	64	0.71	48	0.001	8	0.73
yaleB32	60	0.00096	64	0.7	37	0.00097	8	0.72
yaleB33	79	0.00099	64	0.7	44	0.00099	8	0.73
yaleB34	78	0.00099	64	0.7	37	0.00097	8	0.73
yaleB35	75	0.00099	64	0.7	45	0.00099	8	0.73
yaleB36	72	0.00099	64	0.71	42	0.00098	8	0.73

Table 4.4: CPU times (in seconds) after solving shadow removal problems up to a fixed tolerance using the standard Frank-Wolfe Thresholding algorithm and its multilevel variant. For all experiments we used 2 levels for the multilevel algorithm.

problem		AltProj				ML-AltProj					
	dimensions	rank	sec	f^*	error(L^*)	error(S^*)	sec	f^*	error(L^*)	error(S^*)	
1	5000×100	2	6.3	6.5	74	10	46	1	0.85	0.02	
2	5000×100	5	12	10	113	8.5	50	1	1	0.02	
3	5000×1000	2	42	14	105	5	80	1.4	1.1	0.02	
4	5000×1000	5	61	15	144	7	70	2	1	0.03	

Table 4.5: CPU times (in seconds), achieved objective function values and relative errors from ground truth after running AltProj and ML-AltProj on non-convex robust PCA problems with synthetic data until 10^{-7} convergence error.

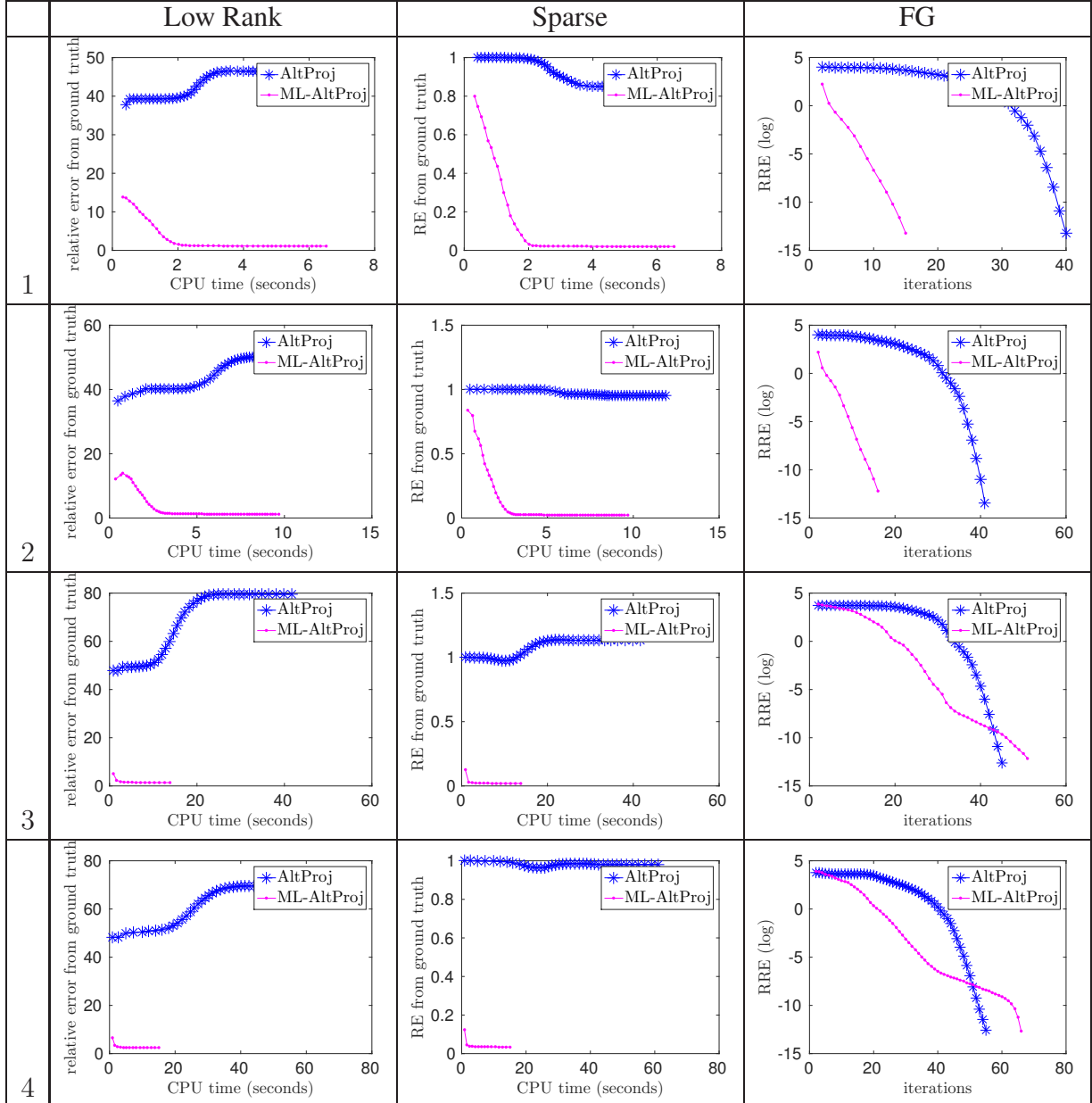


Figure 4.4: Comparing the relative errors during AltProj and ML-AltProj iterations. The first two columns give relative errors compared to the ground truth (L_0, S_0), and the third column gives feasibility gaps (FG) during iterations. Each row corresponds to synthetic data of following dimension and rank as given in Table 4.5.

Problem	CPU time (sec)		Low Rank		Sparse	
	AltProj	ML	AltProj	ML-AltProj	AltProj	ML-AltProj
highway	9.9	7.4	 rank = 1	 rank = 1	 FG = 0.3	 FG = 0.2
copymachine	79.4	21.3	 rank = 1	 rank = 1	 FG = 0.5	 FG = 0.4
walk	580.7	332.7	 rank = 1	 rank = 1	 FG = 1.6	 FG = 1.6
gates	1045	466	 rank = 2	 rank = 2	 FG = 1.6	 FG = 1.5

Figure 4.5: Examples from solving video background extraction problems via non-convex AltProj and ML-AltProj methods. Both AltProj and ML-IALM run until convergence with 10^{-7} error. Each row corresponds respectively to **highway** ($48 \times 64 \times 400$), **copy machine** ($48 \times 72 \times 3,400$), **walk** ($240 \times 320 \times 794$) and **gates** ($240 \times 320 \times 1,895$) videos from top to bottom. With each frame we also report the respective rank of the low rank component and the feasibility gap (FG): $\|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_F / \|\mathbf{D}\|_F$.

pair compares the CPU running times (in seconds), and achieved low-rank and sparse components from each algorithm. As the table shows that both algorithms find visually very similar solutions, however ML-AltProj does it in much less time.

And finally, we test the algorithms for shadow removal problems on the same Yale B database. Here as well, we run both algorithms until convergence with 10^{-7} error. The results are reported in Table 4.6. As it was in all previous examples, the multilevel algorithm converges to a similar solution in less computational time, however in this case the difference is smaller, since these are very small problems and can be solved quickly with very high accuracy. Interestingly, in this case as well, the Alternating Projections method fails for some cases. For the Yale B02 individual the low-rank component contains large amount of noise, while the multilevel version correctly separates the face from strips.

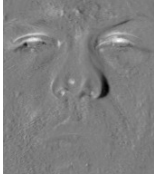
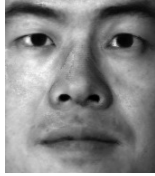
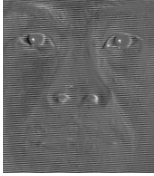
Original	time		Low Rank		Sparse	
	AltProj	ML-AltProj	AltProj	ML-AltProj	AltProj	ML-AltProj
 Yale B01	34	25	 rank = 9	 rank = 9	 $FG = 4 \cdot 10^{-4}$	 $FG = 2 \cdot 10^{-4}$
 Yale B02	36	29	 rank = 9	 rank = 9	 $FG = 4 \cdot 10^{-4}$	 $FG = 2 \cdot 10^{-4}$
 Yale B10	35	26	 rank = 9	 rank = 9	 $FG = 4 \cdot 10^{-4}$	 $FG = 2 \cdot 10^{-4}$

Figure 4.6: Examples from solving facial shadow removal problems via the non-convex alternating projections (AltProj) and multilevel alternating projections (ML-AltProj) algorithms on samples from cropped **Yale B** database ($96 \times 84 \times 65$). We run both methods until convergence with 10^{-7} error. With each experiment we also report the required CPU times in seconds, respective ranks of the low-rank component and the feasibility gaps (FG): $\|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_F / \|\mathbf{D}\|_F$ for each method

4.7 Conclusion

In this chapter we developed a generic multilevel framework for solving robust PCA type problems. We applied the approach on three state of the art optimisation algorithms.

The first algorithm is a multilevel variant of the well-known inexact augmented Lagrange method (or more generally, ADMM), called ML-IALM. We proved that ML-IALM converges to an approximate solution, with approximation error being small for many computer vision problems, including those studied here. To the best of our knowledge this is the first time when an ADMM with approximate steps was proven to converge.

Our second algorithm is a multilevel variant of the well known Frank-Wolfe method modified to be most efficient for CPCP problems. We showed that this multilevel algorithm also converges to the solution of the CPCP problem with the same rate as its standard counterpart, while have much lower per iteration complexity.

The third method is based on the alternating projection algorithm designed for the non-convex formulation of robust PCA. In this case we did not give a formal convergence proof, leaving it for the future research. However we showed that the each iteration of the ML-AltProj method gives a good approximate projection. Moreover, we showed on many numerical examples that the multilevel alternating projections method is not only significantly faster, but also in many situations it can solve problems where its standard counterpart fails.

We tested all three algorithms methods on various synthetic and real life problems. The results clearly show that the multilevel algorithms are not only several times faster (especially on larger problems), but also can often solve problems that their standard counterparts cannot.

Chapter 5

Discussion

5.1 Summary

The aim of this thesis was to develop, implement and test multilevel optimisation algorithms for some of the most important computer vision problems. We started the thesis with a chapter dedicated to the background theory required for the subsequent chapters. Specifically, we had discussions about the convex composite optimisation model, algorithms for solving it and how many statistical machine learning and computer vision problems can be modelled as a convex composite minimisation problem.

The first project of the thesis was aimed to solve large convex composite optimisation problems and specifically facial recognition problems that previously could not be solved due to the lack of efficient algorithms. Based on the observation that facial recognition problems (as many others in computer vision) can be modelled using varying degrees of fidelity we proposed a multilevel algorithm. It uses techniques from several well known optimisation algorithms, namely gradient descent and mirror descent, thus its name - multilevel accelerated gradient mirror descent algorithm (MAGMA). We showed that MAGMA converges to a solution with the best generally possible convergence rate having much cheaper per iteration complexity. Moreover, as we demonstrated on several facial recognition problems, MAGMA is up to a magnitude times faster than the state of the art.

The second chapter was dedicated to developing multilevel optimisation algorithms for the well-

known robust PCA model. In fact, we proposed three state of the art methods, i) multilevel inexact augmented Lagrange method of multipliers, ii) multilevel Frank-Wolfe Thresholding method, and iii) multilevel non-convex alternating projections method. We first showed that the algorithms converge to a (approximate) solution and require much less computational time than their standard counterparts. We also performed comprehensive numerical experiments to demonstrate that the large advantage that multilevel methods have.

5.2 Future Work

Although the two projects of the thesis are self content and complete, there are ways to improve them. First, we used smoothing and line search techniques for MAGMA in Chapter 3, since choosing a step-size is not clear for the multilevel versions of accelerated first order methods. However, smoothing not only makes the method more complicated, but also limits its applicability to problems where either smoothing or line search is expensive. For instance, MAGMA in its current form is not applicable for robust PCA, since it would require several full SVDs for each iteration to determine the best step size. Thus it is extremely interesting to see if it is possible to extend FISTA or APG to use multilevel updates.

Second, we did not proof the convergence of the non-convex multilevel alternating projections method, instead only showing that its each iteration gives a good approximation. This is clearly only a heuristic, though a very good one as we showed on several synthetic and real problems. However, it should be possible to show that ML-AltProj converges to perhaps an approximate solution using similar techniques as in [NNS⁺14].

And finally, it would be very interesting to see multilevel algorithms applied to other vision problems. They have already been applied to deblurring [PLRR14, Par17] and photoacoustic tomography, but there are much more problems that have the potential to greatly benefit much more efficient multilevel methods. More interestingly, we showed in Chapter 4, the same multilevel technique can be applied within very different methods largely improve all of them. So another interesting research direction would be developing more multilevel algorithms. This has partly been done by my colleagues Chin

Pang Ho [Ho16] and Juan Campos [Sal17].

Appendices

Appendix A

List of acronyms

Here we give an index of acronyms used in the thesis.

acronym	full name
AGM	accelerated gradient mirror descent
ALM	augmented Lagrange multipliers
AltProj	alternating projections
APCG	accelerated randomised proximal coordinate gradient method
BP	basis pursuit
BPD	basis pursuit denoising
CD	coordinate descent
CPCP	compressive principle component pursuit
DALM	Dual Proximal Augmented Lagrangian Method
DEC	dense error correction
DFG	distance generating function
ERM	empirical risk minimisation
FG	feasibility gap
FISTA	fast iterative shrinkage thresholding algorithm
FR	face recognition
FW	Frank-Wolfe

FWT	Frank-Wolfe thresholding
IALM	inexact augmented Lagrange method of multipliers
ISTA	iterative shrinkage thresholding algorithm
IT	iterative thresholding
LASSO	least absolute shrinkage and selection operator
MAGMA	multilevel accelerated gradient mirror descent algorithm
ML-AltProj	multilevel alternating projections
ML-FW	multilevel Frank-Wolfe
ML-FWT	multilevel Frank-Wolfe thresholding
ML-IALM	multilevel inexact augmented Lagrange method of multipliers
PDIPA	primal dual interior point method
PCA	principle component analysis
PCG	pre-conditioned conjugate gradient
RPCA	robust principle component analysis
SG	stochastic gradient
SVD	singular value decomposition
SVRG	stochastic variance reduced gradient
SVT	singular value thresholding
TNIPM	truncated Newton interior-point method

Bibliography

- [AK98] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237–260, 1998.
- [AK06] Gilles Aubert and Pierre Kornprobst. *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer Science & Business Media, 2006.
- [AZL16] Zeyuan Allen-Zhu and Yuanzhi Li. Even faster svd decomposition yet without agonizing pain. In *Advances in Neural Information Processing Systems*, pages 974–982, 2016.
- [AZO14] Zeyuan Allen-Zhu and Lorenzo Orecchia. A novel, simple interpretation of nesterov’s accelerated method as a combination of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*, 2014.
- [AZP11] Roland Angst, Christopher Zach, and Marc Pollefeys. The generalized trace-norm and its application to structure-from-motion problems. In *2011 International Conference on Computer Vision*, pages 2502–2509. IEEE, 2011.
- [BCG11] Stephen R Becker, Emmanuel J Candès, and Michael C Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165–218, 2011.
- [BCN16] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

- [BDH99] Dietrich Braess, Maksimillian Dryja, and W Hackbush. A multigrid method for non-conforming FE-discretisations with application to non-matching grids. *Computing*, 63(1):1–25, 1999.
- [Ber99] Dimitri P Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.
- [BJKK11] Peter N Belhumeur, David W Jacobs, David Kriegman, and Neeraj Kumar. Localizing parts of faces using a consensus of exemplars. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 545–552. IEEE, 2011.
- [BM⁺00] William L Briggs, Steve F McCormick, et al. *A multigrid tutorial*. Siam, 2000.
- [BS09] Alfio Borzi and Volker Schulz. Multigrid methods for PDE optimization. *SIAM review*, 51(2):361–395, 2009.
- [BSJ⁺16] Thierry Bouwmans, Andrews Sobral, Sajid Javed, Soon Ki Jung, and El-Hadi Zahzah. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset. *Computer Science Review*, 2016.
- [BT09a] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *Image Processing, IEEE Transactions on*, 18(11):2419–2434, 2009.
- [BT09b] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.
- [BT12] Amir Beck and Marc Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.
- [BTN13] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization*. Georgia Tech, 2013.
- [BV04] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

- [Can06] Emmanuel J Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians: Madrid, August 22-30, 2006: invited lectures*, pages 1433–1452, 2006.
- [CCS10] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
- [CDVLL98] Antonin Chambolle, Ronald A De Vore, Nam-Yong Lee, and Bradley J Lucier. Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. *Image Processing, IEEE Transactions on*, 7(3):319–335, 1998.
- [CLMW11] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.
- [CR09] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.
- [CRT06] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.
- [CW08] Emmanuel J Candès and Michael B Wakin. An introduction to compressive sampling. *IEEE SPM*, 25(2):21–30, 2008.
- [DDDM04] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.
- [DKM06] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on computing*, 36(1):158–183, 2006.

- [Don06] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.
- [DT08] David L Donoho and Yaakov Tsaig. Fast solution of-norm minimization problems when the solution may be sparse. *Information Theory, IEEE Transactions on*, 54(11):4789–4812, 2008.
- [EHN96] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.
- [Ela10] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.
- [EMZ07] Michael Elad, Boaz Matalon, and Michael Zibulevsky. Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. *Applied and Computational Harmonic Analysis*, 23(3):346–367, 2007.
- [FR15] Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.
- [FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.
- [GBK01] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.
- [GMC⁺10] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010.
- [GST08] Serge Gratton, Annick Sartenaer, and Philippe L Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, 19(1):414–444, 2008.
- [GVL12] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

- [Ho16] Chin Pang Ho. *Multilevel Algorithms for the Optimization of Structured Problems*. PhD thesis, Imperial College London, 2016.
- [Hot33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.
- [HP16] Chin Pang Ho and Panos Parpas. Multilevel optimization methods: Convergence and problem structure. 2016.
- [HPZ16] Vahan Hovhannisyan, Panos Parpas, and Stefanos Zafeiriou. Magma: Multilevel accelerated gradient mirror descent algorithm for large-scale convex composite minimization. *SIAM Journal on Imaging Sciences*, 9(4):1829–1857, 2016.
- [HRBLM07] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.
- [HTF02] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: Data mining, inference, and prediction. *Biometrics*, 2002.
- [Hub11] Peter J Huber. *Robust statistics*. Springer, 2011.
- [JH17] Ashkan Javaherian and Sean Holman. A multi-grid iterative method for photoacoustic tomography. *IEEE transactions on medical imaging*, 36(3):696–706, 2017.
- [KKL⁺07] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l_1 -regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606–617, 2007.
- [LBL⁺12] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S Huang. Interactive facial feature localization. In *Computer Vision—ECCV 2012*, pages 679–692. Springer, 2012.
- [LCM10] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

- [LLSG14] Risheng Liu, Zhouchen Lin, Zhixun Su, and Junbin Gao. Linear time principal component pursuit and its extensions using l1 filtering. *Neurocomputing*, 142:529–541, 2014.
- [LLX15] Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015. arXiv preprint arXiv:1407.1296.
- [LM98] Juergen Luetttin and Gilbert Maître. Evaluation protocol for the extended M2VTS database (XM2VTSDB). Technical report, IDIAP, 1998.
- [LP66] Evgeny S Levitin and Boris T Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966.
- [LY12] Guangcan Liu and Shuicheng Yan. Active subspace: Toward scalable low-rank learning. *Neural computation*, 24(12):3371–3394, 2012.
- [MCW05] Dmitry M Malioutov, Müjdat Cetin, and Alan S Willsky. Homotopy continuation for sparse signal representation. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 5, pages v–733. IEEE, 2005.
- [MM15] Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, pages 1396–1404, 2015.
- [MZWG16] Cun Mu, Yuqian Zhang, John Wright, and Donald Goldfarb. Scalable robust matrix recovery: Frank–wolfe meets proximal methods. *SIAM Journal on Scientific Computing*, 38(5):A3291–A3317, 2016.
- [Nas00] Stephen Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14(1-2):99–116, 2000.

- [Nat95] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.
- [Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.
- [Nes04] Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.
- [Nes05] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.
- [Nes09] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.
- [Nes13] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.
- [NNS⁺14] Praneeth Netrapalli, UN Niranjan, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-convex robust pca. In *Advances in Neural Information Processing Systems*, pages 1107–1115, 2014.
- [NYD82] Arkadi Nemirovski, D-B Yudin, and E-R Dawson. Problem complexity and method efficiency in optimization. 1982.
- [OMTSK15] Tae-Hyun Oh, Yasuyuki Matsushita, Yu-Wing Tai, and In So Kweon. Fast randomized singular value thresholding for nuclear norm minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4484–4493, 2015.
- [OPT00] Michael R Osborne, Brett Presnell, and Berwin A Turlach. A new approach to variable selection in least squares problems. *IMA journal of numerical analysis*, 20(3):389–403, 2000.
- [Par17] Panos Parpas. A multilevel proximal gradient algorithm for a class of composite optimization problems. 2017.

- [PB13] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):123–231, 2013.
- [PGW⁺12] Yigang Peng, Arvind Ganesh, John Wright, Wenli Xu, and Yi Ma. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2233–2246, 2012.
- [PLRR14] Panos Parpas, Duy VN Luong, Daniel Rueckert, and Berc Rustem. A multilevel proximal algorithm for large scale composite convex optimization. 2014.
- [Sal17] Juan S Campos Salazar. *THESIS*. PhD thesis, Imperial College London, 2017.
- [Sch98] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.
- [SIL07] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.
- [SNW12] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.
- [SP16] JUAN S CAMPOS SALAZAR and PANOS PARPAS. A multigrid approach to sdp relaxations of sparse polynomial optimization problems. 2016.
- [SPZP14] Christos Sagonas, Yannis Panagakis, Stefanos Zafeiriou, and Maja Pantic. Raps: Robust and efficient automatic construction of person-specific deformable models. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1789–1796. IEEE, 2014.
- [SPZP16] Christos Sagonas, Yannis Panagakis, Stefanos Zafeiriou, and Maja Pantic. Robust statistical frontalization of human and animal faces. *International Journal of Computer Vision*, pages 1–22, 2016.
- [Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, pages 267–288, 1996.

- [Tse08] Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2008.
- [VB96] Lieven Vandenbergh and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.
- [VCWL12] Antoine Vacavant, Thierry Chateau, Alexis Wilhelm, and Laurent Lequière. A benchmark dataset for outdoor foreground/background extraction. In *Asian Conference on Computer Vision*, pages 291–300. Springer, 2012.
- [WG09] Zaiwen Wen and Donald Goldfarb. A line search multigrid method for large-scale nonlinear optimization. *SIAM Journal on Optimization*, 20(3):1478–1503, 2009.
- [WGMM13] John Wright, Arvind Ganesh, Kerui Min, and Yi Ma. Compressive principal component pursuit. *Information and Inference*, 2(1):32–68, 2013.
- [WM10] John Wright and Yi Ma. Dense error correction via-minimization. *Information Theory, IEEE Transactions on*, 56(7):3540–3560, 2010.
- [WMM⁺10] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.
- [WN99] SJ Wright and J Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.
- [WX97] Bo-Ying Wang and Bo-Yan Xi. Some inequalities for singular values of matrix products. *Linear algebra and its applications*, 264:109–115, 1997.
- [WYG⁺09] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.
- [XZ14] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

- [YWHM10] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, 19(11):2861–2873, 2010.
- [YY13] Junfeng Yang and Xiaoming Yuan. Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization. *Mathematics of Computation*, 82(281):301–329, 2013.
- [YZB⁺13] Allen Y Yang, Zihan Zhou, Arvind Ganesh Balasubramanian, S Shankar Sastry, and Yi Ma. Fast-minimization algorithms for robust face recognition. *Image Processing, IEEE Transactions on*, 22(8):3234–3246, 2013.
- [ZGLM12] Zhengdong Zhang, Arvind Ganesh, Xiao Liang, and Yi Ma. Tilt: Transform invariant low-rank textures. *International Journal of Computer Vision*, 99(1):1–24, 2012.