Journal of Math-for-Industry, Vol.2(2010B-5), pp.147–156

Math-for-industry
Education & Research Hub

Convex optimization techniques for the efficient recovery of a sparsely corrupted low-rank matrix

Silvia Gandy and Isao Yamada

Received on August 10, 2010 / Revised on August 31, 2010

Abstract. We address the problem of recovering a low-rank matrix that has a small fraction of its entries arbitrarily corrupted. This problem is recently attracting attention as nontrivial extension of the classical PCA (principal component analysis) problem with applications in image processing and model/system identification. It was shown that the problem can be solved via a convex optimization formulation when certain conditions hold. Several algorithms were proposed in the sequel, including interior-point methods, iterative thresholding and accelerated proximal gradients. In this work we address the problem from two completely different sides. First, we propose an algorithm based on the Douglas-Rachford splitting technique which has inherent convergence guarantees. Second, we propose, based on algorithms from rank minimization and sparse vector recovery, a computationally efficient greedy algorithm that scales better to large problem sizes than existing algorithms. We compare the performance of these proposed algorithms to the accelerated proximal gradients algorithm.

Keywords. PCA, rank minimization, nuclear norm minimization, sparse error, Douglas-Rachford splitting, greedy algorithms

1. Introduction

In recent years, the amount of data that needs to be analyzed, stored or simply handled has been constantly increasing. Fortunately, high-dimensional data can often be modeled as lying in (or close to) a low-dimensional subspace of the ambient dimension. Under this assumption, an approximate representation can be recovered via a principal component analysis (PCA). This problem of finding a low-rank approximation to the given data appears in a number of applications [1], e.g., in image processing, bioinformatic data, system identification.

The classical PCA problem [1] fits a low-rank matrix L to a given data matrix M while trying to minimize the norm of the error $S = M - L$:

$$\underset{(L,S)}{\text{minimize}} \; ||S||_F, \text{ s.t. } \operatorname{rank}(L) \leq r, \quad M = L + S,$$

where $||.||_F$ denotes the Frobenius norm, and M, L and $S \in \mathbb{R}^{n\times n}$. (The restriction on square matrices is only chosen to simplify the notations.) By the Schmidt-Eckart-Young Theorem [2], the minimizer of this problem is given by truncating the singular value decomposition (SVD) of M, retaining only the contribution of its r largest singular values. If $M = L_0 + S_0$ originates from a low-rank matrix L_0 ($\operatorname{rank}(L_0) \leq r$) and S_0 contains Gaussian noise (entries of S_0 distributed as $\mathcal{N}(0, \sigma^2)$ with σ^2 small), then the truncated SVD will recover a matrix $\widehat{L} \approx L_0$. However, if S_0 contains very large entries (outliers) then the truncated SVD will return a matrix that is largely deviating from L_0, even if S_0 only affects a small fraction of the entries of L_0. In other words, if S_0 models a sparse error, i.e., it represents a small number of largely corrupted entries of L_0, the truncated SVD will fail to recover L_0 (in most cases).

Recently, [3] and [4, 5] showed independently from each other that under certain assumptions (on L_0 and S_0) one can exactly recover the low-rank matrix L_0 from $M = L_0 + S_0$, where S_0 is a sparse matrix, by solving the following convex optimization problem:

$$(1) \qquad \underset{(L,S)}{\text{minimize}} \; ||L||_* + \lambda||S||_1, \quad \text{s.t.} \quad M = L + S,$$

where $\lambda \in \mathbb{R}$ is a trade-off parameter between the rank of L and the sparsity of S. Recovery was shown for $\lambda = \frac{1}{\sqrt{n}}$. Here, $||L||_* = \sum_{i=1}^{n} \sigma_i(L)$ denotes the nuclear norm, i.e. the sum of the singular values of L. $||S||_1$ is the ℓ_1 norm of the matrix S when transformed into a vector, i.e., $||S||_1 = \sum_{(i,j)} |S_{ij}|$.
We will refer to problem (1) as *Principal Component Pursuit (PCP)*, following the naming in [5]. This problem is a convex relaxation of the problem:

$$(2) \qquad \underset{(L,S)}{\text{minimize}} \; \operatorname{rank}(L) + \lambda||S||_0, \quad \text{s.t.} \quad M = L + S.$$

Here, $||S||_0$ denotes the so-called ℓ_0-norm which counts the number of non-zero entries of S.

Note that in general, the minimization problem (2) will not recover (L_0, S_0). Consider the matrix $M = e_1 e_1^*$ that contains zeros in all entries except $M_{1,1} = 1$. As this matrix is both sparse and low-rank, there is no hope of recovering the "correct" pair (L_0, S_0). The existence of minimizers of (1) is guaranteed by the coercivity [6] of the objective function $||L||_* + \lambda||S||_1$.

Recently, a number of different algorithms was proposed to solve the convex minimization problem (1) or to give near solutions to it. For example, problem (1) can easily be reformulated as a semidefinite problem and then be solved with off-the-shelf solvers that use interior point methods [3]. Interior point methods have superior convergence guarantees, however, they scale badly for large matrices, having a complexity of $O(n^6)$ to solve the associated Newton system. Thus, this approach is limited to matrices up to size of about 100×100. In [4], an iterative thresholding algorithm was introduced to solve a variant of (1), adding a term containing the norm of L and S to the cost function. This enabled them to solve the problem for matrices of size 800×800 in several hours on a normal PC. The most recent algorithms that were proposed for this setting are proximal gradient algorithms [7], where the ideas of FISTA (fast iterative soft thresholding algorithm) were applied to the PCP problem, also slightly modifying the cost function. In [7], proximal gradient algorithms were shown to solve (1) for data sizes of 2000×2000 in some hours.

This paper is organized as follows: In Section 2 we will discuss applications and give more detail on the equivalence conditions of problem (1) and (2). Then we are set to propose and analyze new algorithms for (1) and (2) which we will introduce in the sections 3 and 4. In Section 3 we will focus on the usage of the Douglas-Rachford splitting technique [8]. This will result in a new algorithm which provenly converges to a minimizer of (1) and outperforms the well performing accelerated proximal gradient algorithm proposed in [7]. Then, we will shift focus and consider directly solving (2) via a greedy approach in Section 4. Utilizing algorithms from low-rank matrix recovery/sparse vector approximation, we will propose a new greedy algorithm which is computationally very efficient and which will allow us to solve the problem for a 2000×2000 matrix in less than 2 minutes on a desktop PC. The advantage of this algorithm is the low computation cost together with good scaling properties to large-scale problem instances. However, it relies on the knowledge of the approximate rank and sparsity of the underlying decomposition (L_0, S_0). We conclude by evaluating and illustrating the performance of the proposed algorithms in Section 5.

2. Preliminaries

2.1. Principal Component Pursuit – Applications

The PCP setting appears in various applications, including:

Sensor failure: If the data in the matrix is collected via sensors, the failure of a portion of the sensors will lead to large errors in these entries. This can be modelled as a sparse error matrix.

Face recognition[4]: It is known that images from a convex Lambertian surface lie in a low-dimensional subspace (approximately 9-dimensional, [9]). Faces can be modeled as such a surface and therefore, sets of images of faces will lie close to low-dimensional surface. Sparse errors then can represent shadows, specularities or occlusions. In Section 5 we show an example, Figure 6, of such a decomposition. Note that each of the three images in Figure (6b), (6c) and (6d) corresponds to one of the columns of the matrices M, L and S respectively, as shown schematically in Figure 6a.

Video surveillance – Background modelling in video analysis[4]: The frames of a video can often be modelled as a low-rank background (almost static scene with changing illumination) and a part corresponding to movements in the scene. The part of the image, where movement occurs will be small, so we can model this as a sparse matrix. The decomposition low-rank + sparse will then return the background as low-rank and the movement as sparse matrices.

Other applications are found in *System identification*[3], i.e., the identification of a system which is the superposition of a low-order LTI system and a LTI system with a sparse impulse response, *Latent Semantic Indexing*, e.g. document-versus-term matrices used in web search engines, and *Ranking and Collaborative Filtering*, e.g., recommender systems, see [5].

2.2. Theoretical background – when does the convex relaxation yield exact recovery

In the introduction, we mentioned that the equivalence of problem (1) and (2) depends on some properties of the solution pair (L_0, S_0). In this section, we will present the recovery result of [5] which uses the following definition to quantify the incoherence of L_0.

Definition 1. (Incoherence condition with parameter μ) Let $L = U\Sigma V^T$ be a singular value decomposition of the matrix L, with $U, V \in \mathbb{R}^{n\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$ and r stands for the rank of L. We say that L fulfills the incoherence condition with parameter μ iff

$$\max_{i\in\{1,\dots,n\}} ||\mathcal{P}_U e_i||^2 \leq \frac{\mu r}{n}, \quad \max_{i\in\{1,\dots,n\}} ||\mathcal{P}_V e_i||^2 \leq \frac{\mu r}{n},$$

$$||UV^T||_\infty \leq \sqrt{\frac{\mu r}{n^2}},$$

where $||.||$ denotes the Euclidean norm of $\mathbb{R}^n$, $||.||_\infty$ the ℓ_∞-norm of $\mathbb{R}^{n\times n}$, $e_i \in \mathbb{R}^n$ are the standard basis vectors (i-th entry of $e_i = 1$, zero otherwise) and $\mathcal{P}_U : \mathbb{R}^n \to \mathbb{R}^n$ is the metric projection onto the space spanned by the columns of U.

The incoherence parameter μ has the following properties: Its value lies within the interval $\mu \in [1, \frac{n}{r}]$. It is also connected to the entries of U (and V). If all entries of U (and V) have an absolute value of $\frac{1}{\sqrt{n}}$ then μ is minimal,

i.e., $\mu = 1$. If one of the column vectors of U (or V) is a basis element e_i then μ is maximal, i.e., $\mu = \frac{n}{r}$. The incoherence parameter μ only depends on U and V and not on the singular values of L, i.e., the singular values of the low-rank matrix can take arbitrary values.

The properties of the sparse error matrix S_0 can be described as follows. We want the support of the non-zero entries to be well-distributed over the whole matrix, in order to avoid corruption of whole blocks or whole rows/columns of L_0. Thus we assume that S_0 has $s = \rho_s n^2$ non-zero entries (an equivalent description is that a fraction ρ_s of the entries of L_0 is corrupted), whose location is chosen uniformly at random from all possible support sets of size s. We only impose a condition on the support of S_0, whereas the magnitude of the non-zero entries can be arbitrary.

If (L_0, S_0) has these properties, then it was shown in [5] that with very high probability the convex minimization problem (1) has (L_0, S_0) as unique minimizer ($\lambda = \frac{1}{\sqrt{n}}$):

Theorem 1 ([5], Theorem 1.1)**.** *Suppose L_0 is $n \times n$ and obeys the incoherence property with parameter μ, and that the support set of S_0 is uniformly distributed among all sets of cardinality s. Then there is a numerical constant c such that with probability at least $1 - cn^{-10}$ (over the choice of support of S_0), Principal Component Pursuit with $\lambda = 1/\sqrt{n}$ is exact, i.e., the solution $(\widehat{L}, \widehat{S})$ of (1) satisfies $\widehat{L} = L_0$ and $\widehat{S} = S_0$, provided that*

$$\operatorname{rank}(L_0) \le c_r n \mu^{-1} (\log n)^{-2} \quad \text{and} \quad s \le c_s n^2.$$

Above, c_r and c_s are positive numerical constants. These constants do not depend on n, so for $n \to \infty$, the probability of success approaches perfect recovery, i.e., $1 - cn^{-10} \to 1$.

The equivalence of (1) and (2) was also shown for two different settings. In [10], the authors discuss changing λ for guaranteeing recovery when the fraction of known entries ρ_s gets large ($\rho_s \to 1$). In [11], recovery guarantees are derived for the case of added Gaussian noise ($M' = M + N$, where the entries of N are drawn from a Gaussian distribution).

3. Algorithm based on the Douglas-Rachford splitting technique

In this section we will derive a new algorithm for solving the Principal Component Pursuit (1) based on the Douglas-Rachford splitting technique.

We first need to fix some notation. Let $\Gamma_0(\mathcal{H})$ denote the class of all lower semicontinuous convex functions from a real Hilbert space $\mathcal{H}$ to $(-\infty, \infty]$ which are not identically equal to $+\infty$. We denote the norm on $\mathcal{H}$ with $||.||_{\mathcal{H}}$.

The Douglas-Rachford splitting technique has a long history [8, 12]; it addresses the minimization of the sum of two functions $(f + g)(x)$, where f and g are assumed to be elements of Γ_0. It was recently extended in [13] for the minimization of a sum over multiple functions in Γ_0, based on a product space formulation. The Douglas-Rachford splitting was also identified as an instance of a Mann iteration [14].

The Douglas-Rachford splitting algorithm approximates a minimizer of $(f + g)(x)$ with the help of the following sequence $(x_n)_{n\ge 0}$:

$$x_{n+1} := x_n + t_n \big\{ \operatorname{prox}_{\gamma f}\left[2 \operatorname{prox}_{\gamma g}(x_n) - x_n\right] - \operatorname{prox}_{\gamma g}(x_n) \big\}, \tag{3}$$

where $(t_n)_{n\ge 0} \subset [0, 2]$ satisfies $\sum_{n\ge 0} t_n(2 - t_n) = \infty$.
In [13], more involved versions of this process are studied in view of unavoidable numerical errors during the calculation of the iterates.
Under certain conditions (see Theorem 2 for details), the iteration process (3) converges (weakly) to a point $\widetilde{x}$, which has the property that $\operatorname{prox}_{\gamma g}(\widetilde{x})$ is a minimizer of $(f+g)(x)$.

The proximal map, $\operatorname{prox}_{\gamma f}$, is defined as

$$\operatorname{prox}_{\gamma f} x = \underset{y\in\mathcal{H}}{\arg\min} \left\{ f(y) + \frac{1}{2\gamma} ||x - y||^2_{\mathcal{H}} \right\}. \tag{4}$$

Let $\mathcal{H}_0$ denote the Hilbert space given by the 2-fold Cartesian product of $\mathbb{R}^{n\times n}$, i.e., $\mathcal{H}_0 := \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$.
As inner product on $\mathcal{H}_0$, we use for $X := (X_1, X_2)$, $Y := (Y_1, Y_2) \in \mathcal{H}_0$:

$$\begin{aligned} \langle X, Y\rangle_{\mathcal{H}_0} &:= \frac{1}{2}\langle X_1, Y_1\rangle + \frac{1}{2}\langle X_2, Y_2\rangle \\ &= \frac{1}{2}\operatorname{trace}(X_1^T Y_1) + \frac{1}{2}\operatorname{trace}(X_2^T Y_2). \end{aligned} \tag{5}$$

The norm on $\mathcal{H}_0$ is induced by $\langle ., .\rangle_{\mathcal{H}_0}$, i.e.,

$$||X||^2_{\mathcal{H}_0} := \langle X, X\rangle_{\mathcal{H}_0} = \frac{1}{2}||X_1||^2_F + \frac{1}{2}||X_2||^2_F.$$

We can recast problem (1) into the unconstrained minimization of $(f + g)(x)$ as follows:

$$\underset{Z\in\mathcal{H}_0}{\text{minimize}} \quad f(Z) + g(Z),$$

where $Z = (Z_1, Z_2)$, $D = \big\{ Z \in \mathcal{H}_0 \;\big|\; M = Z_1 + Z_2 \big\}$,

$$f(Z) := \sum_{i=1}^{2} f_i(Z_i) = ||Z_1||_* + \lambda ||Z_2||_1 \tag{6}$$

and

$$g(Z) := i_D(Z) = \begin{cases} 0, & \text{if } Z \in D \\ +\infty, & \text{otherwise}. \end{cases} \tag{7}$$

This problem formulation is equivalent to (1), so we only need to identify the proximal maps of f and g in order to use algorithm (3).

The proximal map of f is given by

$$\begin{aligned} \operatorname{prox}_{\gamma f} X &= \underset{Y\in\mathcal{H}_0}{\arg\min} \left\{ \sum_{i=1}^{2} f_i(Y_i) + \frac{1}{2\gamma} ||Y - X||^2_{\mathcal{H}_0} \right\} \\ &= \underset{Y\in\mathcal{H}_0}{\arg\min} \left\{ \sum_{i=1}^{2} f_i(Y_i) + \frac{1}{2} \sum_{i=1}^{2} \left(\frac{1}{2\gamma} ||Y_i - X_i||^2_F \right) \right\} \\ &= \left(\operatorname{prox}_{2\gamma f_1} X_1, \operatorname{prox}_{2\gamma f_2} X_2 \right), \end{aligned}$$

where we used (4) and (5).

The proximal map of the nuclear norm function, i.e., f_1, is the shrinkage operator; this operator performs a soft-thresholding operation on the singular values of the matrix it is applied to.

$$\begin{aligned}\mathrm{prox}_{\tau f_1}\, T &= \underset{Y\in\mathbb{R}^{n\times n}}{\arg\min}\left\{||Y||_* + \frac{1}{2\tau}||Y-T||_F^2\right\}\\ &= \mathrm{shrink}\,(T,\tau)\,.\end{aligned}$$

Let $T = U_T\Sigma_T V_T^T$ be a singular value decomposition of the matrix $T\in\mathbb{R}^{n\times n}$. Then, $\Sigma_T = \mathrm{diag}(\sigma_1(T),\ldots,\sigma_r(T))$ is a diagonal matrix containing the singular values $\sigma_k(T)$ on the diagonal. Define $\widetilde{\Sigma}_T$ as the diagonal matrix that contains the singular values shrunk by τ on the diagonal, i.e.,

$$\widetilde{\Sigma}_T := \mathrm{diag}(\max\{\sigma_1(T)-\tau,0\},\ldots,\max\{\sigma_r(T)-\tau,0\}).$$

Then, the singular value shrinkage operator is given by

$$\mathrm{shrink}(T,\tau) := U_T\widetilde{\Sigma}_T V_T^*.$$

So the first component of the proximal map of f is:

$$\mathrm{prox}_{2\gamma f_1}\, X_1 = \mathrm{shrink}\,(X_1,\ 2\gamma)\,,$$

The second component's proximal map reduces to a soft-thresholding of the entries of the matrix.

$$\begin{aligned}\mathrm{prox}_{2\gamma f_2}\, X_2 &= \underset{Y_2\in\mathbb{R}^{n\times n}}{\arg\min}\left\{\lambda||X_2||_1 + \frac{1}{2\cdot(2\gamma)}||Y_2-X_2||_F^2\right\}\\ &= \mathrm{soft\_threshold}\,(X_2,\ 2\gamma\lambda)\end{aligned}$$

The soft-thresholding operator is defined entry-wise

$$\mathrm{soft\_threshold}(X_{ij},\tau) = \mathrm{sign}(X_{ij})\cdot\max\,(0,|X_{ij}|-\tau)\,.$$

Finally, the proximal map of the indicator function i_D is the metric projection $\mathcal{P}_D$ onto the set D is given by:

$$\begin{aligned}\mathcal{P}_D(Z_1,Z_2) &:= \underset{(X_1,X_2)\in D}{\arg\min}\,||(X_1,X_2)-(Z_1,Z_2)||_{\mathcal{H}_0}\\ &= (M+Z_1-Z_2,\,M-Z_1+Z_2)/2.\end{aligned}$$

Having identified all ingredients, we can now specialize the Douglas-Rachford splitting algorithm of (3) to this choice of f and g ((6) and (7)):

(DR-PCP): Douglas-Rachford splitting algorithm

input: M, t_k, λ
initialization: $L^{(0)} = S^{(0)} = 0$

repeat until convergence:
$(\widehat{L},\widehat{S}) = \mathcal{P}_D(L^{(k)},S^{(k)})$
$L^{(k+1)} = L^{(k)} + t_k\left(\mathrm{shrink}(2\widehat{L}-L^{(k)},2\gamma)-\widehat{L}\right)$
$S^{(k+1)} = S^{(k)} + t_k\left(\mathrm{soft\_threshold}(2\widehat{S}-S^{(k)},2\lambda\gamma)-\widehat{S}\right)$
k = k+1

output: $(L,S) = \mathcal{P}_D(L^{(k)},S^{(k)})$

Only a slight change in the definition of the set D is necessary in order to cover the noisy case, where instead of equality $M = L+S$, we will consider $||M-L-S||_F\le\delta$. Let

$$D_{\mathrm{noise}} := \left\{\ (Z_1,Z_2)\ \middle|\ ||M-Z_1-Z_2||_F\le\delta\ \right\}$$

and $V := (Z_1,Z_2) - \mathcal{P}_D(Z_1,Z_2)$. Then the metric projection onto D_{noise}, i.e.,

$$\mathcal{P}_{D_{\mathrm{noise}}}(Z_1,Z_2) = \begin{cases}(Z_1,Z_2), & \text{if } ||V||_{\mathcal{H}_0}\le\delta,\\ \mathcal{P}_D(Z_1,Z_2)+\delta\frac{V}{||V||_{\mathcal{H}_0}}, & \text{otherwise,}\end{cases}$$

is the proximity operator of

$$g'(Z) := i_{D_{\mathrm{noise}}}(Z) = \begin{cases}0 & \text{if } Z\in D_{\mathrm{noise}}\\ \infty & \text{otherwise}\,.\end{cases} \tag{8}$$

Another way to cover the noisy case is to minimize $f+g''$ with g'' defined as

$$g''(Z_1,Z_2) := \frac{\mu}{2}||M-Z_1-Z_2||_F^2\,. \tag{9}$$

The corresponding proximity operator $\mathrm{prox}_{\gamma g''}(X)$ is

$$\begin{aligned}&\mathrm{prox}_{\gamma g''}(X_1,X_2)\\ &\quad= \frac{1}{(\mu+\nu)^2-\mu^2}\begin{pmatrix}\nu\mu M+\nu(\nu+\mu)X_1-\nu\mu X_2\\ \nu\mu M-\nu\mu X_1+\nu(\nu+\mu)X_1\end{pmatrix}^T\end{aligned}$$

where $\nu := \frac{1}{2\gamma}$.

An algorithm for the noisy problem setting is then immediately obtained by replacing the proximal map of g in (DR-PCP) with the proximal map of g' or g''.

3.1. Convergence of the Douglas-Rachford splitting

In order to state a convergence theorem of the Douglas-Rachford splitting algorithm, we need the notion of domain ($\mathrm{dom}(f)$) of a function f: Define $\mathrm{dom}(f) := \{x\in\mathcal{H}\mid f(x)<\infty\}$ and

$$\begin{aligned}&\mathrm{dom}(f)-\mathrm{dom}(g)\\ &\quad:= \left\{x_1-x_2\in\mathcal{H}\ \middle|\ x_1\in\mathrm{dom}(f)\wedge x_2\in\mathrm{dom}(g)\right\}.\end{aligned}$$

The following theorem states the convergence properties of the Douglas-Rachford splitting algorithm (3):

Theorem 2 ([14]). *Let $f,g\in\Gamma_0(\mathcal{H})$ satisfy $S := \underset{x\in\mathcal{H}}{\arg\min}\{f(x)+g(x)\}\neq\emptyset$. Suppose that*

$$\begin{aligned}&\mathrm{cone}(\,\mathrm{dom}(f)-\mathrm{dom}(g))\\ &\quad:= \bigcup_{\lambda>0}\left\{\lambda x\ \middle|\ x\in\mathrm{dom}(f)-\mathrm{dom}(g)\right\}\end{aligned} \tag{10}$$

is a closed subspace of $\mathcal{H}$. Then the sequence $(x_n)_{n=0}^{\infty}$ generated by (3) converges weakly to a point in $(\mathrm{prox}_{\gamma g})^{-1}(S)$. This holds for any initial value $x_0\in\mathcal{H}$, any $\gamma\in(0,\infty)$ and any $(t_n)_{n\ge0}\subset[0,2]$ that satisfies $\sum_{n\ge0}t_n(2-t_n)=\infty$.

We will show that condition (10) holds for the DR-PCP algorithms.

Lemma 1. *The qualifying condition (10) holds for the functions f and g as defined in (6) and (7). The same is true for (f, g') and (f, g'') as defined in (8) and (9).*

Proof. The domain of f, $\mathrm{dom}(f)$, is the whole space $\mathcal{H}_0$, as $f(X_1, X_2) = ||X_1||_* + \lambda||X_2||_1 < \infty$, $\forall (X_1, X_2) \in \mathcal{H}_0$. From (7) it follows immediately that

$$\mathrm{dom}(g) = \{(X_1, X_2) \mid M = X_1 + X_2\}.$$

Thus, $\mathrm{dom}(f) - \mathrm{dom}(g) = \mathcal{H}_0$, therefore $\mathrm{cone}(\mathcal{H}_0) = \mathcal{H}_0$ and (10) holds. As both $\mathrm{dom}(g') = D_{\mathrm{noise}}$ and $\mathrm{dom}(g'') = \mathcal{H}_0$ are non-empty, and $\mathrm{dom}(f) = \mathcal{H}_0$, we have $\mathrm{dom}(f) - \mathrm{dom}(g') = \mathrm{dom}(f) - \mathrm{dom}(g'') = \mathcal{H}_0$. Thus, (10) also holds for g' and g''. □

Applying Lemma 1 to Theorem 2, we can now state the convergence of algorithm (DR-PCP):

Theorem 3. *Let all parameters of the algorithm (DR-PCP) be chosen as in Theorem 2. Then $\mathcal{P}_D(L^{(k)}, S^{(k)})$ converges to a minimizer of (1).*

4. Greedy algorithms

We were inspired by recent work on greedy algorithms for compressed sensing and low-rank matrix recovery to devise a greedy algorithm for problem (1), as it contains both an ℓ_1-minimization part and a nuclear norm minimization part.[1] We build upon two algorithms, namely CoSaMP [16] and Admira [17].

CoSaMP is a greedy algorithm which solves the s-sparse vector approximation problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ ||Ax - b||_2, \quad \text{s.t.} \quad ||x||_0 \le s. \tag{11}$$

Linear convergence of the algorithm to the sparsest vector lying close to the affine subspace defined by $Ax = b$ can be guaranteed if the linear map $A : \mathbb{R}^n \to \mathbb{R}^m$ fulfills the restricted isometry property with a small constant δ [16]. The CoSaMP algorithm uses the following iterative process to obtain a solution:

CoSaMP
input: A, b, s
initialization: $\widehat{x} = 0$, $\widehat{\Omega} = 0$
repeat until convergence:
$\Omega' = \underset{\Omega\subset\{1,\dots,n\}}{\arg\max} \left\{ ||\mathcal{P}_\Omega\left(A^*(b - A\widehat{x})\right)||_2 : |\Omega| \le 2s \right\}$
$\widetilde{\Omega} = \widehat{\Omega} \cup \Omega'$
$\widetilde{x} = \underset{x\in\mathbb{R}^n}{\arg\min} \left\{ ||b - Ax||_2 : \mathrm{supp}(x) \in \widetilde{\Omega} \right\}$
$\widehat{\Omega} = \underset{\Omega\subset\{1,\dots,n\}}{\arg\max} \{\mathcal{P}_\Omega(\widetilde{x}) : |\Omega| \le s\}$
$\widehat{x} = \mathcal{P}_{\widehat{\Omega}}(\widetilde{x})$
output: $\widehat{x}$

[1] A short version of the content of this section was presented by the authors at the conference ICASSP 2010, [15].

The first line of the inner loop finds the best possible support set of size at most $2s$ ($|\Omega|$ returns the number of elements of Ω) for the approximation of the residual $(b - A\widetilde{x})$ from the last iteration. Here, $A^* : \mathbb{R}^m \to \mathbb{R}^n$ denotes the adjoint of A and $\mathcal{P}_\Omega : \mathbb{R}^n \to \mathbb{R}^n$ stands for the projection onto the entries indexed by the set Ω, as given by the entry-wise definition: $(\mathcal{P}_\Omega(\xi))(i) = \xi_i$ if $i \in \Omega$ and $(\mathcal{P}_\Omega(\xi))(i) = 0$ if $i \notin \Omega$. Then the best solution using the union of this support set and the support set from the previous iteration is calculated ($\widetilde{x}$, having at most $3s$ non-zeros). Finally, $\widetilde{x}$ is then truncated to contain only s non-zeros.

The second algorithm, Admira (Atomic Decomposition for Minimum Rank Approximation, [17]), solves the rank-r approximation problem:

$$\underset{X\in\mathbb{R}^{n\times n}}{\text{minimize}}\ ||\mathcal{A}(X) - b||_2, \quad \text{s.t.} \quad \mathrm{rank}(X) \le r. \tag{12}$$

The convergence properties are similar to CoSaMP. Linear convergence of Admira to the best rank-r approximation of the minimal rank solution of $\mathcal{A}(X) = b$ is guaranteed [17], if the linear map $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ fulfills the rank-restricted isometry property [18] with a low constant δ_r. This condition holds with high probability if, for example, the matrix representation of the linear map $\mathcal{A}$ contains Gaussian distributed entries and the number of measurements m is large enough [18]. However, the special case when $\mathcal{A}$ "samples" a subset of the entries of the matrix, as appearing in the matrix completion problem [19], is not covered in the analysis of the convergence rate [17]. The update step of Admira parallels the CoSaMP algorithm. We use the same notation as in [17]. Ψ stands for a set of rank-one matrices, the so-called atoms. $\mathbb{O}$ denotes the set of atoms of $\mathbb{R}^{n\times n}$, $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{R}^{n\times n}$ denotes the adjoint of $\mathcal{A}$ and $\mathcal{P}_\Psi(Z) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ denotes the metric projection of a matrix Z onto the space spanned by the elements of the set Ψ.

Admira
input: $\mathcal{A}, b, r$
initialization: $\widehat{X} = 0$, $\widehat{\Psi} = 0$
repeat until convergence:
$\Psi' = \underset{\Psi\subset\mathbb{O}}{\arg\max} \left\{ ||\mathcal{P}_\Psi\left(\mathcal{A}^*(b - \mathcal{A}(\widehat{X}))\right)||_F : |\Psi| \le 2r \right\}$
$\widetilde{\Psi} = \Psi' \cup \widehat{\Psi}$
$\widetilde{X} = \underset{X\in\mathbb{R}^{n\times n}}{\arg\min} \left\{ ||b - \mathcal{A}(X)||_2 : X \in \mathrm{span}(\widetilde{\Psi}) \right\}$
$\widehat{\Psi} = \underset{\Psi\subset\mathbb{O}}{\arg\max} \left\{ ||\mathcal{P}_\Psi(\widetilde{X})||_F : |\Psi| \le r \right\}$
$\widehat{X} = \mathcal{P}_{\widehat{\Psi}}(\widetilde{X})$
output: $\widehat{X}$

The first line of the inner loop finds the best possible set of atoms of size at most $2r$ for the approximation of the residual from the last iteration. Next, the best solution using the union of these atoms and the atoms from the previous iteration is calculated ($\widetilde{X}$, containing at most $3r$ atoms). Finally, $\widetilde{X}$ is truncated to use at most r atoms.

Summarizing the above, the algorithm CoSaMP finds the sparsest solution by solving the problem of finding the best s-sparse approximation w.r.t. the error on the the linear constraints. Along the same lines, ADMIRA finds the minimal rank solution by solving the problem of finding the best rank-r approximation. Of course, the solution of CoSaMP can only be exact, if the sparsest solution has a sparsity less or equal to s. The same holds for ADMIRA, which returns the minimal rank solution if the rank of the minimal rank solution is less than or equal to r.

We will use the same approach for problem (2). Given a data matrix M, we fix a target rank r and a target sparsity s and solve:

$$(13)\quad \arg\min_{(L,S)} ||M-S-L||_F \text{ s.t. } \operatorname{rank}(L)\le r, ||S||_0 \le s$$

The idea is to alternatingly use CoSaMP and ADMIRA to update L and S respectively. The problem setting (13) tries to split a sum M into its parts (L_0, S_0). In contrast to (11) and (12), there are no linear measurements (A /$\mathcal{A}$) involved. Therefore, direct application of ADMIRA and CoSaMP simplifies to the following algorithm, which we name Atomic Decomposition Alternating Least Squares (AD_ALS):

Algorithm (AD_ALS):

input: M, s, r

initialization: $L^{(0)} = S^{(0)} = 0$, $\Psi^{(0)} = \emptyset$, $k = 0$

repeat until convergence:

1. *Update S:*
$S^{(k+1)} = \arg\min_{S\in\mathbb{R}^{n\times n}} \{||M-L^{(k)}-S||_F : |\operatorname{supp}(S)| \le s\}$
2. *Update L:*
$\Psi' = \arg\max_{\Psi\subset\mathbb{O}} \{||\mathcal{P}_\Psi(M-S^{(k+1)}-L^{(k)})||_F : |\Psi| \le 2r\}$
$\widetilde{\Psi} = \Psi' \cup \Psi^{(k)}$
$\widetilde{L} = \arg\min_{L\in\mathbb{R}^{n\times n}} \left\{||M-S^{(k+1)}-L||_F : L \in \operatorname{span}(\widetilde{\Psi})\right\}$
$\Psi^{(k+1)} = \arg\max_{\Psi\subset\mathbb{O}} \left\{||\mathcal{P}_\Psi(\widetilde{L})||_F : |\Psi| \le r\right\}$
$L^{(k+1)} = \mathcal{P}_\Psi(\widetilde{L})$

$k = k+1$

output: $(L,S) = (L^{(k)}, S^{(k)})$

The update of S simply computes the best s-sparse approximation to the residual $M - L^{(k)}$. (This is all that remains from the CoSaMP algorithm). The step to compute $L^{(k+1)}$ is a little bit more involved. The first step in the update of L is the calculation of a set of $2r$ (rank-one) matrices that represent the singular spaces of the best rank-$2r$-approximation of the new residual $M - L^{(k)} - S^{(k+1)}$. Then, the best approximation to $M - S^{(k+1)}$ within the space spanned by the elements in Ψ' is calculated. Thus, $\widetilde{L}$ is a linear combination of the $2r$ atoms computed before and the r rank-one matrices which are derived from the singular vectors of $L^{(k)}$. ([17]: $\Psi^{(k+1)}$ contains the r matrices $\psi_i^{(k+1)}$ which are the products of the right and left singular vectors u_i and v_i belonging to r largest singular values of $\widetilde{L}$: $\psi_i^{(k)} = u_i v_i^T$, $\Psi^{(k)} = \{\psi_i^{(k)}\}$).

Algorithm (ALS):

We also introduce a further simplified version of AD_ALS, the alternating least squares (ALS) algorithm.

ALS updates S and L via alternately minimizing the residual. Therefore, step 1 (update of S) stays as stated above. Step 2 is changed to:

2'. *Update L:*
$L^{(k+1)} = \arg\min_{L} \{||M - S^{(k+1)} - L||_F : \operatorname{rank}(L) \le r\}$,

whose minimizer can be computed via a truncated SVD of the matrix $M - S^{(k+1)}$. The algorithms ALS and AD_ALS depend strongly on a good choice of the target rank r and the target sparsity s. If no target rank r and target sparsity s is given, then it is reasonable to run the algorithm for increasing pairs (r, s) and to choose the minimal pair $(\bar{L}(r,s), \bar{S}(r,s))$ that achieves a small enough residual $||M - \bar{L} - \bar{S}||_F$.

It is not an easy task to give any theoretical convergence guarantees. The algorithms CoSaMP and ADMIRA deduce their convergence rates from properties of the linear map A (resp. $\mathcal{A}$) when applied to sparse vectors/low-rank matrices. In the current setting, no such tool is available.

The error is guaranteed to decrease in each step, but the algorithm can become stationary if a new iterate of the sequence $L^{(k)}$ decreases the residual such that $S^{(k+1)} = S^{(k)}$. Then, the algorithm stays stationary with this pair $(L^{(k)}, S^{(k)})$.

5. Numerical Experiments

We randomly generated an input pair (L_0, S_0) as follows: $L_0 := X_L X_R^T$, where $X_L, X_R \in \mathbb{R}^{n\times r}$ with the entries of X_L, X_R being i.i.d. Gaussian variables with mean 0 and variance 1. We chose the support set of S_0 uniformly at random from all support sets of size $\rho_s n^2$. The non-zero entries are independently drawn from a uniform distribution on $[-500, 500]$. We fixed $\lambda = \frac{1}{\sqrt{n}}$ in the experiments.

We used the same simulation setting as in [7], in order to be able to roughly compare the results. The numerical experiments in [7] showed that the proximal gradient algorithms outperform iterative thresholding. Therefore we will limit our comparison to the proximal gradient algorithm APG, [7].

Implementation: We used Lanczos iterations to calculate the truncated SVDs, i.e., we only calculate the largest singular values and their singular vectors, in order to avoid the computational burden of a complete singular value decomposition.

The tests were run on an Intel (R) Core(TM) 2 Quad CPU (four cores) with 3.0 GHz. In [7] a Mac Pro computer having eight cores and 2.8 GHz was used. Thus, the machine specifications are rather close, so we expect comparable computation times.

The algorithms are empirically stable w.r.t. additional small Gaussian noise affecting M, as the problem formulation (13) is robust to additional noise contained within M.

We ran experiments using a perturbed data matrix M,

$$M = L_0 + S_0 + N, \tag{14}$$

where N contains Gaussian noise, i.e., the entries of N are distributed as $\mathcal{N}(0, \sigma^2)$, with sufficiently small σ^2. The algorithms recovered (L_0, S_0) up to noise level (as given by $||N||$ which depends on σ) as long as the pair (L_0, S_0) was low-rank and sparse enough. In [11], recovery guarantees for the data model (14) which contains the added noise N are available.

We compared ALS and AD_ALS on several runs, with typical outputs shown in Figure 1. The number of necessary iterations was slightly higher for ALS than for AD_ALS, but the overall behavior of the ADMIRA-inspired algorithm and ALS is similar, as expected. ALS is computationally less expensive, as it avoids computing the projection of $D - E^{(k)}$ onto the $3r$-dimensional subspace, which is an additional least squares problem.

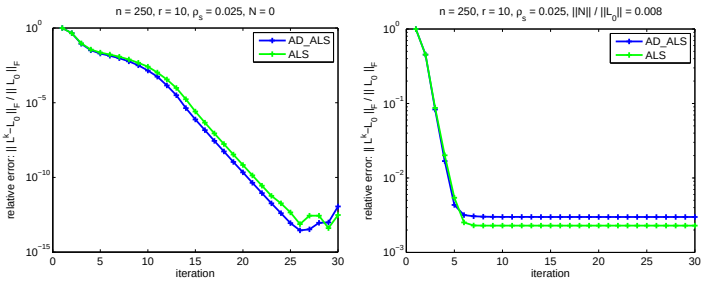


Figure 1: Two typical runs of the algorithms AD_ALS and ALS: On the left, no Gaussian error (N, see (14) was added and the relative error on L_0 approaches zero. On the right, the relative error reaches the noise level after few iterations.

Figure 2 shows the recovery performance of AD_ALS. For small values of $r = \text{rank}(L_0)$ and a very sparse matrix S_0, the figure shows perfect recovery in all tries. This result shows the same type of transition from recovery in all tries to non-recovery as is to be expected for the direct minimization of the convex problem formulation (1). In [4], a similar figure was given for a proximal gradient algorithm.

Figure 3 was obtained from the same setting as Figure 2. We additionally introduced a mismatch between the correct rank/sparsity of (L_0, S_0) and the target rank/target sparsity of the algorithm. The greedy algorithms depend strongly on the parameters. It is obvious that the performance degraded from Figure 2 to Figure 3. However, for small values of the rank and the sparsity, perfect recovery occurs despite the presence of the mismatch.

Next, we compared the computation time of our algorithms (AD_ALS / ALS), DR-PCP and APG (accelerated proximal gradient, [7]) in Table 1. The setting is $r = 0.05n$ and $\rho_s = 0.1$. AD_ALS is outperforming APG and DR-PCP already in the case of small problem sizes, i.e., where $n \in \{100, \ldots, 800\}$. The times for ALS are not shown, but ALS converges even faster. We set the stopping criterion such that the algorithms stopped with a relative error of about 10^{-5}. The algorithm DR-PCP converges slightly faster than the APG algorithm. The algorithms APG and DR-PCP have the same amount of prior information, as neither of them assumes knowledge of a target rank and target sparsity. So the speed-up of the algorithm AD_ALS is based on this additional information.

Table 2 shows the comparison of the computation time for a large-scale problem ($n = 2000$). These experiments demonstrate the effectiveness of the proposed ALS method. We used a heuristic for the Douglas-Rachford splitting algorithm to obtain the values in this table. As the performance depends largely on the singular value decomposition we approximated the output of the shrinkage operator as follows. We introduced a bound of $0.2n$ as maximal number of singular values of $2\widehat{L} - L^{(k)}$ that were computed and shrunk. Once the iterations get close to the solution, this bound has no effect, however in the first iterations this heuristic speeds up the calculation of the shrinkage operator. With this heuristic, our algorithm was faster than APG.

Due to a non-optimized ad-hoc implementation that we used to test AD_ALS, we ran into memory problems for problem sizes of $n = 2000$. Therefore, we only compare APG and ALS. ALS is by far faster than APG, recovering the input pair (L_0, S_0) in less than 2 minutes (compared to some hours for APG).

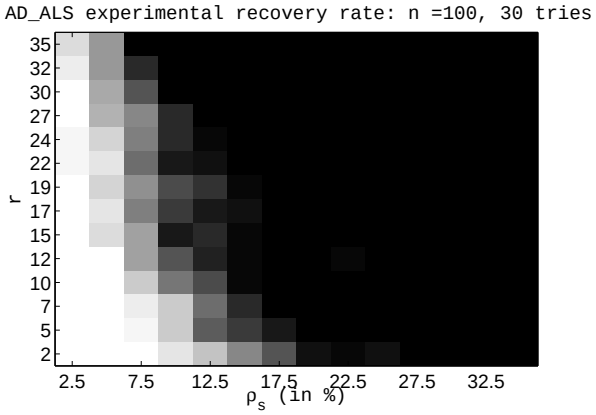


Figure 2: Recovery of (L_0, S_0) depending on the rank r of L_0 and the sparsity ρ_s of S_0. White denotes recovery in all experiments, whereas black denotes no recovery. The matrix size was fixed to $n = 100$ and 30 trials run for each pair (r, ρ_s). The algorithm was stopped after 50 iterations.

We experimented with image data and the ALS algorithm next. Figures 4 and 5 show two examples of images that are a superposition of a background image and a sparse addition. The background in these examples is low-rank, thus making the separation of the two layers by Principal Component Pursuit possible. In the part (a) of the figures, we show the input to the ALS algorithm. The algorithm is able to identify the position of the errors and to split the input into the background image (b) and the text image (c). We inverted the colors black and white in the representation of (c) purely for esthetical reasons.

The following example is an application of the Douglas-

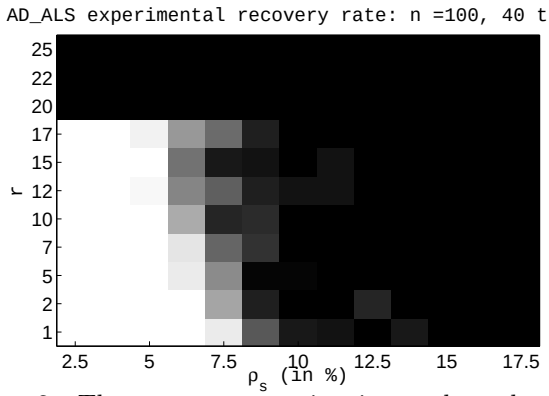


Figure 3: The parameter setting is equal to the one in Figure 2, but with a non-optimal setting of the target rank and the target sparsity. The mismatch was set to 5 percent, i.e., $r^{\text{target}} = 1.05 \cdot r = 1.05 \cdot \text{rank}(L_0)$ and $\rho_s^{\text{target}} = 1.05 \cdot \rho_s$

Table 1: Comparison of the computation time for different matrix sizes n. APG (Accelerated Projected Gradient) is significantly slower than AD_ALS ($r = 0.05n$, $\rho_s = 0.1$. The algorithm DR-PCP is slightly faster than the APG algorithm.

	APG [7]		AD_ALS	
n	$\frac{\|\|L-L_0\|\|_F}{\|\|L_0\|\|_F}$	time (s)	$\frac{\|\|L-L_0\|\|_F}{\|\|L_0\|\|_F}$	time (s)
100	3.4×10^{-5}	3.1	9.9×10^{-6}	0.35
200	2.2×10^{-5}	16	4.9×10^{-6}	1.1
400	1.6×10^{-5}	111	7.1×10^{-6}	8.2
800	1.1×10^{-5}	766	4.5×10^{-6}	68.9

	DR-PCP	
n	$\frac{\|\|L-L_0\|\|_F}{\|\|L_0\|\|_F}$	time (s)
100	1.15×10^{-5}	2.2
200	6.5×10^{-6}	10.3
400	1.15×10^{-5}	39
800	1.28×10^{-5}	220

Rachford splitting algorithm DR-PCP (using the noisy formulation with D_{noise}) to images from the Yale B database [20]. This database contains images of faces taken under different illumination conditions. The images are already aligned, so the position of the face in the different images does not change. We used 58 images and converted each image into a vector. These vectors were used as the columns of the matrix M, as shown in Figure 6a. After decomposing M into the matrices L and S, we reinterpreted the columns of these matrices as images. Figures (6b) - (6d) show the results of such a decomposition. The "sparse" image (column vector of the matrix S) contains the specularities and shadows that were contained in the images, whereas the low-rank part is a regularized version of the original image.

Table 2: Large-size example: Alternating least squares recovers (L_0, S_0) of sizes 2000×2000 matrix in less than 2 minutes, even if the rank is $0.1n = 200$ and 400 000 entries are corrupted. The accelerated proximal gradient algorithm needs some hours in the same setting.

n = 2000, rel. error $\frac{\|\|L-L_0\|\|_F}{\|\|L_0\|\|_F}$				
$\text{rank}(L_0)/n$	$\|\|S_0\|\|_0/n^2$	APG [7]	ALS	DR-PCP
0.05	0.05	14 300 s	29.9 s	2037 s
0.05	0.1	14 700 s	67.8 s	2875 s
0.1	0.1	14 300 s	95.5 s	3652 s

6. Discussion

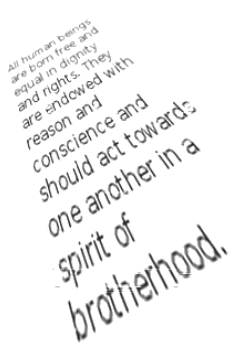
In this paper, we proposed new algorithms for the Principal Component Pursuit based on two different principles. The algorithm DR-PCP is based on the Douglas-Rachford splitting technique. It shows similar to better results compared to the well-performing accelerated proximal gradient algorithm [7]. We proved the convergence of this algorithm and its variants to a minimizer of the Principal Component Pursuit problem. The algorithm DR-PCP does not rely on any information on the rank/sparsity level of the underlying decomposition. We showed results of the algorithm's performance on a face recognition task, where specularities and shadows were identified correctly by the algorithm.

Then we presented two computationally efficient greedy algorithms (AD_ALS and ALS) for the robust principal component analysis problem. The main point of interest of these greedy algorithms is that they are the analog of CoSaMP and ADMIRA for the PCP problem. AD_ALS alternates the update of S and L by using CoSaMP to compute $S^{(k+1)}$ and ADMIRA to compute $L^{(k+1)}$. As there is no linear map A (resp. $\mathcal{A}$) that acts on L or S involved in the linear constraint, the proposed algorithm essentially boils down to the alternating least squares algorithm, given by ALS. ALS also alternates the updates of S and L, but ALS is even simpler than AD_ALS because $L^{(k+1)}$ in ALS is obtained via one truncated SVD. Although these are very straightforward algorithms, they still show very good performance and can easily handle large data matrices. However, in order to be of practical importance it is necessary to have a good estimate of the expected rank r of the low-rank matrix L_0 and the sparsity level of S_0 as they are crucial parameters of the algorithm. For the optimal parameter choice of r and s we outperform the state-of-the-art algorithm APG, the accelerated proximal gradient algorithm.

Acknowledgments

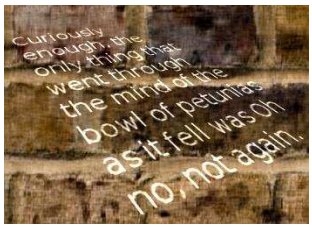
S.G. acknowledges funding as Research Fellow of the Japan Society for the Promotion of Science (JSPS), DC2 grant.

(a) input to the algorithm

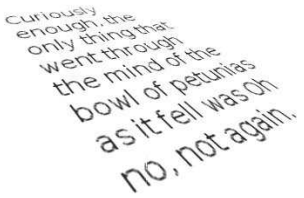


(b) low-rank output (c) sparse (b/w inverted)

Figure 4: Splitting of an input image into a low-rank scene (image of the facade of a building) and a sparse image, representing the text part of the image



(a) input to the algorithm



(b) low-rank output (c) sparse (b/w inverted)

Figure 5: Splitting of an input image into a low-rank image of a wall and a sparse image, representing the text part of the image

References

[1] I. T. Jolliffe: *Principal Component Analysis*, Springer-Verlag, 1986.

[2] A. Ben-Israel and T. Greville, *Generalized Inverses – Theory and Applications*, CMS Books in Mathematics, Springer, second edition, 2003.

[3] V. Chandrasekaran, S. Sangavi, P. Parrilo and A. Willsky, *Sparse and low-rank matrix decompositions*, IFAC Symposium on System Identification, 2009.

[4] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma *Robust PCA: Exact Recovery of Corrupted Low-Rank Matrices via Convex Optimization*, Proceedings of Neural Information Processing Systems (NIPS), December 2009.

[5] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust Principal Component Analysis?*, Submitted for publication, 2009.

[6] J.-B. Hiriart-Urrut and C. Lemaréchal, *Fundamentals of convex analysis*, Springer, 2004.

[7] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen and Y. Ma, *Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix*, UIUC Technical Report UILU-ENG-09-2214, July 2009.

[8] J. Douglas and H. Rachford, *On the numerical solution of heat conduction problems in two and three space variables*, Transactions of the American Mathematical Society, vol. 82, pp. 421–439, 1956.

[9] R. Basri and D. Jacobs, *Lambertian Reflectance and Linear Subspaces*, NEC Research Institute Technical Report 2000-172R, Weizmann Institute of Science Technical Report MCS00-21, 2000.

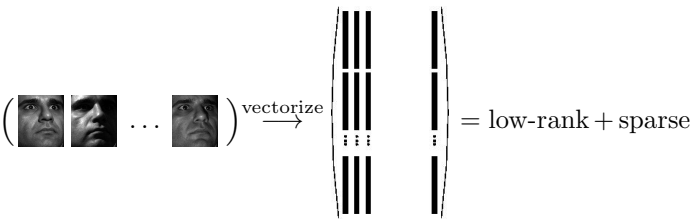
[10] A. Ganesh, J. Wright, X. Li, E. J. Candès and Y. Ma. *Dense error correction for low-rank matrices via Principal Component Pursuit*, Proceedings of International Symposium on Information Theory, 2010.

[11] Z. Zhou, J. Wright, X. Li, E. J. Candès and Y. Ma, *Stable Principal Component Pursuit*, Proceedings of International Symposium on Information Theory, 2010.

[12] P. L. Combettes and J.-C. Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE J. Sel. Top. Signal Process., vol. 1 (4), pp. 564–574,2007.

[13] P. L. Combettes and J.-C. Pesquet, *A proximal decomposition method for solving convex variational inverse problems*, Journal of Inverse Problems, vol. 25 (6), 065014, 2008.

[14] I. Yamada, M. Yukawa and M. Yamagishi, *Minimizing the Moreau envelope of nonsmooth convex functions over the fixed point set of certain quasi-nonexpansive mappings*, in: Fixed-Point Algorithms



(a) Face recognition: a set of images is transformed into vectors which are used as the column vectors of the data matrix M. Splitting it into a low-rank and a sparse part

(b) Original image (c) Low-rank part (d) Sparse part

Figure 6: Example using the Yale B database [20]. The image is split into a low-rank part and a sparse part which represents specularities and shadows are extracted. The Douglas-Rachford splitting algorithm of Section 3 was used to generate the image from an input set of 58 images.

for Inverse Problems in Science and Engineering, (H. H. Bauschke, R. Burachik, P. L. Combettes, V. Elser, D. R. Luke, H. Wolkowicz, Editors). New York: Springer-Verlag, 2010.

[15] S. Gandy and I. Yamada, *Alternating minimization techniques for the efficient recovery of a sparsely corrupted low-rank matrix*, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2010.

[16] D. Needell and J. A. Tropp, *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples*, Applied and Computational Harmonic Analysis, vol. 26 (3), pp. 301–321, 2008.

[17] K. Lee and Y. Bresler, *ADMIRA: Atomic Decomposition for Minimum Rank Approximation*, accepted for publication, IEEE Transactions on Information Theory, 2009.

[18] B. Recht and M. Fazel and P. A. Parrilo: *Guaranteed Minimum-Rank Solutions of Linear Equations via Nuclear Norm Minimization*, SIAM Review, vol. 52 (3), pp. 471–501, 2010.

[19] E. J. Candès and T. Tao, *The Power of Convex Relaxation: Near-Optimal Matrix Completion*, IEEE Transactions on Information Theory, vol. 56 (5), pp. 2053-2080, 2009.

[20] 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, vol. 23 (6), pp. 643-660, 2001.

Silvia Gandy
Department of Communications and Integrated Systems, Tokyo Institute of Technology, 152-8550 Tokyo, Japan.
E-mail: gandy(at)sp.ss.titech.ac.jp

Isao Yamada
Department of Communications and Integrated Systems, Tokyo Institute of Technology, 152-8550 Tokyo, Japan.
E-mail: isao(at)sp.ss.titech.ac.jp