ROBUST LOCALLY LINEAR ANALYSIS WITH APPLICATIONS TO IMAGE DENOISING AND BLIND INPAINTING

By

Yi Wang

Arthur Szlam

and

Gilad Lerman

IMA Preprint Series # 2379

(September 2011)

INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA
400 Lind Hall
207 Church Street S.E.
Minneapolis, Minnesota 55455–0436

Phone: 612/624-6066 Fax: 612/626-7370
URL: http://www.ima.umn.edu

Robust Locally Linear Analysis with Applications to Image Denoising and Blind Inpainting [*]

Yi Wang [§], Arthur Szlam [‡], and Gilad Lerman [†]

Abstract. We study the related problems of denoising images corrupted by impulsive noise and blind inpainting (i.e., inpainting when the deteriorated region is unknown). Our basic approach is to model the set of patches of pixels in an image as a union of low dimensional subspaces, corrupted by sparse but perhaps large magnitude noise. For this purpose, we develop a robust and iterative RANSAC like method for single subspace modeling and extend it to an iterative algorithm for modeling multiple subspaces. We prove convergence for both algorithms and carefully compare our methods with other recent ideas for such robust modeling. We demonstrate state of the art performance of our method for both imaging problems.

Key words. Denoising, impulsive noise, blind inpainting, alternating least squares, locally linear, robust PCA, multiple subspaces modeling, subspace clustering

AMS subject classifications. 62H35, 65D18, 68U10, 94A08, 68Q32

1. Introduction.

We consider the problem of denoising images corrupted with impulsive noise, and the problem of blind inpainting where the image has been "scratched". In both of these settings, some percentage of the pixels of the image have been grossly corrupted and the locations and the values of the corrupted pixels are unknown. In the former, it is assumed that the corruptions are unstructured (i.e., they are completely random), whereas the latter has geometrically structured corruptions (e.g., lying along curves). We take advantage of some recent progress in sparse coding, dictionary design and geometric data modeling in order to develop state-of-the-art performing methods for these two problems.

Sparse coding and dictionary design have been shown to be effective at denoising images corrupted with Gaussian noise as well as at standard inpainting with corruptions in known locations [11]. The basic approach is to fix a width $\sqrt{m}$, and extract all the $\sqrt{m} \times \sqrt{m}$ patches from the image, forming an $m \times n$ matrix $\mathbf{X}$, where n is the number of patches. Then $\mathbf{X}$ is decomposed into

$$\mathbf{DA} \simeq \mathbf{X},$$

where $\mathbf{D}$ is an $m \times k$ dictionary matrix, and $\mathbf{A}$ is the matrix of coefficients; $\mathbf{A}$ is either constrained to be sparse or encouraged to be sparse (via a regularization term). Using the sparsity assumption and some assumptions on the dictionary one can learn both $\mathbf{A}$ and $\mathbf{D}$ and create a new model for $\mathbf{X}$ that can be used for both standard denoising and inpainting.

Yu et al. [36] suggested a structured sparsity constraint. That is, they require $\mathbf{A}$ to have a block structure with a prespecified number of blocks; each block corresponds to a fixed

[*] This work was supported by NSF grants DMS-0811203, DMS-0915064 and DMS-09-56072, GL was also partially supported by the IMA during an annual program on simulating our complex world. We thank Guoshen Yu for carefully explaining details of [36]; Sathiya Keerthi for referring us to the initial technical report of [30] and providing some other references for alternating least squares; Joel Tropp for corresponding with YW regarding [30], Rene Vidal for some feedback on GL's short presentation of this work and some useful references and Teng Zhang for commenting on an earlier version of this manuscript.

[§] School of Mathematics, University of Minnesota, 127 Vincent Hall, 206 Church St. SE, Minneapolis, MN, 55455, wangx857@umn.edu

[‡] Courant Institute, New York University, 251 Mercer Street, New York, NY, 10012, aszlam@courant.nyu.edu

[†] School of Mathematics, University of Minnesota, 127 Vincent Hall, 206 Church St. SE, Minneapolis, MN, 55455, lerman@umn.edu

set of columns of $\mathbf{D}$. They learn this structure by partitioning $\mathbf{X}$ into clusters associated to subdictionaries of $\mathbf{D}$ via a variant of the K-subspaces algorithm [16, 4, 31, 15]. Given parameters r and K, this algorithm attempts to find K r-dimensional subspaces, represented by $m \times q$ (where $q < m$) orthogonal matrices $\mathbf{D}_1, ..., \mathbf{D}_K$ (i.e., matrices with orthonormal columns), minimizing

$$\sum_{j=1}^{N} \min_{1 \leq i \leq K} \|\mathbf{x}_j - \mathbf{D}_i \mathbf{D}_i^T \mathbf{x}_j\|^2. \tag{1.1}$$

Finding $\mathbf{D} = [\mathbf{D}_1, ...\mathbf{D}_K]$ can be thought of as a block structured dictionary learning problem, or as finding the best secant planes to the given data set.

The optimization for the K-subspaces model can be done via Lloyd iteration: holding the clusters fixed, find the best $\mathbf{D}$, and then update the clusters. With the clustering fixed, the problem reduces to finding the best rank r approximation to the columns in $\mathbf{X}$ associated to the cluster. In the case of Gaussian noise, "best" is usually chosen to be measured by Frobenius norm, in which case the solution is given via SVD.

In this paper we adapt the approach of [36] to images corrupted by impulsive noise (and perhaps with additional Gaussian noise) as well as to blind inpainting. In this framework, finding $\mathbf{D} = [\mathbf{D}_1, ...\mathbf{D}_K]$ with the clusters fixed requires a different tool than the SVD. In particular, we will need a "robust" low rank method, which allows large corruptions of a controlled percentage of the corresponding matrix. Recently, there has been a large interest in developing robust low rank models for matrices corrupted this way [8, 7, 27]. Our goal is to locally use such a low rank model in a framework like K-subspaces for image denoising and blind inpainting. However, we will find that these recent methods of low rank modeling are not well suited to this task, and will use instead a greedy and fast RANSAC like method for the updates of the dictionary $\mathbf{D}$.

The major contributions of this paper are as follows:

1. We describe a RANSAC like alternating least squares (ALS) algorithm for robust recovery of low rank matrices and compare it to other low-rank modeling algorithms (for the same corruption model).

2. We introduce local versions of the ALS (and other low-rank modeling algorithms) for block structured dictionary learning (or equivalently locally linear modeling) in the presence of large and sparse corruptions.

3. We prove that under some mild conditions the ALS algorithm and its local version (the K-ALS algorithm) converge to a fixed point with probability 1.

4. We show how the local methods can be used for image denoising with impulsive noise and blind inpainting. In addition to five common images, we use a database of 100 images to demonstrate the state of the art performance by our method.

2. Robust PCA Algorithms.

We discuss here algorithms for robust principal component analysis (robust PCA or RPCA) for recovering a given $m \times n$ matrix $\mathbf{X}$ when a percentage of the entries have been corrupted by noise. It is clear that in general, this is impossible: without some prior knowledge about the matrix, any value at any entry is as legitimate as any other value. However, it often happens in applications that $\mathbf{X}$ can be modeled as $\mathbf{X} = \mathbf{L} + \mathbf{S}$, where $\mathbf{L}$ is (approximately) low rank, and $\mathbf{S}$ is sparse.

While RPCA has a long history, most of the older work focused on the case where entire columns where corrupted; we will need to consider the case where any entry in the matrix may be corrupted. Recently this form of the problem has been intensely studied, starting with two parallel ground-breaking works by Candès et al. [7] and Chandrasekaran et al. [8]. They both proposed the following convex minimization for RPCA ([7] refers to it as principal component pursuit (PCP)):

$$\min_{\mathbf{L}\in\mathbb{R}^{m\times n}} \|\mathbf{L}\|_* + \lambda\|\mathbf{X}-\mathbf{L}\|_1, \tag{2.1}$$

where $||\mathbf{X}-\mathbf{L}||_1 := \sum_{i,j} |\mathbf{X}_{ij} - \mathbf{L}_{ij}|$, $||\mathbf{L}||_*$ is the sum of the singular values of $\mathbf{L}$ (i.e., its nuclear norm) and λ is a fixed parameter.

It was proved in [7] that if $\lambda = 1/\sqrt{n}$ and the data matrix can be represented as $\mathbf{X} = \mathbf{L}+\mathbf{S}$, where $\mathbf{L}$ is low-rank and "strongly incoherent" and $\mathbf{S}$ is sparse enough with uniformly sampled non-zero elements, then the minimizer of (2.1) is $\mathbf{L}$ with overwhelming probability (of the uniform sampling). On the other hand, [8] provided a condition for deterministic exact recovery for various distributions of corrupted elements, though it implied a stronger restriction on the fraction of corruption; clearly [8] could not fix λ for its more general setting.

The minimization of (2.1) can be done via an augmented Lagrangian approach, which alternates between space shrinkage and singular value shrinkage, see e.g., [19]. This procedure can be made very efficient when one has a good guess at an upper bound for the rank of the minimizer, which we refer as PCP(capped) and implement in the supplemental material. We remark though that [19] regularizes the minimization of (2.1) as follows:

$$\min_{\mathbf{L},\mathbf{S}\in\mathbb{R}^{m\times n}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \nu\|\mathbf{X}-\mathbf{L}-\mathbf{S}\|_F^2, \tag{2.2}$$

where ν is a sufficiently small parameter.

Another approach for RPCA, LMaFit [27], uses only the last two terms of (2.2). That is, it minimizes the objective function

$$\min_{\mathbf{B},\mathbf{C},\mathbf{S}} \|\mathbf{S}+\mathbf{BC}-\mathbf{X}\|_F^2 + \mu\|\mathbf{S}\|_1 \tag{2.3}$$

where $\mathbf{S}\in\mathbb{R}^{m\times n}$, $\mathbf{B}\in\mathbb{R}^{m\times d}$ and $\mathbf{C}\in\mathbb{R}^{d\times n}$. The LMaFit introduces narrow matrices $\mathbf{B}$ and $\mathbf{C}$ to control the rank of the approximation and allows distortion by adding the Frobenius norm term to the objective function. Thus the LMaFit model is expected to handle well Gaussian noise.

We remark that Favaro et al. [12] suggested to directly minimize (2.2), where λ and ν need to be carefully selected. No theoretical guarantees are provided for this generalization of PCP and their code was not available to us.

2.1. RPCA via an Alternating Least Squares Algorithm.

The simplest interpretation of the decomposition of $\mathbf{X}$ into sparse $\mathbf{S}$ and low rank $\mathbf{L}$ is to fix the number of nonzero elements N_0 in $\mathbf{S}$ (or its percentage p, so that $N_0 = \lfloor p\cdot m\cdot n\rfloor$) and to fix a rank d for $\mathbf{L}$. In this situation, we might pay for errors with some matrix norm. If we choose the Frobenius norm, we obtain the problem

$$\min_{\substack{\mathbf{B}\in\mathbb{R}^{m\times d},\,\mathbf{C}\in\mathbb{R}^{d\times n}\\ \mathcal{I}\subseteq\{1,\ldots,m\}\times\{1,\ldots,n\}\,:\,|\mathcal{I}|=N_0}} J(\mathbf{B},\mathbf{C},\mathcal{I}), \tag{2.4}$$

where

$$J(\mathbf{B},\mathbf{C},\mathcal{I}) = \sum_{\substack{i\in\{1,\ldots,m\}\\ j\in\{1,\ldots,n\}\\ (i,j)\notin\mathcal{I}}} |(\mathbf{BC})_{ij} - \mathbf{X}_{ij}|^2. \tag{2.5}$$

The low rank approximation is $\mathbf{L} = \mathbf{B}\,\mathbf{C}$ and the sparse matrix $\mathbf{S} = \mathbf{X} - \mathbf{L}$ is supported on the set of indices $\mathcal{I}$. This formulation suggests a simple algorithm, which we describe in Algorithm 1 and will henceforth refer to as ALS. In order to obtain a well-conditioned optimization problem and thus stable solutions, we regularize the objective function as follows:

$$J(\mathbf{B}, \mathbf{C}, \mathcal{I}) = \sum_{\substack{i \in \{1,\dots,m\} \\ j \in \{1,\dots,n\} \\ (i,j) \notin \mathcal{I}}} |(\mathbf{BC})_{ij} - \mathbf{X}_{ij}|^2 + \lambda_1 \|\mathbf{B}\|_F^2 + \lambda_2 \|\mathbf{C}\|_F^2. \tag{2.6}$$

where λ_1 and λ_2 are some small positive constants.

In practice we use the regularized ALS algorithm and refer to it as the ALS algorithm. Throughout all the experiments we fix $\lambda_1 = \lambda_2 = 10^{-10}$ and $n_1 = 1$. For the simulated data n_2 is chosen such that either the algorithm converges or n_2 obtains a sufficiently large value. More precisely, the outer loop stops if one of the following criterions is achieved: 1) $n_2 = 100$; 2) the relative change of J is less than 10^{-3}; 3) the absolute change of J is less than 10^{-4}. For the image data (§5) ALS is applied within K-ALS and it was enough to require $n_2 = 1$ and consequently speed up the algorithm. We note that the other parameters, p and d, have a clear interpretation in terms of the data and one may expect a bound on them for various data sets. As in many RANSAC-like algorithms, the results of ALS are robust to overestimation of p and d, but not robust to underestimation (see §A). In §B, we propose and test an estimation strategy for d.

Algorithm 1: An ALS algorithm of recovering a low rank matrix from corrupted data

```
Input: $\mathbf{X} \in \mathbb{R}^{m \times n}$, $d$: low rank, $p$: the percentage of corrupted entries, $n_1, n_2$: numbers
of iteration, $\lambda_1, \lambda_2 \geq 0$: regularization parameters.
Output: $\mathbf{B} \in \mathbb{R}^{m \times d}$, $\mathbf{C} \in \mathbb{R}^{d \times n}$, $\mathcal{I} \subseteq \{1,\dots,m\} \times \{1,\dots,n\} \ : \ |I| = \lfloor p \cdot m \cdot n \rfloor$.
Initialization: $\mathcal{I} = \emptyset$, $\mathbf{B} \in \mathbb{R}^{m \times d}$ with i.i.d. entries: $\mathbf{B}_{ij} \sim \mathcal{N}(0,1)$.
for $l = 1 : n_2$ do
  for $t = 1 : n_1$ do
    $\mathbf{C} = \arg\min_{\mathbf{C}} J$ ($J$ is defined in (2.6)).
    $\mathbf{B} = \arg\min_{\mathbf{B}} J$.
  end for
  $\mathcal{I}$ = indices of the $\lfloor p \cdot m \cdot n \rfloor$ greatest elements of $\{|\mathbf{X}_{ij} - (\mathbf{BC})_{ij}|^2\}_{i=1}^{m}{}_{j=1}^{n}$.
end for
```

ALS algorithms for PCA have long history and have been widely used for matrix completion, i.e., when the locations of the corrupted elements are specified, see for example [35, 29, 32, 6, 25, 24, 38]. In fact, the ALS algorithm for matrix completion is practically obtained by fixing the outlier set $\mathcal{I}$ and performing a restricted version of the ALS algorithm of this paper. In our more general setting, the ALS iterations for finding $\mathbf{B}$, $\mathbf{C}$ and $\mathcal{I}$ can be thought of as a version of RANSAC [14] (though we do not randomly initialize the inliers each time, but we iterate between the choice of inliers and the low-rank model with a single random initialization for each of them).

2.2. Why a New RPCA Algorithm?.

The reader may wonder why we use an alternating least squares method rather than the convex method of [8, 7], which has theoretical guarantees. In fact, we use RPCA in a K-subspaces framework for modeling the underlying data by multiple subspaces. One cannot expect a fully convex formulation for such setting. Indeed, Lerman and Zhang [18, §5.1] have shown that it is impossible to generalize convex strategies as in [8, 7] for recovering multiple subspaces by energy minimization. Moreover, strategies like [21, 20], which generalize [8, 7] to obtain an affinity matrix via a convex procedure (and then apply the non-convex spectral clustering), are extremely slow for our application (as we exemplify in §5). Furthermore, it does not make much sense to apply [8, 7] within a K-subspaces algorithm because it is unclear how to extend the nuclear norm out-of-sample (see §3.2). In particular, we cannot prove weak convergence theorems for the naive K-subspaces versions of PCP, unlike ALS (see §4.4).

A second concern with many of the above methods is sensitivity to non-sparse corruptions (such as additive Gaussian noise). While they often can perform better than ALS with only sparse corruptions, the exact recovery results do not extend to this more general situation. In practice, we have found that many of these methods are brittle with additive white Gaussian noise.

Finally, and in some sense, most importantly, in our application, we have a good initialization. One of the complaints with non-convex methods like K-means or K-subspaces is the dependence on the initialization for good performance. But the initialization in [36] has shown to be effective for image denoising. In our setting, because we have such a good idea where to start, dependence on initialization is actually a feature, and not a bug.

2.3. Numerical Simulations with a Single Subspace. To evaluate the performance of ALS in comparison to other RPCA methods, several simulations are conducted. We distinguish here and thereafter between p_0, the ground truth percentage of sparse corruptions (i.e., impulsive noise), and p, the input parameter of ALS estimating the percentage of corruption; d_0, the ground truth rank of the clean matrix, and d, the input rank of the matrices $\mathbf{B}$ and $\mathbf{C}$ for the ALS algorithm.

We generate a low rank matrix $\mathbf{X}_0$ as a product of two independent $m \times d_0$ and $d_0 \times n$ matrices whose entries are i.i.d. $\mathcal{N}(0,1)$ random variables and scale it to have unit spectral norm. We form $\mathbf{X}$ by independently adding to each element of $\mathbf{X}_0$ Gaussian noise with standard deviation σ (where $\sigma = 0.05$ or $\sigma = 0$ when no Gaussian noise). Finally, p_0 randomly chosen pixels of $\mathbf{X}$ ($p_0 = 5\%, 10\%, 20\%$) are assigned uniform random values in the interval $[-a, a]$, where $a = \max_{i,j} |\mathbf{X}_{0ij}|$.

We test the following RPCA algorithms on this data: PCP, PCP(capped), LMaFit and ALS. For PCP, we use the fast implementation suggested in [19] with $\lambda = (mn)^{-0.25}$. For ALS, PCP(capped) and LMaFit(oracle) we input the intrinsic rank to be $d_0 + 3$ (since they all can handle an overestimation for this parameter; see §A). We recall that PCP(capped) is the version of PCP mentioned in §2, which requires an upper bound, i.e., a cap, on the intrinsic dimension (we defer the full implementation details to the supplemental code). For PCP(capped) we also input two other parameters: λ of (2.1) and γ, which is present in every ADMM (alternating direction method of multipliers) implementation for PCP (see supplemental code). We tested $(\lambda, \gamma) = (10^k, 10^l)$, for $k, l = -3, \ldots, 3$ and selected $(\lambda, \gamma) = (0.1, 10)$ based on the true fitting error (defined below); therefore our implementation of PCP(capped) is of oracle type. For LMaFit, we chose $\mu = 10^3$ by minimizing the true fitting error among $\mu = 10^k$, $k = -3, \ldots, 5$. We thus refer to the method as LMaFit(oracle). The parameters for ALS were specified in §2.2.

For all of the experiments $n = 1000$. For any RPCA algorithm, we quantify the recovery of the underlying low rank model by the following relative fitting error:

$$e = \frac{\|\tilde{\mathbf{X}} - \mathbf{X}_0\|_F}{\|\mathbf{X}_0\|_F}, \tag{2.7}$$

where $\tilde{\mathbf{X}}$ is the low rank estimation for X. We report the mean of this error for different values of m , d_0 and p_0 in Tables 2.1 and 2.2 when $\sigma = 0$ and $\sigma = 0.05$ respectively. Note that in those tables, the error is shown in percentage (%), ALS(p_0) inputs p_0 as the estimate of portion of corruptions, i.e., $p = p_0$, while ALS($2p_0$) has $p = 2p_0$.

Table 2.1: Relative fitting error on simulations without Gaussian noise

m	100			200			400			800		
d_0	5			10			20			40		
p_0	5%	10%	20%	5%	10%	20%	5%	10%	20%	5%	10%	20%
ALS($2p_0$)	1.10	6.02	20.95	0.49	2.58	10.71	0.08	0.49	5.48	0.04	0.06	1.19
ALS(p_0)	0.15	0.54	2.93	0.02	0.05	0.66	0.00	0.01	0.11	0.00	0.01	0.02
PCP(capped)	0.43	6.56	19.86	0.44	3.69	10.97	0.61	2.51	5.93	0.90	2.60	5.35
PCP	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
LMaFit(oracle)	0.95	1.43	3.77	1.01	1.54	2.47	1.13	1.71	2.80	1.36	2.05	3.34

Table 2.2: Relative fitting error on simulations with Gaussian noise ($\sigma = 0.05$)

m	100			200			400			800		
d_0	5			10			20			40		
p_0	5%	10%	20%	5%	10%	20%	5%	10%	20%	5%	10%	20%
ALS($2p_0$)	7.53	9.21	21.60	6.60	7.82	12.70	6.06	7.13	10.32	5.87	6.91	10.00
ALS(p_0)	6.65	7.04	8.63	5.80	6.13	6.84	5.28	5.56	6.19	5.09	5.36	5.97
PCP(capped)	14.27	15.31	20.56	11.89	12.74	14.91	10.03	10.74	12.43	8.74	9.41	10.95
PCP	20.80	24.37	41.59	15.86	18.97	26.25	12.40	13.34	17.61	10.28	11.50	14.43
LMaFit(oracle)	5.38	5.84	6.95	5.23	5.69	6.79	5.11	5.56	6.69	5.14	5.65	6.90

We observe that PCP performs the best for clean data, but it could not handle well Gaussian noise. ALS works better with higher ambient dimensions, since in this case there are more equations for the least squares solution (with the same number of variables) and thus it increases the accuracy. ALS is also more accurate when the portion of corruptions decreases. We note that ALS(p_0) performs better than LMaFit(oracle) for clean data and they are comparable for noisy data. As for ALS($2p_0$), it performs better than PCP(capped) in general. When the ambient dimension is high or the corruption level is low then it performs better than LMaFit(oracle) for clean data and comparably to it for noisy data, whereas for the rest of the cases it is not as good as LMaFit(oracle).

Table 2.3 reports the speed of these algorithms (with $\sigma = 0.05$; the running times are comparable when $\sigma = 0$). The experiments were performed using a Dual processor Intel Core 2 CPU at 2.66GHz with 2 GB of memory. We observe that PCP(capped) is the fastest method (though we did not take into account the repetitions for the choices of the oracle parameters) and ALS is right after it. LMaFit is also comparable to these two in terms of speed (again repetitions for best parameters were not taken into account). On the other hand, PCP is extremely slow.

Table 2.3: The running time (in s) of the simulations with Gaussian noise ($\sigma = 0.05$)

m	100			200			400			800		
d_0	5			10			20			40		
p_0	5%	10%	20%	5%	10%	20%	5%	10%	20%	5%	10%	20%
ALS($2p_0$)	0.18	0.17	0.18	0.48	0.50	0.48	0.84	0.81	0.80	2.14	2.10	2.04
ALS(p_0)	0.18	0.18	0.18	0.49	0.49	0.50	0.84	0.83	0.82	2.16	2.15	2.10
PCP(capped)	0.06	0.06	0.06	0.18	0.18	0.19	0.34	0.32	0.33	0.78	0.78	0.78
PCP	4.83	4.81	5.21	16.54	16.42	16.15	89.94	86.61	83.94	581.77	569.24	555.28
LMaFit(oracle)	0.26	0.31	0.46	0.56	0.66	0.93	0.93	1.10	1.54	1.79	2.17	3.11

3. Piecewise Linear Models. We introduce local versions of the ALS (and other low-rank modeling algorithms) for block structured dictionary learning. Alternatively, we introduce local linear modeling of data with low-dimensional structure in the presence of large and sparse corruption. Our elementwise matrix corruption model completely distorts the nearest neighbors' distances between the uncorrupted data points (stored as matrix columns). Therefore, standard methods for manifold learning [28, 26, 17] will completely fail on such corrupted data. We thus model the data by several multiple subspaces via a K-subspaces strategy, which replaces the least squares best fit subspaces by different robust low-rank best fit subspaces.

3.1. The K-ALS Algorithm.

The generalization of ALS to the setting of K-subspaces is straightforward. We fix a number of subspaces, K, a rank, d (though mixed ranks for the different subspaces are also possible), and a percentage of corrupted entries, p. We initialize a list of subspaces (details of the initialization are discussed in §5) and assume no corrupted elements in the initial iteration. The algorithm then iteratively repeats the following two stages. The first step finds clusters of "nearest" data points to the K subspaces. To formulate it precisely, we associate to each of the ith d-dimensional subspaces ($1 \le i \le K$) a basis, which we represent as the columns of an $m \times d$ matrix $\mathbf{B}_i$. Given a data point $\mathbf{x} \in \mathbb{R}^m$ (i.e., column of the matrix $\mathbf{X}$), we denote by $\overline{\mathbf{x}}$ the uncorrupted entries in $\mathbf{x}$ (i.e., uncorrupted elements in the corresponding column of $\mathbf{X}$) and by $\overline{\mathbf{B}}_i$ the $\dim(\overline{\mathbf{x}}) \times d$ matrix obtained by the rows of $\mathbf{B}_i$ corresponding to the same uncorrupted indices of $\mathbf{x}$. Using this notation, $\mathbf{x}$ is nearest to the subspace indexed by

$$j = \arg\min_{1 \le i \le K} \min_{\mathbf{c} \in \mathbb{R}^d} \|\overline{\mathbf{B}}_i \mathbf{c} - \overline{\mathbf{x}}\|_2^2. \tag{3.1}$$

The second step computes a subspace for each cluster of the first step by the ALS procedure of §2.

3.2. Other Possibilities of K-RPCA Algorithms.

While in some sense, any algorithm for robust PCA can be extended to a K-RPCA algorithm, the generalization of the PCP model ($\|\cdot\|_* + \lambda \cdot \|\cdot\|_1$) to the setting of K-subspaces is less straightforward. Indeed, the inclusion of new data points (i.e., the first step above) affects the nuclear norm of the whole data set and may require changing the representations of other points. Nevertheless, to run the local version of the algorithm, if $\mathbf{x} \in \mathbb{R}^m$ is a data point and $\mathbf{B}_1, ..., \mathbf{B}_K$ are the orthonormal bases of the d-dimensional subspaces representing the clusters, then we associate to $\mathbf{x}$ the subspace $\mathbf{B}_j$, where

$$j = \arg\min_{1 \le i \le K} \min_{\mathbf{c} \in \mathbb{R}^d} \|\mathbf{x} - \mathbf{B}_i \mathbf{c}\|_1. \tag{3.2}$$

On the other hand, the LMaFit model extends to a K-LMaFit model without complications. If $\mathbf{x} \in \mathbb{R}^m$ is a data point and $\mathbf{B}_1, ..., \mathbf{B}_K$ are the orthonormal bases of the d-dimensional subspaces representing the clusters, then we associate to $\mathbf{x}$ the subspace $\mathbf{B}_j$, where

$$j = \arg\min_{1 \le i \le K} \min_{\mathbf{c} \in \mathbb{R}^d, \mathbf{e} \in \mathbb{R}^m} \|\mathbf{e} + \mathbf{B}_i \mathbf{c} - \mathbf{x}\|_F^2 + \mu \|\mathbf{e}\|_1. \tag{3.3}$$

3.3. Some Implementation Details.

All the piecewise linear models discussed above have hard decision boundaries for cluster membership, therefore, points near the boundary may not be well represented. One simple remedy to this kind of problem is to repeat the clustering several times and generate multiple reconstructions via overlapping clusterings. In the wavelet literature, this technique is called cycle spinning [9]; in this context, it is simply the idea that the charts of a manifold should overlap. If the final goal is denoising, then we average the outputs of all repetitions.

In the K-PCP model, when computing the nearest low rank model for a data point, we need to cap the rank allowed for each of the models, else the model with the highest rank will become the representative for every point.

4. Mathematical Analysis of Convergence.

We establish theoretical analysis for slightly modified versions (via additional regularization) of the ALS and K-ALS algorithms. Our numerical experiments indicated similar performance of these modified versions and the original versions presented earlier. Therefore, we use the modified versions only for theoretical analysis and the simpler algorithms in the rest of the paper.

4.1. Preliminaries.

We denote the Hadamard product by $\circ$, i.e., $(\mathbf{A} \circ \mathbf{B})_{ij} = \mathbf{A}_{ij}\mathbf{B}_{ij}$. We let $\mathbf{A}_{\cdot j}$ and $\mathbf{A}_{i\cdot}$ denote the j-th column and i-th row respectively of the matrix $\mathbf{A}$. We represent the set $\mathcal{I}$ by the matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ such that $\mathbf{W}_{ij} = 0$, if $(i, j) \in \mathcal{I}$; and $\mathbf{W}_{ij} = 1$ otherwise.

We modify the objective function J for ALS by adding another regularization term (depending on a sufficiently small $\lambda_3 > 0$ and $\mathbf{U} \in \mathbb{R}^{m\times n}$ whose elements are i.i.d. samples from a continuous distribution on $[0, 1]$) and express J using the above notation in the following way:

$$J(\mathbf{B}, \mathbf{C}, \mathbf{W}) = \|(\mathbf{BC} - \mathbf{X}) \circ \mathbf{W}\|_F^2 + \lambda_1\|\mathbf{B}\|_F^2 + \lambda_2\|\mathbf{C}\|_F^2 + \lambda_3\|\mathbf{U} \circ \mathbf{W}\|_F^2. \tag{4.1}$$

We also extend this objective function for K-ALS (with further regularization) as follows:

$$\begin{aligned} &J\left(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathbf{W}_k\}_{k=1}^K, \eta\right) \qquad (4.2)\\ &= \sum_{k=1}^K (\|(\mathbf{B}_k\mathbf{C}_k - \mathbf{X}_k) \circ \mathbf{W}_k\|_F^2 + \lambda_1\|\mathbf{B}_k\|_F^2 + \lambda_2\|\mathbf{C}_k\|_F^2) + \lambda_3\|\mathbf{U}_k \circ \mathbf{W}_k\|_F^2 + \lambda_4 \sum_{j=1}^n \mathbf{V}_{\eta(j),j}^2, \end{aligned}$$

where η maps the indices of data points $(1, \ldots, n)$ to indices of subspaces $(1, \ldots, K)$, $\mathbf{X}_k \in \mathbb{R}^{m\times n_k}$ is a submatrix of $\mathbf{X}$ representing the n_k data points in cluster $\eta(j) = k$, $\mathbf{W}_k \in \{0, 1\}^{m\times n_k}$, $\mathbf{U}_k \in \mathbb{R}^{m\times n_k}$, $\mathbf{V} \in \mathbb{R}^{K\times n}$ (both $\{\mathbf{U}_k\}_{k=1}^K$ and $\mathbf{V}$ are initialized by i.i.d. samples from a continuous distribution on $[0, 1]$), $\mathbf{V}_{\eta(j),j}$ is the element at the $\eta(j)$-th row and j-th column and $\lambda_1, \ldots, \lambda_4$ are arbitrarily small and positive regularization parameters.

Theorems 4.1 and 4.2 below use the technical notion of a fixed point of an algorithm. We delay its definition as well as the proof of these theorems to §D.

4.2. Theorem for the Convergence of ALS.

We establish the following result for convergence of the regularized ALS algorithm for robust PCA.

Theorem 4.1.*For the regularized ALS algorithm described above, the following statements hold with probability one: 1) All accumulation points of the iterates* $(\mathbf{B}^t, \mathbf{C}^t, \mathbf{W}^t)$ *produced by this algorithm are its fixed points; 2)* $J(\mathbf{B}^t, \mathbf{C}^t, \mathbf{W}^t) \to J(\mathbf{B}^*, \mathbf{C}^*, \mathbf{W}^*)$*, where* $(\mathbf{B}^*, \mathbf{C}^*, \mathbf{W}^*)$ *is a fixed point; 3)* $\|(\mathbf{B}^{t+1}, \mathbf{C}^{t+1}, \mathbf{W}^{t+1}) - (\mathbf{B}^t, \mathbf{C}^t, \mathbf{W}^t)\| \to 0$*; and 4) either* $\{(\mathbf{B}^t, \mathbf{C}^t, \mathbf{W}^t)\}$ *converges or the accumulation points form a continuum.*

4.3. Main Theorem: Convergence of K-ALS.

We establish the following result for convergence of the regularized K-ALS algorithm for robust PCA. For simplicity, we denote $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathbf{W}_k\}_{k=1}^K, \eta)$ by Ω.

Theorem 4.2.*For the regularized K-ALS algorithm described above, the following statements hold with probability one: 1) All accumulation points of the iterates* Ω^t *produced by this algorithm are its fixed points; 2)* $J(\Omega^t) \to J(\Omega^*)$*, where* Ω^* *is a fixed point; 3)* $\|\Omega^t - \Omega^{t+1}\| \to 0$*; and 4) either* Ω *converges or the accumulation points form a continuum.*

4.4. Discussion on Statements of Theorems.

The purpose of our theory is to guarantee convergence of the ALS and K-ALS algorithms. While the convergence is not fully guaranteed (when the accumulation points form a continuum), we note that also the theoretical guarantees for the agglomerative Lagrangian for PCP [19] are only formulated in terms of convergence of the objective function as we have in the second part of our theoretical statements (but they do not have results for convergence in norm).

The type of convergence we have for K-ALS is slightly weaker than the one of K-means or K-subspaces. For these algorithms one can prove convergence to a local minimum [3], whereas here we only prove convergence to a fixed point. The difference is that for K-means and K-subspaces, the best fit center or subspace is easily determined for each cluster, whereas the robust ALS only guarantees convergence for each cluster. It is interesting to note that while K-PCP easily determines a subspace for each fixed cluster, we cannot guarantee convergence for the full algorithm (as in K-ALS). Indeed, K-PCP minimizes $||\mathbf{L}||_* + \lambda||\mathbf{X} - \mathbf{L}||_1$ within each cluster, but for the partition it minimizes $||\mathbf{X} - \mathbf{L}||_1$ among clusters (due to the difficulty of extending the nuclear norm out of sample). Therefore the resulting objective function for K-PCP is not even guaranteed to be monotonic.

5. Restoration of Corrupted Images.

5.1. Problem Setting. We test different methods of local (and global) low rank modeling for denoising images with impulsive noise as well as blind inpainting. The data consists of 5 popular images and also the 100 test images of the Berkeley segmentation database [22]. We use two noise parameters p_0 and σ to specify the corruptions to the images. The images are corrupted with i.i.d. additive Gaussian noise with standard deviation σ, and a percentage of p_0 random pixels per image (uniformly selected) are corrupted with an integer uniformly sampled between 0 and 255. For inpainting, the images are further degraded by drawing a scratch. For many of the methods (in particular, K-ALS) the images are represented by their 8×8 patches. That is, the actual data matrix $\mathbf{X}$ is formed by stacking the vectors representing these patches as columns. Its size is thus $64 \times n$, where n is the number of pixels in the image. This matrix is transformed back to the image (after enhancing it) by finding the average value of all coordinates representing the same pixel.

5.2. Methods, Implementation Details and Parameters. We compared K-ALS with the following methods (sometimes combined with each other, though we reported only methods that were sufficiently fast and accurate): Median filter (MF), iterated median filter (IMF), structured sparse model selection (SSMS) [36], K-SVD [11, 1], low rank representation (LRR) [21, 20], PCP, PCP(capped), LMaFit, K-PCP, K-PCP(capped), K-LMaFit and three variations of the non-local means method of [5] (described in §5.3).

We explain here various implementation issues of the different algorithms and the choice of parameters. For IMF, we chose among 10 iterations the one with highest PSNR, thus giving an oracle advantage to this method. For local methods based on the K-subspaces strategy (i.e., SSMS, K-ALS, K-PCP, K-LMaFit), we followed [36] to learn $K = 19$ subspaces of dimension $d = 8$ with 18 bases for subspaces initialized on artificial edges at given orientations and one low frequency DCT basis (our implementation is available online). Note that this initialization is important for good results.

In order to account for Gaussian noise, we apply additional steps in the spirit of [36] (in order to be consistent for all methods). We first recall that [36] uses the following thresholding step for their estimator for $\mathbf{X}$, $\tilde{\mathbf{X}}$, in order to remove additional Gaussian noise. The modified estimator $\hat{\mathbf{X}}$ is obtained by replacing each column $\tilde{\mathbf{x}}$ of $\tilde{\mathbf{X}}$ by the following column:

$$\hat{\mathbf{x}} = \mathbf{B}\delta(\mathbf{B}^T\tilde{\mathbf{x}}), \tag{5.1}$$

where $\mathbf{B} \in \mathbb{R}^{m \times d'}$ is formed by stacking as columns the first d' components of the basis of the cluster containing $\tilde{\mathbf{x}}$ (they use the full dimension $d' = 64$) and $\delta(\mathbf{a}_j) = \mathbf{a}_j$ if $|\mathbf{a}_j| > 3\sigma$;

and $= 0$ otherwise (i.e., SSMS assumes knowledge of the model parameter σ described in §5.1).

For K-ALS and K-PCP we applied a similar thresholding procedure within each iteration with $d' = 20$. We chose this parameter since the SVD of the clusters of natural image patches can be well approximated with rank 15 and slight overestimation cannot effect the results.

Before thresholding, K-ALS applies the following procedure within each iteration to re-estimate the locations of corruption in the corresponding image: let $\mathbf{Y}$ denote the original image, $\text{med}(\mathbf{Y})$ the image obtained by applying 4×4 median filter to $\mathbf{Y}$, $\tilde{\mathbf{Y}}$ the image translated from the patches of the current iteration (before thresholding) and $\text{abs}(\cdot)$ the elementwise absolute value. Furthermore, let $r = \text{abs}(\mathbf{Y} - \tilde{\mathbf{Y}})$ and $s = \text{abs}(\mathbf{Y} - \text{med}(\mathbf{Y}))$. For denoising, we identify in each iteration the corrupted pixels of a fixed percentage (p) as the ones with the largest entries in r; for inpainting, we do the same with $r + s$ at the first iteration and with r thereafter (the purpose of the additional s for inpainting is to improve scratch detection, however, for denoising such a process can falsely detect edges as corruption).

For the global low rank model (in particular, PCP), we address Gaussian noise by further applying the SSMS algorithm [36] (so the method is comparable to the other strategies). Numerical experiments indicated that applying SSMS after the global low rank model was slightly better than applying the thresholding described above and clearly better than PCP alone.

In terms of parameters, for K-ALS, $\lambda_1 = \lambda_2 = 10^{-10}$, $n_1 = n_2 = 1$ and $n_3 = 5$ (where n_3 is the number of iterations between ALS and the subspace clustering, i.e., the number of iterations for the very outer loop). For p, we tested both $p = p_0$ and $p = 2p_0$. It is important to note that we used the same fixed parameters for all 105 test images. We capped both PCP(capped) and K-PCP(capped) at 20. For PCP and K-PCP, the parameter λ was chosen to be 0.01 after testing the largest PSNR obtained among $\lambda = 10^k$, $k = -3, \ldots, 3$. Similarly, $(0.1, 0.001)$ was chosen for (λ, γ) after testing $(10^k, 10^l)$, $k, l = -3, \ldots, 3$ for PCP(capped)+SSMS and K-PCP(capped). For LMaFit, K-LMaFit and LRR [21, 20], we fine-tuned the best parameters based on PSNRs of the outputs. We first tested 7 values of 10^k, $k = -3, \ldots, 3$ and then searched around the best one in a finer scale until the results were not improved.

5.3. An Alternative Approach: Non-Local Medians.

Many of the best performing image denoising methods (in the additive Gaussian noise setting) are versions of the non-local means algorithm (NLM) [5]. A standard version takes in parameters m, ϵ, and l, and defines a weight matrix $\mathbf{W}$ with elements

$$\mathbf{W}_{ij} = h(i)\, e^{-d(\mathbf{x}_i, \mathbf{x}_j)^2/\epsilon^2},$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are patches of width $\sqrt{m}$ about pixels i and j respectively (represented as vectors in $\mathbb{R}^m$), h is 1 in a square neighborhood centered at i of sidelength l pixels and 0 elsewhere, and d denotes the Euclidean distance between the patches considered as vectors. A normalized weight $\tilde{\mathbf{W}}$ is formed by $\tilde{\mathbf{W}} = \mathbf{D}^{-1}(\mathbf{W} + \mathbf{W}^T)$, where $\mathbf{D}$ is the diagonal matrix with the row sums of $\mathbf{W} + \mathbf{W}^T$ on its diagonal. The noisy image is then smoothed by multiplying it with $\tilde{\mathbf{W}}$. The standard versions of NLM cannot handle impulsive noise or scratches, because the patch distance is too sensitive to outliers or gross corruptions.

We propose and test three robust versions of non-local means, which we refer to as the non-local medians. First, we define three different similarity functions between patches $\mathbf{x}_j$ and $\mathbf{x}_i$ (of width $\sqrt{m}$) as follows: 1) d_q is defined as the q-th power of the l_q (quasi)-norm, where $q \leq 1$; 2) $\tilde{d}_r(\mathbf{x}_i - \mathbf{x}_j) := \sum_{k=1}^{m} \mu_k(\mathbf{x}_i(k) - \mathbf{x}_j(k))^2$, where $\mu_k = 1$ for the r coordinates of $\mathbf{x}_i - \mathbf{x}_j$ with smallest absolute difference $|\mathbf{x}_i - \mathbf{x}_j|$ and $\mu_k = 0$ on the other $m - r$ coordinates; and 3) apply first a median filter to the entire image, and then use the regular Euclidean distance d_2. To find the non-local medians, we define a square neighborhood of length l for the i-th pixel and denote the indices of the pixels in this neighborhood by S_i. We compute the distances (d_q, $\tilde{d}_r$, or d_2 after median filtering) between $\mathbf{x}_i$ and $\mathbf{x}_j$ for all $j \in S_i$. Then we form $\tilde{S}_i$ by keeping only s patches corresponding to the smallest distances. Finally, an estimate for the intensity value at pixel i is made by taking the median value among all intensities of pixels of patches in $\tilde{S}_i$.

In our experiments testing these three non-local median methods, we let $q = 0.5$, $m = 16$, $s = 12$, $l = 7$ and $r = \lfloor (1 - 2p_0)m \rfloor$. We remark that the algorithm is not sensitive to the choice of l as long as it is sufficiently large (but the algorithm slows as l gets larger). The parameter r was set to be adaptive to p_0. The other parameters were chosen by fine tuning numerical results.

We noticed that the application of SSMS after any of these methods does not in general help with reducing Gaussian noise. In fact, it tended to make things worse for small (or no) Gaussian noise and improved results for larger amounts, e.g., $\sigma \geq 20$ (though, this improvement with known σ and p_0 was still not better than K-ALS(p_0)).

5.4. Results. For each image, method and choice of corruption/denoising parameters, we averaged the PSNR over five simulations. Tables 5.1 and 5.2 report results of the different methods for denoising impulsive noise and blind inpainting respectively for the five common images. Figures 5.2 and 5.3 plot the PSNR of the database of 100 images in several scenarios for denoising and inpainting respectively. The image IDs are sorted according to increasing PSNR of K-ALS($2p_0$) for better illustration.

The tables and figures omit some methods for the following reasons. The PCP algorithm is extremely slow (as in Table 2.3), we thus run PCP+SSMS and K-PCP only for these five common images. We reported the best of the 3 non-local medians methods. Since MF+SSMS is similar to but better than MF+KSVD and since SSMS and K-SVD are designed only for Gaussian noise and thus not competitive at all, we only reported MF+SSMS among these. The PSNR of K-LMaFit was in the order of $25db$ and we thus did not report it in the tables. The reason for this poor performance is not clear to us since the distributed software of LMaFit is not an open source; however, we suspect their initialization may be to blame. For LRR, it is not feasible to assign the data matrix itself as the dictionary (as done in [21, 20]). We thus tried the following two options: the first is to use the initial basis used for K-ALS and the second is to use the output basis learned by K-ALS. We observed that both options could achieve good results by carefully choosing the regularization parameter. However, they are very sensitive to the choice of the regularization parameter. In fact, they needed to be accurate up to the third decimal place to obtain comparable results as K-ALS($2p_0$) and those best values can be different for different images. Also, LRR is slow; for example, it takes about a minute for K-ALS to finish the *House* image and several hours for LRR.

In addition to PSNR, we also evaluated the image quality by the Structural Similarity (SSIM) index [34]. We report these results in §C. We observe that the SSIM index favors K-ALS more than PSNR. The performance of PCP(capped)+SSMS in terms of SSIM oscillates much more, while it stays close to a certain level in terms of PSNR. Moreover, we see that the performance of K-ALS(p_0) is more correlated with K-ALS($2p_0$) in SSIM than in PSNR.

Besides PSNR and SSIM, we also demonstrate the visual quality of some denoising and blind inpainting results in Figures 5.4 and 5.5 respectively. The visual quality is important.

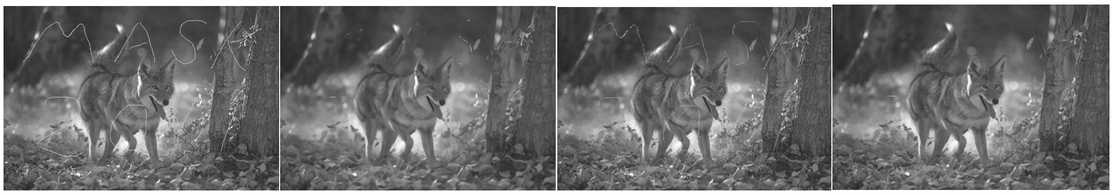


Figure 5.1: Demonstration of the importance of visual quality. From left to right: corrupted image ($\sigma = 0$, $p_0 = 0$), NLM3 (PSNR=29.51, SSIM=0.8637), PCP(capped)+SSMS (PSNR=32.66, SSIM=0.9425) and ALS($2p_0$) (PSNR=31.70, SSIM=0.9425)

Indeed, Figure 5.1 shows an image where PCP(capped)+SSMS obtains high values of both PSNR and SSIM for blind inpainting, but its visual output is clearly not satisfying.

To summarize, we observe that PCP+SSMS and K-PCP are not practical (in terms of long running times). Moreover, PCP+SSMS is not successful to restore images in general and K-PCP (with additional thresholding) also fails in some cases, for example the image *House*. PCP(capped)+SSMS and K-PCP(capped) are fast but do not work well in general. LRR is inappropriate because it is too slow and sensitive to the choice of parameters. LMaFit and K-LMaFit failed in general. The variants of the non-local median cannot handle Gaussian noise well. The best of them is comparable to K-ALS when Gaussian noise is low, but obviously worse as Gaussian noise arises. Our method, K-ALS, even with overestimating its main parameter p, outperforms the other competitors that we have discussed.

Table 5.1: Performance of denoising (in PSNR)

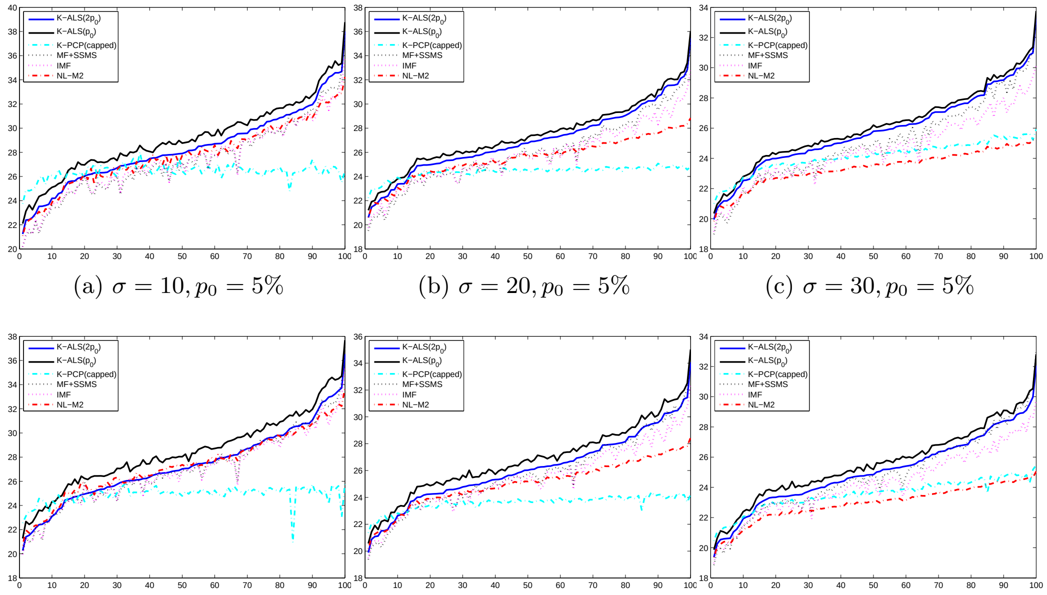
	p_0	σ	K-ALS($2p_0$)	K-ALS(p_0)	PCP(capped)+SSMS	K-PCP(capped)	KSVD	SSMS	MF+KSVD	MF+SSMS	IMF	NL-Median1	NL-Median2	NL-Median3	PCP+SSMS	K-PCP
Barbara	5%	10	**30.77**	**31.84**	26.35	26.61	22.06	22.81	24.74	24.96	24.63	29.69	29.73	26.32	24.11	29.18
		20	**28.68**	**29.22**	24.07	24.61	22.06	24.98	23.81	24.24	23.84	26.57	26.66	24.75	25.38	27.34
		30	**26.99**	**27.38**	23.56	24.55	23.16	26.22	22.93	23.67	23.05	23.87	23.96	23.15	26.19	25.67
	10%	10	**29.30**	**30.84**	24.34	25.31	19.26	19.59	24.59	24.77	24.35	28.97	29.20	25.79	19.77	28.45
		20	**27.55**	**28.64**	23.18	23.75	19.58	21.82	23.71	24.08	23.61	25.99	26.09	24.32	21.91	26.70
		30	**25.99**	**26.63**	22.99	23.93	21.00	23.85	22.81	23.52	22.89	23.41	23.47	22.82	23.96	25.20
Boat	5%	10	**30.74**	**31.49**	26.11	26.50	22.43	23.32	28.50	29.10	28.95	29.56	29.63	29.26	24.61	29.05
		20	**28.88**	**29.30**	24.51	24.55	22.54	25.46	26.55	27.46	27.06	26.59	26.69	27.11	25.94	27.43
		30	**27.38**	**27.69**	24.43	24.63	23.97	26.63	25.05	26.34	25.58	24.03	24.10	25.00	26.67	26.00
	10%	10	**29.70**	**30.84**	24.52	25.30	19.76	20.15	28.29	28.84	28.42	28.92	29.13	28.84	20.43	28.52
		20	**27.99**	**28.69**	23.74	23.79	20.55	22.55	26.39	27.23	26.69	26.11	26.21	26.70	22.67	26.98
		30	**26.58**	**27.09**	23.98	24.10	21.85	24.56	24.86	26.10	25.26	23.61	23.66	24.63	24.55	25.61
House	5%	10	**34.15**	**34.84**	26.11	28.15	22.42	23.51	31.85	32.16	31.09	31.84	31.85	31.66	33.90	28.26
		20	**31.76**	**32.17**	24.32	26.17	22.90	25.87	29.81	30.13	28.60	27.82	27.89	28.49	31.38	27.14
		30	29.79	**30.10**	24.80	26.43	24.92	28.11	27.24	28.81	26.93	24.74	24.81	25.80	**30.23**	25.99
	10%	10	**32.96**	**34.12**	24.75	26.66	19.75	20.25	31.27	31.52	30.42	31.24	31.41	31.20	28.75	27.74
		20	**30.55**	**31.32**	23.67	25.40	20.05	22.83	29.43	29.76	28.12	27.34	27.42	28.05	28.69	25.73
		30	**28.82**	**29.43**	24.35	25.71	22.34	25.55	26.93	28.38	26.47	24.29	24.35	25.35	28.16	25.61
Lena	5%	10	**33.47**	**34.19**	25.63	26.31	22.40	23.36	31.75	32.49	31.71	31.80	31.86	31.99	25.33	31.23
		20	**31.29**	**31.69**	23.92	24.53	22.76	25.62	29.28	30.40	29.17	27.72	27.81	28.64	26.36	29.39
		30	**29.54**	**29.85**	24.64	25.09	24.31	27.95	27.37	28.95	27.53	24.68	24.76	25.90	27.96	28.02
	10%	10	**32.35**	**33.49**	24.16	25.23	19.74	20.15	31.48	32.11	31.22	31.19	31.39	31.55	20.40	30.74
		20	**30.20**	**30.97**	23.39	23.85	20.01	22.61	29.07	30.11	28.82	27.25	27.34	28.17	22.56	28.96
		30	28.55	**29.15**	24.17	24.52	21.73	25.30	27.10	**28.60**	27.18	24.28	24.35	25.49	25.39	27.54
Peppers	5%	10	**32.81**	**33.44**	25.22	26.20	22.05	22.88	32.05	32.55	31.89	31.50	31.55	31.66	24.85	30.01
		20	**31.01**	**31.42**	24.16	24.54	22.32	24.94	30.03	30.64	29.53	27.64	27.71	28.44	25.87	28.75
		30	**29.48**	**29.82**	24.73	25.01	23.73	27.50	28.15	29.25	27.91	24.68	24.75	25.79	27.68	27.83
	10%	10	31.66	**32.81**	23.92	25.13	19.36	19.72	31.65	**32.08**	31.31	30.77	30.94	31.09	20.01	29.54
		20	29.88	**30.74**	23.40	23.78	19.57	21.96	29.69	**30.24**	29.10	27.13	27.20	27.91	21.99	28.37
		30	28.41	**29.07**	24.18	24.40	20.97	24.62	27.75	**28.81**	27.47	24.25	24.32	25.34	24.76	27.15

Table 5.2: Performance of inpainting (in PSNR)

	p_0	σ	K-ALS($2p_0$)	K-ALS(p_0)	PCP(capped)+SSMS	K-PCP(capped)	KSVD	SSMS	MF+KSVD	MF+SSMS	IMF	NL-Median1	NL-Median2	NL-Median3	PCP+SSMS	K-PCP
Barbara	0%	0	**32.26**	**32.97**	29.41	29.00	27.33	27.33	25.22	25.22	25.22	28.14	28.14	26.75	28.53	29.99
		5	**31.32**	**32.23**	28.49	28.31	27.09	27.10	25.15	25.23	25.06	27.79	27.85	26.60	27.95	29.25
		10	**30.41**	**30.90**	26.70	26.42	26.85	26.97	24.72	24.96	24.69	26.97	27.14	26.04	27.40	28.51
	5%	0	**29.67**	**31.60**	27.84	27.11	20.77	20.76	24.91	24.91	24.91	27.48	27.45	26.21	21.53	29.15
		5	**29.30**	**31.02**	27.23	26.68	20.96	21.00	24.90	24.99	24.77	27.20	27.25	26.05	21.72	28.56
		10	**28.98**	**29.97**	25.53	25.16	21.07	21.66	24.58	24.78	24.42	26.45	26.58	25.53	22.37	27.92
	10%	0	25.43	**30.45**	25.74	25.32	18.33	18.33	24.56	24.54	24.55	26.86	27.08	25.68	18.44	**28.40**
		5	26.05	**30.03**	25.34	25.12	18.44	18.43	24.62	24.70	24.46	26.55	26.84	25.53	18.53	**27.93**
		10	26.81	**28.95**	24.06	24.21	18.68	18.98	24.34	24.53	24.18	25.83	26.12	25.05	19.08	**27.28**
Boat	0%	0	**30.67**	**32.80**	29.76	28.63	26.96	26.96	30.31	30.31	30.31	27.83	27.94	29.54	28.88	30.65
		5	**30.25**	**31.84**	28.93	28.07	26.67	26.66	29.73	29.95	29.87	27.56	27.72	29.25	27.91	29.46
		10	**29.59**	**30.45**	26.48	26.15	26.40	26.51	28.41	29.01	28.89	26.82	27.07	28.57	27.08	28.58
	5%	0	29.66	**31.65**	27.59	26.93	21.06	21.05	**29.74**	29.72	29.72	27.28	27.32	29.24	21.92	29.37
		5	29.53	**30.70**	26.95	26.52	21.16	21.19	29.32	**29.55**	29.32	27.05	27.12	29.02	22.01	28.55
		10	**28.92**	**29.65**	25.32	25.01	21.35	21.98	28.14	28.72	28.41	26.36	26.51	28.26	22.69	28.00
	10%	0	28.00	**30.88**	25.57	25.42	18.66	18.66	28.84	28.94	**28.95**	26.73	26.94	28.90	18.83	28.75
		5	28.11	**30.02**	25.06	25.11	18.82	18.81	28.77	**28.98**	28.69	26.46	26.69	28.59	18.95	28.05
		10	27.99	**29.30**	23.94	24.11	19.08	19.42	27.74	**28.31**	28.00	25.87	26.12	27.92	19.63	27.50
House	0%	0	**36.44**	**35.89**	27.35	25.66	23.67	23.67	31.11	31.10	32.05	25.97	26.11	31.44	33.64	29.44
		5	**33.93**	**33.63**	27.02	25.49	23.66	23.66	30.89	31.05	31.57	25.65	25.97	30.37	31.38	29.10
		10	**32.19**	**31.42**	25.71	24.84	23.71	23.83	30.08	30.52	30.54	25.13	25.54	29.51	29.63	28.38
	5%	0	**35.41**	**36.58**	26.38	24.65	20.00	20.00	29.98	29.98	31.40	25.52	25.64	30.81	28.95	28.39
		5	**33.29**	**33.90**	25.81	24.62	20.08	20.12	29.99	30.06	30.92	25.25	25.49	30.14	27.59	28.52
		10	**31.79**	**31.54**	24.54	24.26	20.35	20.89	29.16	29.75	30.00	24.74	25.10	29.03	27.15	28.21
	10%	0	**32.67**	**35.80**	24.58	23.63	18.02	18.02	28.43	28.53	30.48	25.07	25.26	30.19	22.67	27.93
		5	**32.15**	**33.67**	24.26	23.69	18.16	18.16	28.94	28.74	30.23	24.84	25.18	29.71	22.36	28.85
		10	**31.20**	**32.23**	23.60	23.56	18.47	18.80	28.60	28.70	29.39	24.33	24.76	28.54	23.16	27.94
Lena	0%	0	**35.32**	**36.65**	30.94	29.23	27.72	27.72	34.38	34.38	34.38	29.40	29.50	33.11	30.90	32.81
		5	**34.15**	**34.97**	29.72	28.69	27.49	27.50	33.24	33.54	33.33	29.00	29.16	32.64	29.73	31.76
		10	**32.84**	**33.03**	26.26	26.62	27.36	27.47	31.49	32.29	31.70	28.03	28.28	31.30	28.75	30.82
	5%	0	**33.71**	**35.46**	27.99	27.00	21.19	21.19	33.50	33.44	33.44	28.87	28.93	32.78	22.46	31.71
		5	**33.10**	**34.29**	26.87	26.94	21.37	21.43	32.73	32.94	32.55	28.46	28.62	32.33	22.54	30.94
		10	**32.15**	**32.77**	24.80	25.39	21.45	22.12	31.14	31.89	31.28	27.64	27.81	30.90	23.23	30.25
	10%	0	31.24	**34.81**	25.45	25.55	18.72	18.72	32.00	32.18	**32.56**	28.38	28.57	32.44	18.90	31.15
		5	31.73	**33.61**	24.92	25.57	18.84	18.85	31.91	**32.19**	32.02	28.00	28.25	31.93	19.05	30.50
		10	31.17	**32.22**	23.76	24.45	19.17	19.51	30.57	**31.29**	30.75	27.14	27.42	30.49	19.75	29.75
Peppers	0%	0	29.22	31.26	31.36	29.23	27.68	27.67	**34.22**	**34.22**	**34.22**	29.19	29.28	32.93	30.97	32.08
		5	28.66	30.24	29.19	28.68	27.38	27.37	33.22	**33.46**	**33.34**	28.82	28.98	32.58	29.78	30.98
		10	29.50	30.85	27.62	26.80	20.93	20.93	31.89	**32.40**	**32.00**	27.89	28.18	31.28	28.74	30.23
	5%	0	29.50	30.85	27.62	26.80	20.93	20.93	**33.31**	33.25	**33.45**	28.64	28.73	32.48	22.21	31.31
		5	29.07	30.64	26.49	26.85	21.04	21.09	32.68	**32.89**	**32.80**	28.31	28.43	32.07	22.42	30.37
		10	29.09	30.40	24.60	25.38	21.16	21.74	31.45	**31.92**	**31.50**	27.49	27.69	30.79	23.02	29.75
	10%	0	28.54	31.09	25.12	25.32	18.43	18.42	31.85	**31.99**	**32.69**	28.16	28.37	32.12	18.69	30.65
		5	29.07	30.14	24.60	25.35	18.55	18.55	31.74	**31.96**	**32.17**	27.77	28.00	31.62	18.69	29.86
		10	28.70	29.98	23.62	24.40	18.80	19.11	30.74	**31.26**	**30.95**	27.02	27.27	30.37	19.36	29.14

6. Conclusions. We have shown that localized versions of robust PCA are sometimes necessary and sufficient to model corrupted data which has local (but not global) low dimensional structure. In particular, we see that the problems of denoising images with impulsive noise and blind inpainting can be approached via localized robust PCA methods. For these problems we have shown that a robust PCA algorithm, which is based on a RANSAC like alternating least square approach, performs very well. We have also established a convergence result for the proposed procedure. There is still much to do. In particular, there is almost no theory for when recovery is possible with a manifold model; we suspect that it may be possible to construct a theory analogous to that of [33].

Appendix A. Effect of the Parameters of the ALS. In this section, we show by experiments the effects of the selection for the two parameters p and d on the ALS algorithm. We first fix $d = d_0 + 3$ and vary the values for p. We compute the relative fitting error and plot it versus p for different experimental settings (with different m, d_0 and p_0) in Figures A.1 and A.2. Figures A.3 and A.4 plot the relative fitting error versus d when fixing $p = 2p_0$. We conclude from these figures that the ALS algorithm tends to be robust to the overestimation of the rank and it also allows the overestimation of p to some degree. The



(a) $\sigma = 10, p_0 = 5\%$ (b) $\sigma = 20, p_0 = 5\%$ (c) $\sigma = 30, p_0 = 5\%$

(d) $\sigma = 10, p_0 = 10\%$ (e) $\sigma = 20, p_0 = 10\%$ (f) $\sigma = 30, p_0 = 10\%$

Figure 5.2: Performance of denoising for BSD database.

ALS algorithm is less robust to the underestimation of these two parameters, however, it can still perform well under very mild underestimation.

Appendix B. Estimation of the Rank in ALS. Since ALS is robust to overestimation of the rank, we suggest the following strategy to estimate the underlying low rank of a corrupted matrix. We apply ALS with d_i,where $d_i < d_{i+1}$, $i = 1, \ldots, t$, and denote the corresponding low-rank estimators of $\mathbf{X}$ by $\tilde{\mathbf{X}}_1, \ldots, \tilde{X}_t$. We then estimate the ranks of these estimators and thus obtain a sequence of estimated ranks $\{\tilde{d}_1, \ldots, \tilde{d}_t\}$. This sequence would increase first, be equal to the underlying rank d_0 for a while and finally increase again because of the corruptions. Finding the "stable phase" (i.e., when the sequence does not increase) will determine the appropriate rank.

To estimate the ranks of the different estimators, we apply second order differences (SODs) to detect abrupt changes in the singular values of these estimators. Indeed, if the underlying rank of a matrix is r, then the top $(r+1)$ singular values of the matrix drop very quickly, while the ones following them decrease more slowly. The SOD of the singular values thus expect to obtain the maximum at $r+1$. If $\sigma_1, \ldots, \sigma_m$ are the singular values of an $m \times n$ estimator (assume $m < n$ WLOG), then the SODs obtain the form:

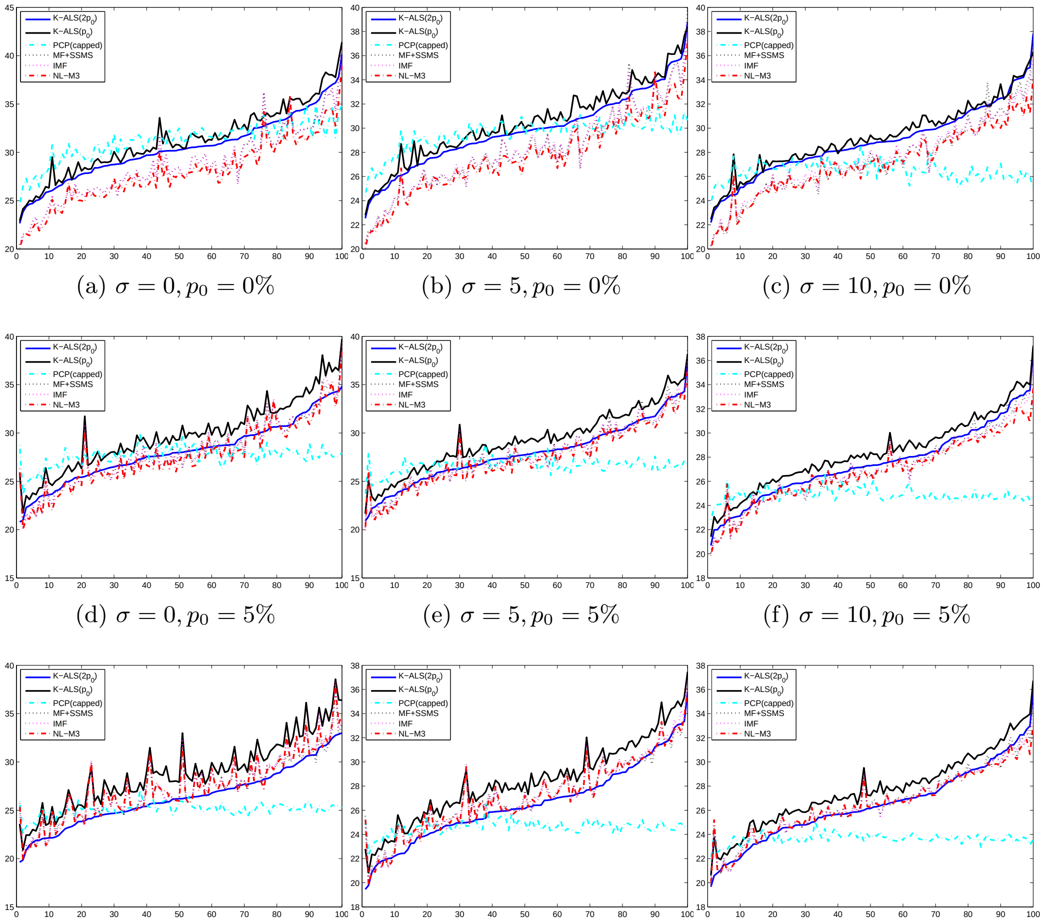
$$S(i) = \sigma_{i-1} + \sigma_{i+1} - 2\sigma_i, \; i = 2, \ldots, m-1, \tag{B.1}$$

and the estimated rank $\hat{d}$ can be expressed by

$$\hat{d} = \arg\max_i S(i) - 1. \tag{B.2}$$

Note that this procedure is not able to detect the rank of a matrix if $r \geq m-2$. However, in most problems where the rank is relatively low, this is not the case.

We estimated the rank for the synthetic setting of §2.3. We chose an arithmetic sequence $\{d_i\}_{i=1}^{2}0$ such that $d_1 = 1$ and $d_{20} = 0.2m$. We compared with the rank estimated by PCP and PCP(capped), where the cap for PCP(capped) was $0.2m$. We considered both output



(a) $\sigma = 0, p_0 = 0\%$ (b) $\sigma = 5, p_0 = 0\%$ (c) $\sigma = 10, p_0 = 0\%$

(d) $\sigma = 0, p_0 = 5\%$ (e) $\sigma = 5, p_0 = 5\%$ (f) $\sigma = 10, p_0 = 5\%$

(g) $\sigma = 0, p_0 = 10\%$ (h) $\sigma = 5, p_0 = 10\%$ (i) $\sigma = 10, p_0 = 10\%$

Figure 5.3: Performance of denoising for BSD database.

values of d_0 and d_0+1 as successes (after all, ALS is not sensitive to overestimation and the estimated ranks were either no greater than d_0+1 or equal to d_{20}). All synthetic scenarios were repeated 100 times. The successful rank identification rates for ALS, PCP(capped) and PCP are shown in Tables B.1 and B.2.

Table B.1: The correctness($e\%$) of rank estimation without Gaussian noise

m	100			200			400			800		
d_0	5			10			20			40		
p_0	5%	10%	20%	5%	10%	20%	5%	10%	20%	5%	10%	20%
ALS-reg($2p_0$)	100	100	100	100	100	100	100	100	100	100	100	100
PCP(capped)	100	100	79	100	100	100	100	100	100	100	100	100
PCP	100	100	100	100	100	100	100	100	100	100	100	100

Table B.2: The correctness($e\%$) of rank estimation with Gaussian noise ($\sigma = 0.05$)

m	100			200			400			800		
d_0	5			10			20			40		
p_0	5%	10%	20%	5%	10%	20%	5%	10%	20%	5%	10%	20%
ALS-reg($2p_0$)	100	100	99	100	100	100	100	100	100	100	100	100
PCP(capped)	100	100	73	100	100	77	100	100	100	100	100	100
PCP	100	100	100	100	100	100	100	100	100	100	100	100

Appendix C. Restoration of Corrupted Images in SSIM. We report the results of the

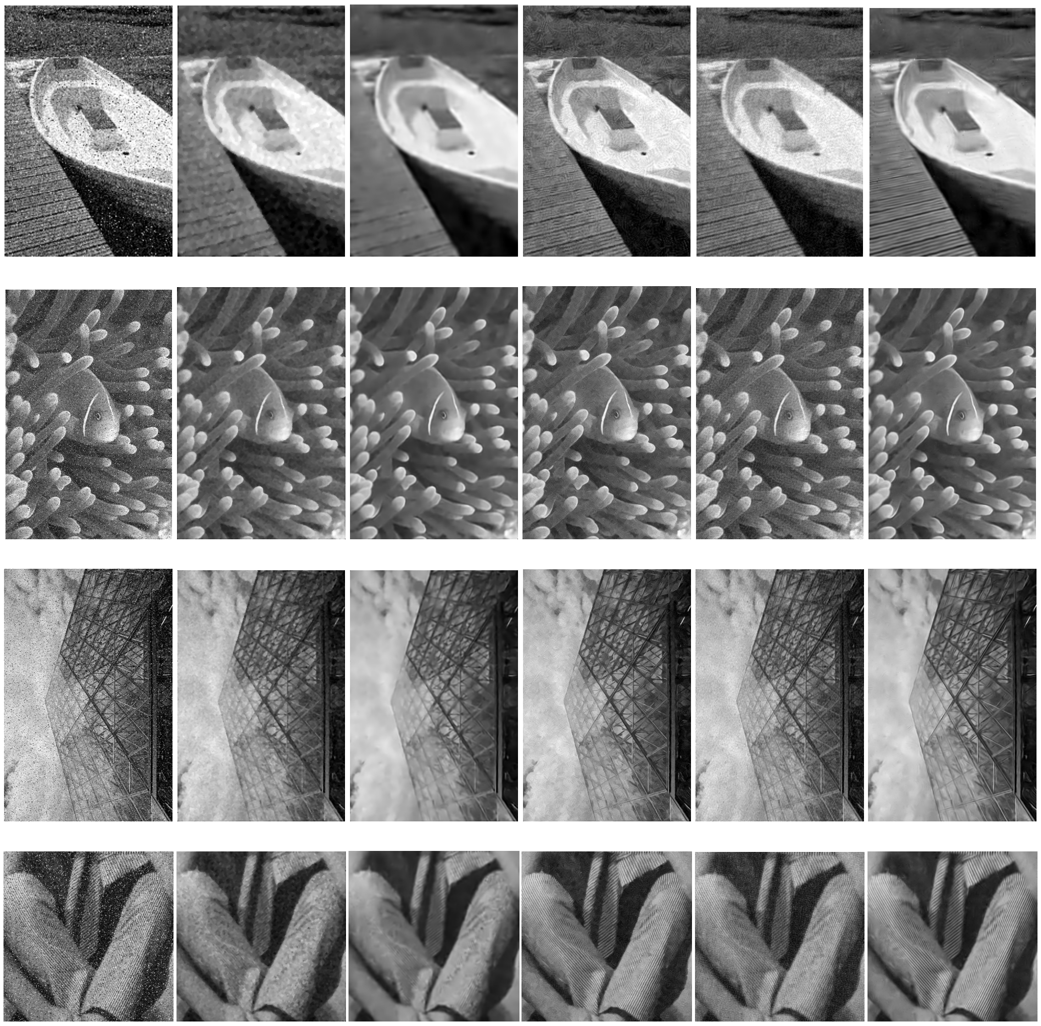


Figure 5.4: From left to right: noisy images, IMF, MF+SSMS, NLM3, K-PCP(capped) and K-ALS($2p_0$). The images at the third row are transposed for larger illustration. For the noisy images, the first one has Gaussian noise with $\sigma = 30$ and the rest has it with $\sigma = 20$. While the first three at the same time has 5% impulsive noise and the last has 10%.

experiments in §5.4 using SSIM [36] instead of PSNR. The SSIM of the five common images for impulsive noise denoising are shown Table C.1 and for blind inpainting are shown in Table C.2. The SSIM of the BSD database are shown in Figure C.1 and C.2 for denoising and blind inpainting respectively.

Appendix D. Mathematical Analysis of the ALS Algorithm. We analyze the performance of the ALS algorithm following the strategies of [30, 2].

D.1. Preliminaries.

D.1.1. Notation. Let $\mathbf{1}_{m\times n}$ denote an $m \times n$ matrix whose elements are all equal to 1; $\mathbf{I}$ denote the identity matrix (whose dimension will be clear from the context) and $|\mathbf{A}|_0$ denote the number of nonzero elements of the matrix $\mathbf{A}$.

D.1.2. Point-to-Set Maps. Definition D.1 (Point-to-Set Map). *Given two sets $\mathcal{X}$,$\mathcal{Y}$, a point-to-set map Ω is a function $\Omega\colon \mathcal{X} \to \mathcal{P}(\mathcal{Y})$. The composition of two point-to-set maps $\Omega_1\colon \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ and $\Omega_2\colon \mathcal{Y} \to \mathcal{P}(\mathcal{Z})$ is defined by $(\Omega_2 \circ \Omega_1)(\mathbf{x}) = \bigcup_{\mathbf{y}\in\Omega_1(\mathbf{x})} \Omega_2(\mathbf{y})$.*

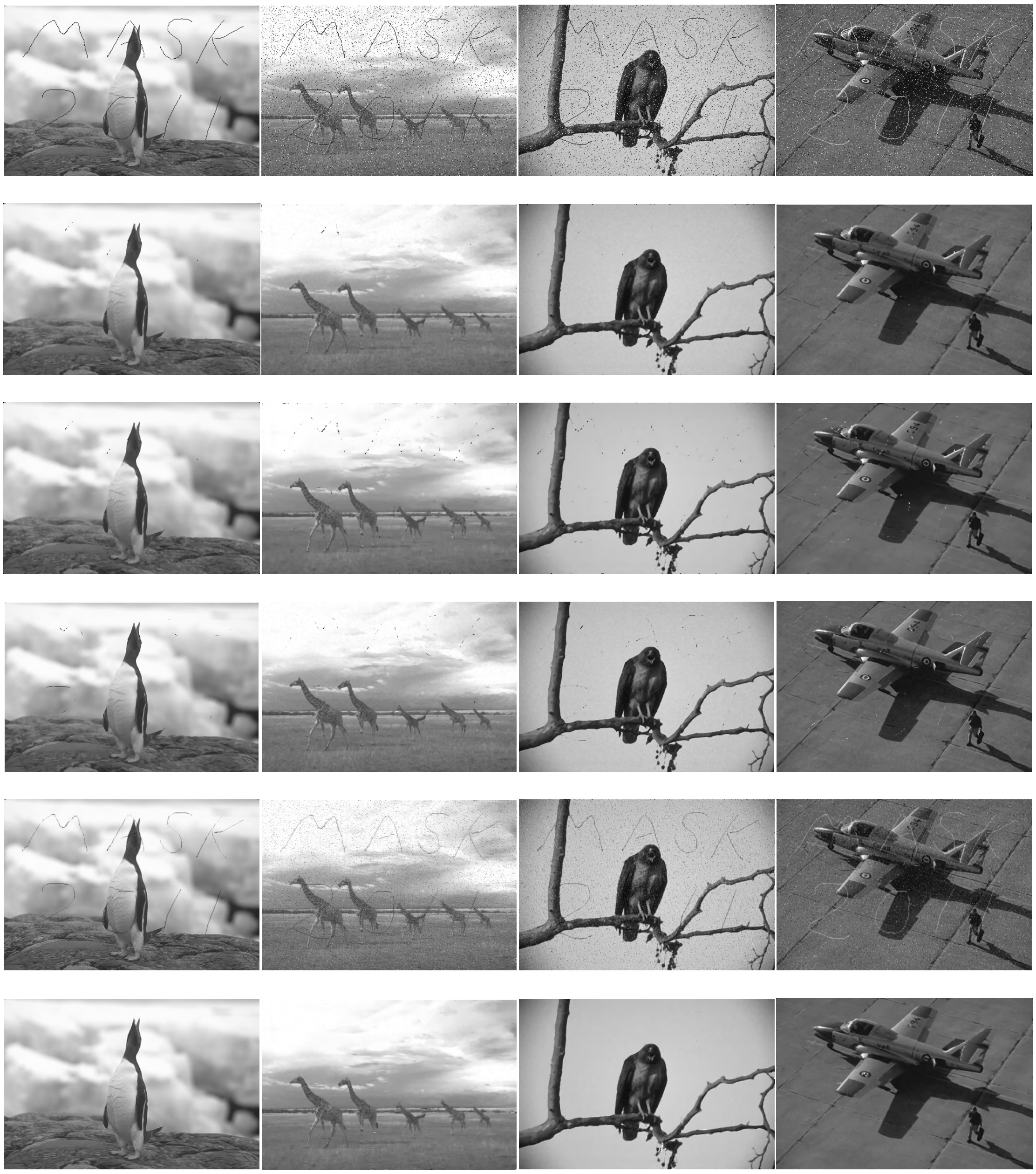


Figure 5.5: From top to bottom: noisy images, IMF, MF+SSMS, NLM3, PCP(capped)+SSMS and K-ALS($2p_0$). The first noisy image has only scratches. The rest also has 5% impulsive noise and $\sigma = 10$ Gaussian noise.

Definition D.2 (Closed Map).*A point-to-set map Ω is closed at $\hat{\mathbf{x}} \in \mathcal{X}$ if $\{\mathbf{x}_k\} \subset \mathcal{X}$, $\mathbf{x}_k \to \hat{\mathbf{x}}$, $\mathbf{y}_k \in \Omega(\mathbf{x}_k)$ and $\mathbf{y}_k \to \hat{\mathbf{y}}$ together imply that $\hat{\mathbf{y}} \in \Omega(\hat{\mathbf{x}})$.*

Definition D.3 (Fixed Point).*A fixed point of the map $\Omega : \mathcal{X} \to \mathcal{P}(\mathcal{X})$ is a point $\mathbf{x}$ for which $\{\mathbf{x}\} = \Omega(\mathbf{x})$. A generalized point of Ω is a point $\mathbf{x}$ for which $\mathbf{x} \in \Omega(\mathbf{x})$.*

D.1.3. Iterative Algorithms. An iterative algorithm is a point-to-set map $\Omega : \mathcal{X} \to \mathcal{P}(\mathcal{X})$. Given an initial point $\mathbf{x}_0$, an algorithm generates a sequence of points via the rule $\mathbf{x}_{k+1} \in \Omega(\mathbf{x}_k)$. Now, suppose that $\phi : \mathcal{X} \to \mathbb{R}_+$ is a continuous, non-negative function. An algorithm Ω is *monotonic* with respect to ϕ whenever $\mathbf{y} \in \Omega(\mathbf{x})$ implies that $\phi(\mathbf{y}) \leq \phi(\mathbf{x})$.

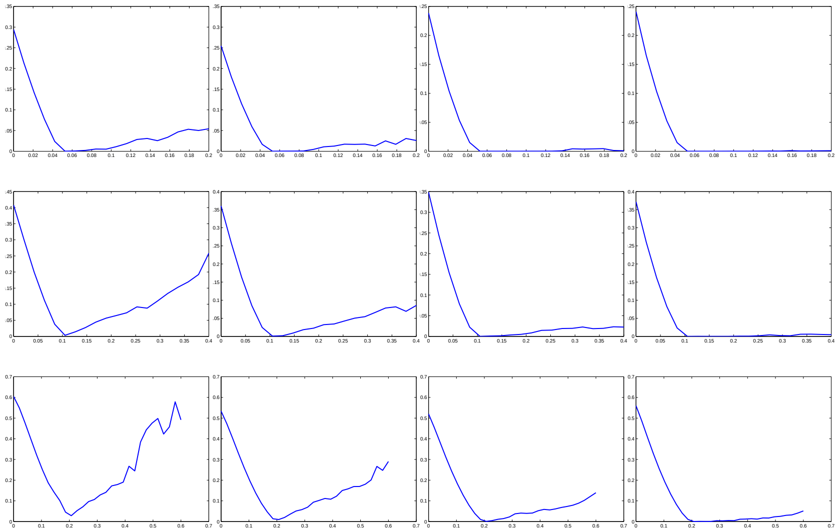


Figure A.1: The relative fitting error with different values of the parameter p without Gaussian noise. From top to bottom, $p_0 = 0.05, 0.1$ and 0.2 respectively. From left to right, $m = 100, 200, 400$ and 800 respectively.

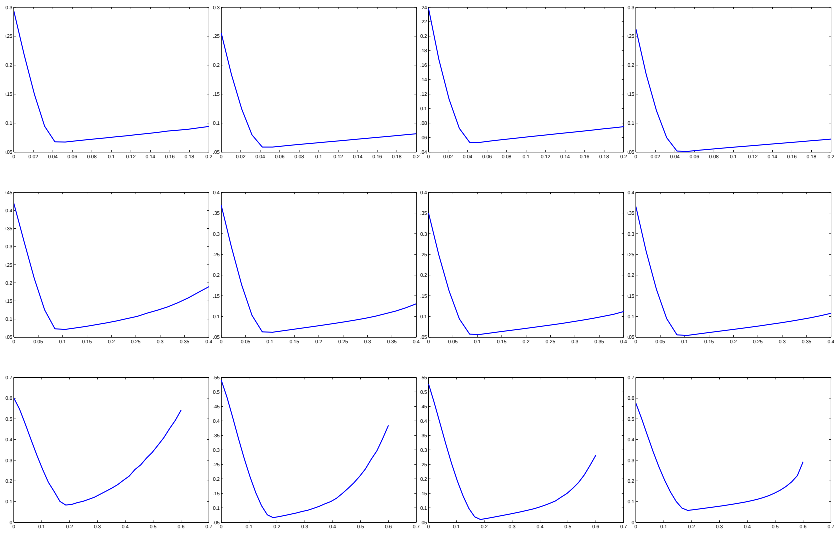


Figure A.2: The relative fitting error with different values of the parameter p with Gaussian noise ($\sigma = 0.05$). From top to bottom, $p_0 = 0.05, 0.1$ and 0.2 respectively. From left to right, $m = 100, 200, 400$ and 800 respectively.

If, in addition, $\mathbf{y} \in \Omega(\mathbf{x})$ and $\phi(\mathbf{y}) = \phi(\mathbf{x})$ imply that $\mathbf{y} = \mathbf{x}$, then we say that the algorithm is *strictly monotonic.*

Theorem D.4 (Zangwill [37]).*Let Ω be an algorithm that is monotonic with respect to ϕ. Given an initial point $\mathbf{x}_0$, suppose that the algorithm generates a sequence $\{\mathbf{x}_k\}$ that lies in a compact set. Then the sequence has at least one accumulation point $\hat{\mathbf{x}}$, and $\phi(\hat{\mathbf{x}}) = \lim \phi(\mathbf{x}_k)$. Moreover, if Ω is closed at $\hat{\mathbf{x}}$ then $\hat{\mathbf{x}}$ is a generalized fixed point of the algorithm.*

Theorem D.5 (Meyer [23]).*Assume that the algorithm Ω is strictly monotonic with respect to ϕ and that it generates a sequence $\{\mathbf{x}_k\}$ which lies in a compact set. If Ω is closed at an accumulation point $\hat{\mathbf{x}}$ then $\hat{\mathbf{x}}$ is a fixed point of Ω. Moreover, if $\mathcal{X}$ is normed, $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \to$*

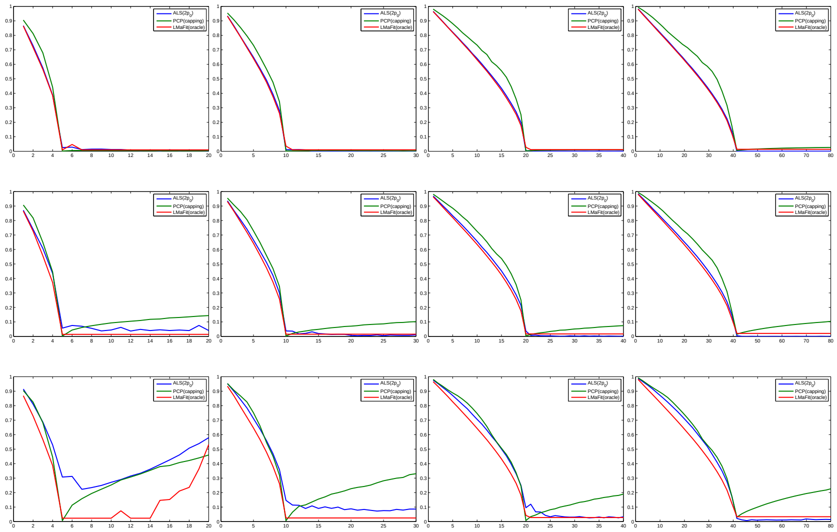


Figure A.3: The relative fitting error with different values of the parameter d without Gaussian noise. From top to bottom, $p_0 = 0.05, 0.1$ and 0.2 respectively. From left to right, $m = 100, 200, 400$ and 800 respectively.

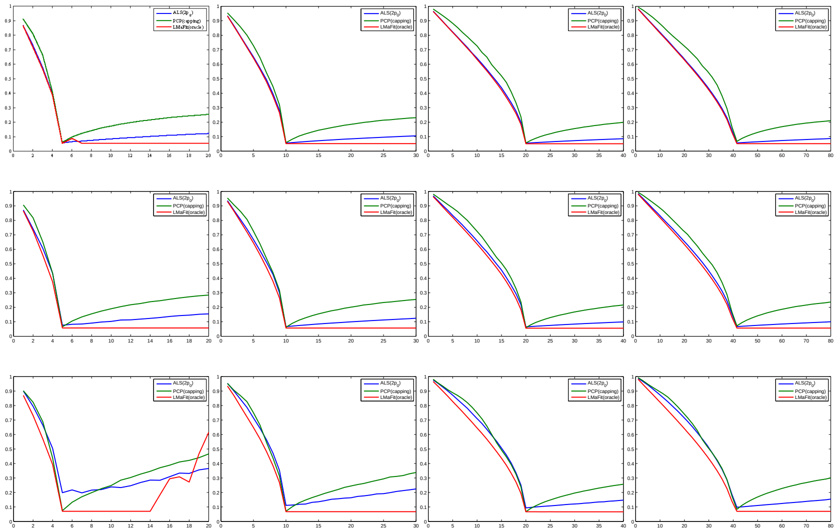


Figure A.4: The relative fitting error with different values of the parameter d with Gaussian noise ($\sigma = 0.05$). From top to bottom, $p_0 = 0.05, 0.1$ and 0.2 respectively. From left to right, $m = 100, 200, 400$ and 800 respectively.

0. *It follows that* $\{\mathbf{x}_k\}$ *converges in norm to* $\hat{\mathbf{x}}$ *or that the accumulation points of* $\mathbf{x}_k$ *form a continuum.*

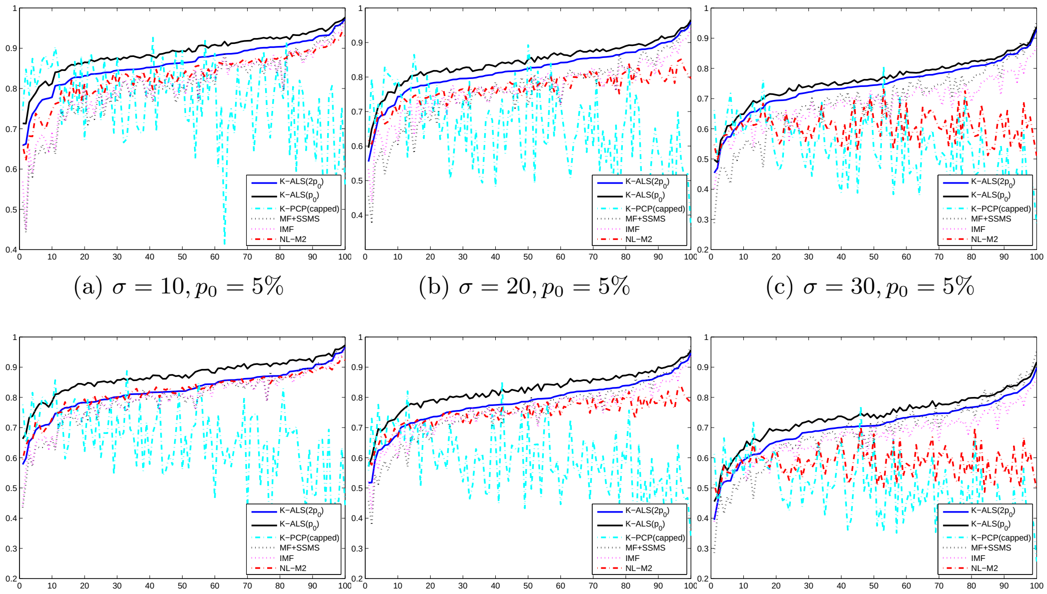
D.1.4. Infimal Maps. For $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, we define the *infimal map* $M_{\mathbf{y}} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ by $M_{\mathbf{y}}(\mathbf{x}) = \arg\min_{\mathbf{y} \in \mathcal{Y}} \phi(\mathbf{x}, \mathbf{y})$. We similarly define $M_{\mathbf{x}} : \mathcal{Y} \to \mathcal{P}(\mathcal{X})$.

Theorem D.6 (Dantzig-Folkman-Shapiro [10]).*If* $\phi(\hat{\mathbf{x}}, \cdot)$ *is continuous on* $\mathcal{Y}$*, then* $M_{\mathbf{y}}$ *is closed at* $\hat{\mathbf{x}}$*.*

Theorem D.7 (Fiorot-Huard [13]).*If the infimal maps* $M_{\mathbf{x}}$ *and* $M_{\mathbf{y}}$ *are both single-valued then the algorithm* $\Omega \triangleq M_{\mathbf{x}} \circ M_{\mathbf{y}}$ *is strictly monotonic with respect to* ϕ*.*

Table C.1: Performance of denoising (in SSIM)

	p_0	σ	K-ALS$(2p_0)$	K-ALS(p_0)	PCP(capped)+SSMS	K-PCP(capped)	KSVD	SSMS	MF+KSVD	MF+SSMS	IMF	NL-Median1	NL-Median2	NL-Median3	PCP+SSMS	K-PCP
Barbara	5%	10	**0.897**	**0.912**	0.654	0.656	0.582	0.616	0.726	0.754	0.727	0.845	0.846	0.790	0.641	0.874
		20	**0.836**	**0.849**	0.538	0.557	0.578	0.672	0.654	0.692	0.642	0.674	0.679	0.652	0.681	0.812
		30	**0.777**	**0.791**	0.509	0.554	0.625	0.710	0.604	0.653	0.595	0.530	0.535	0.529	0.708	0.750
	10%	10	**0.867**	**0.896**	0.566	0.607	0.425	0.442	0.723	0.749	0.717	0.827	0.832	0.772	0.444	0.859
		20	**0.799**	**0.826**	0.485	0.521	0.421	0.516	0.651	0.686	0.634	0.653	0.655	0.630	0.517	0.797
		30	0.733	**0.763**	0.473	0.529	0.495	0.600	0.601	0.648	0.586	0.507	0.509	0.507	0.603	0.739
Boat	5%	10	**0.839**	**0.854**	0.610	0.618	0.525	0.571	0.750	0.792	0.787	0.790	0.794	0.790	0.601	0.807
		20	**0.770**	**0.783**	0.496	0.500	0.523	0.633	0.673	0.727	0.700	0.613	0.619	0.663	0.640	0.742
		30	**0.715**	**0.727**	0.489	0.496	0.581	0.663	0.622	0.682	0.641	0.468	0.473	0.545	0.662	0.688
	10%	10	**0.812**	**0.841**	0.529	0.565	0.383	0.404	0.746	0.787	0.777	0.769	0.777	0.777	0.404	0.795
		20	**0.740**	**0.765**	0.465	0.464	0.368	0.485	0.668	0.723	0.690	0.594	0.596	0.646	0.491	0.728
		30	**0.679**	**0.704**	0.472	0.471	0.461	0.567	0.618	0.678	0.630	0.447	0.448	0.528	0.572	0.679
House	5%	10	**0.878**	**0.888**	0.508	0.600	0.482	0.537	0.848	0.856	0.834	0.803	0.804	0.815	0.874	0.821
		20	**0.836**	**0.844**	0.410	0.497	0.519	0.604	0.812	0.824	0.776	0.594	0.598	0.655	0.816	0.794
		30	**0.797**	**0.805**	0.435	0.527	0.612	0.684	0.759	0.800	0.712	0.436	0.442	0.519	0.790	0.767
	10%	10	**0.861**	**0.879**	0.452	0.547	0.323	0.350	0.845	0.853	0.792	0.801	0.790	0.801	0.681	0.816
		20	0.810	**0.828**	0.391	0.469	0.338	0.447	0.811	**0.821**	0.767	0.578	0.578	0.635	0.715	0.794
		30	0.765	**0.784**	0.416	0.498	0.449	0.559	0.761	**0.794**	0.702	0.421	0.422	0.502	0.698	0.777
Lena	5%	10	**0.884**	**0.894**	0.498	0.530	0.477	0.531	0.849	0.871	0.846	0.809	0.811	0.829	0.574	0.868
		20	**0.837**	**0.847**	0.402	0.429	0.516	0.616	0.799	0.831	0.781	0.597	0.603	0.670	0.632	0.826
		30	0.794	**0.805**	0.427	0.451	0.605	0.700	0.759	**0.800**	0.719	0.438	0.442	0.532	0.703	0.793
	10%	10	0.864	**0.884**	0.433	0.485	0.318	0.342	0.848	**0.868**	0.839	0.795	0.796	0.817	0.351	0.863
		20	0.808	**0.828**	0.383	0.400	0.337	0.447	0.797	**0.828**	0.773	0.582	0.582	0.652	0.441	0.819
		30	0.758	0.781	0.414	0.428	0.441	0.576	0.755	**0.795**	0.709	0.420	0.422	0.512	0.578	**0.784**
Peppers	5%	10	**0.850**	**0.860**	0.479	0.523	0.453	0.503	0.834	0.847	0.830	0.793	0.794	0.801	0.549	0.838
		20	0.812	**0.822**	0.401	0.425	0.488	0.578	0.802	**0.818**	0.776	0.592	0.596	0.650	0.600	0.810
		30	0.776	**0.786**	0.437	0.444	0.575	0.673	0.771	**0.792**	0.715	0.434	0.439	0.515	0.676	0.783
	10%	10	0.833	**0.851**	0.425	0.478	0.302	0.324	0.831	**0.844**	0.825	0.779	0.780	0.790	0.332	0.834
		20	0.780	0.802	0.380	0.391	0.313	0.416	0.798	**0.813**	0.768	0.575	0.575	0.631	0.414	**0.803**
		30	0.740	0.763	0.417	0.422	0.407	0.538	0.766	**0.787**	0.706	0.417	0.419	0.498	0.546	**0.777**



(a) $\sigma = 10, p_0 = 5\%$ (b) $\sigma = 20, p_0 = 5\%$ (c) $\sigma = 30, p_0 = 5\%$

(d) $\sigma = 10, p_0 = 10\%$ (e) $\sigma = 20, p_0 = 10\%$ (f) $\sigma = 30, p_0 = 10\%$

Figure C.1: Performance of denoising (in SSIM) for BSD database.

D.2. The ALS Algorithm as a Point-to-Set Map. In ALS the energy function J of (4.1) serves as the function ϕ of §D.1.3. Clearly J is continuous on $\mathbb{R}^{m\times d} \times \mathbb{R}^{d\times n} \times \mathbb{R}^{m\times n}$. Let $\mathcal{F} := \{\mathbf{W} : \mathbf{W} \in \{0,1\}^{m\times n}, |\mathbf{W}|_0 = \lfloor(1-p)mn\rfloor\}$. We assume that $d < \min(m,n)$

Table C.2: Performance of inpainting (in SSIM)

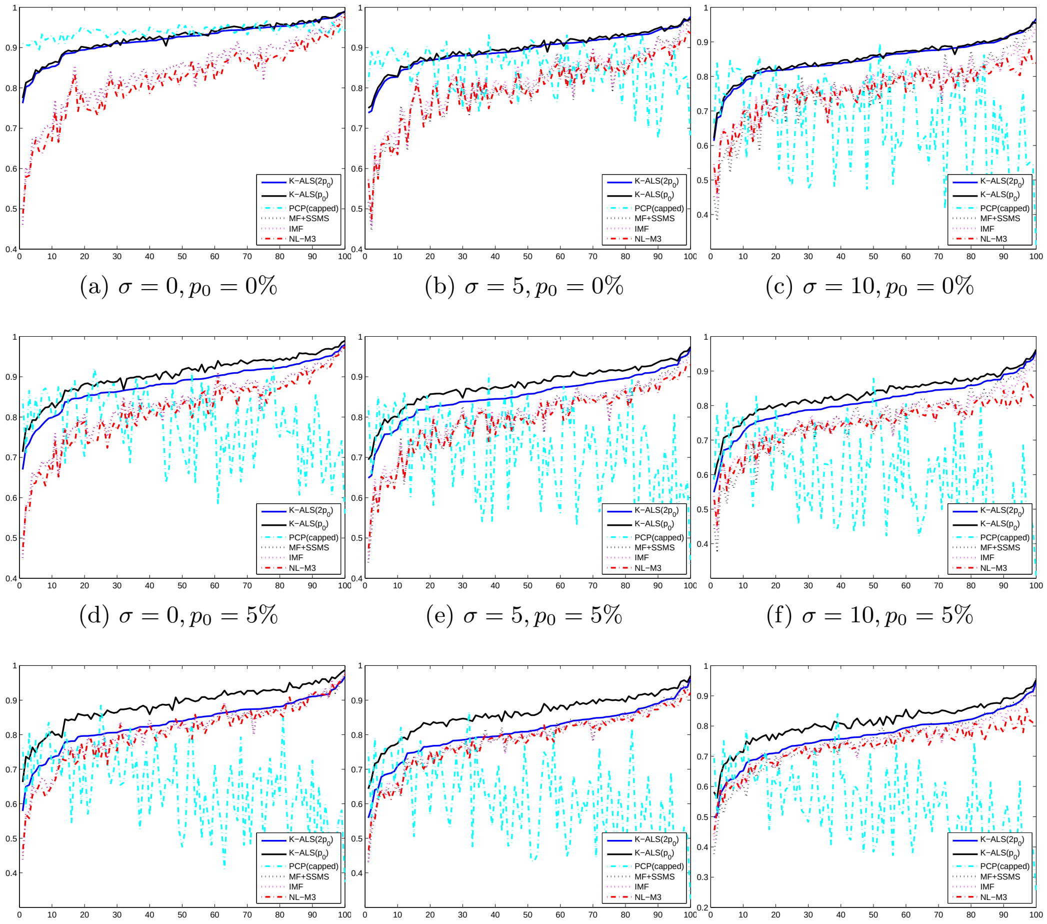
	p_0	σ	K-ALS($2p_0$)	K-ALS(p_0)	PCP(capped)+SSMS	K-PCP(capped)	KSVD	SSMS	MF+KSVD	MF+SSMS	IMF	NL-Median1	NL-Median2	NL-Median3	PCP+SSMS	K-PCP
Barbara	0%	0	**0.956**	**0.959**	0.938	0.944	0.945	0.945	0.809	0.809	0.809	0.916	0.915	0.860	0.949	0.937
		5	**0.932**	**0.936**	0.873	0.891	0.912	0.911	0.780	0.789	0.784	0.885	0.887	0.839	0.915	0.898
		10	**0.904**	**0.908**	0.730	0.717	0.886	0.887	0.725	0.755	0.732	0.811	0.819	0.788	0.888	0.866
	5%	0	**0.936**	**0.950**	0.851	0.808	0.564	0.5664	0.798	0.798	0.798	0.902	0.897	0.846	0.587	0.920
		5	**0.913**	**0.927**	0.791	0.765	0.562	0.567	0.773	0.782	0.772	0.873	0.871	0.824	0.583	0.883
		10	**0.882**	**0.895**	0.657	0.644	0.557	0.589	0.721	0.749	0.722	0.801	0.802	0.770	0.610	0.851
	10%	0	0.863	**0.938**	0.710	0.690	0.408	0.407	0.784	0.784	0.784	0.883	0.890	0.831	0.406	**0.903**
		5	0.848	**0.914**	0.670	0.667	0.406	0.407	0.764	0.772	0.756	0.853	0.860	0.807	0.407	**0.868**
		10	0.830	**0.877**	0.574	0.594	0.406	0.423	0.715	0.741	0.711	0.781	0.786	0.751	0.431	**0.837**
Boat	0%	0	0.920	0.930	0.930	0.921	**0.932**	**0.932**	0.847	0.847	0.847	0.872	0.871	0.851	0.941	0.929
		5	**0.884**	**0.894**	0.853	0.853	0.878	0.871	0.812	0.830	0.825	0.842	0.844	0.832	0.877	0.852
		10	**0.843**	**0.851**	0.664	0.673	0.826	0.826	0.748	0.791	0.789	0.765	0.773	0.787	0.828	0.809
	5%	0	0.900	**0.918**	0.793	0.784	0.538	0.538	0.838	0.838	0.838	0.852	0.848	0.845	0.566	**0.902**
		5	**0.864**	**0.882**	0.730	0.737	0.526	0.528	0.808	0.824	0.818	0.824	0.822	0.825	0.553	0.831
		10	**0.821**	**0.837**	0.598	0.602	0.508	0.548	0.744	0.786	0.781	0.749	0.751	0.776	0.572	0.792
	10%	0	0.862	**0.907**	0.656	0.665	0.369	0.369	0.824	0.824	0.827	0.828	0.839	0.837	0.372	**0.886**
		5	**0.829**	**0.868**	0.613	0.635	0.369	0.369	0.800	0.815	0.810	0.800	0.809	0.814	0.371	0.821
		10	**0.790**	**0.824**	0.520	0.548	0.367	0.386	0.737	0.779	0.770	0.729	0.735	0.764	0.390	0.777
House	0%	0	**0.955**	**0.954**	0.866	0.855	0.849	0.849	0.868	0.868	0.883	0.840	0.839	0.881	0.929	0.899
		5	**0.914**	**0.915**	0.775	0.810	0.809	0.805	0852	0.860	0.865	0.796	0.802	0.846	0.867	0.856
		10	**0.870**	**0.866**	0.567	0.629	0.767	0.770	0.831	0.841	0.831	0.704	0.715	0.788	0.817	0.815
	5%	0	**0.953**	**0.961**	0.703	0.692	0.457	0.457	0.854	0.855	0.877	0.824	0.824	0.872	0.755	0.871
		5	**0.907**	**0.915**	0.614	0.677	0.438	0.443	0.845	0.848	0.858	0.785	0.787	0.841	0.744	0.837
		10	**0.863**	**0.867**	0.480	0.563	0.435	0.474	0.820	0.831	0.826	0.698	0.702	0.776	0.740	0.809
	10%	0	**0.925**	**0.956**	0.540	0.566	0.300	0.300	0.829	0.833	0.867	0.806	0.816	0.867	0.454	0.861
		5	**0.881**	**0.910**	0.498	0.564	0.295	0.297	0.829	0.831	0.848	0.769	0.778	0.832	0.435	0.839
		10	**0.844**	**0.862**	0.433	0.510	0.301	0.323	0.812	0.818	0.819	0.686	0.690	0.766	0.504	0.809
Lena	0%	0	**0.950**	**0.955**	0.932	0.924	0.933	0.933	0.918	0.918	0.918	0.900	0.900	0.909	0.943	0.940
		5	**0.916**	**0.921**	0.809	0.837	0.883	0.880	0.883	0.894	0.889	0.860	0.864	0.885	0.888	0.887
		10	**0.886**	**0.888**	0.549	0.590	0.851	0.852	0.846	0.869	0.847	0.762	0.772	0.829	0.856	0.861
	5%	0	**0.936**	**0.949**	0.722	0.710	0.480	0.480	0.909	0.910	0.910	0.887	0.884	0.905	0.513	0.916
		5	**0.903**	**0.914**	0.629	0.672	0.469	0.476	0.879	0.889	0.884	0.849	0.849	0.880	0.512	0.874
		10	**0.872**	**0.881**	0.474	0.518	0.458	0.501	0.843	0.865	0.841	0.756	0.759	0.818	0.540	0.850
	10%	0	0.907	**0.942**	0.553	0.582	0.305	0.305	0.897	0.897	0.900	0.860	0.878	0.900	0.306	**0.908**
		5	**0.883**	**0.906**	0.510	0.563	0.302	0.304	0.872	0.882	0.878	0.833	0.840	0.872	0.308	0.867
		10	0.851	**0.871**	0.428	0.471	0.308	0.329	0.837	**0.859**	0.833	0.742	0.745	0.806	0.335	0.842
Peppers	0%	0	**0.928**	0.924	0.912	0.920	**0.928**	**0.928**	0.886	0.886	0.886	0.880	0.880	0.875	**0.938**	0.916
		5	**0.878**	**0.875**	0.753	0.810	0.860	0.851	0.856	0.864	0.866	0.842	0.846	0.855	0.860	0.849
		10	**0.847**	**0.846**	0.512	0.571	0.818	0.818	0.832	0.846	0.831	0.749	0.759	0.803	0.822	0.827
	5%	0	**0.916**	**0.925**	0.689	0.688	0.466	0.466	0.878	0.878	0.879	0.865	0.864	0.872	0.497	0.911
		5	**0.869**	**0.877**	0.603	0.657	0.449	0.452	0.852	0.860	0.861	0.830	0.831	0.849	0.491	0.844
		10	0.840	**0.844**	0.465	0.507	0.437	0.477	0.828	**0.841**	0.826	0.742	0.745	0.793	0.514	0.823
	10%	0	0.889	**0.919**	0.533	0.561	0.291	0.291	0.865	0.866	0.874	0.845	0.857	0.868	0.296	**0.903**
		5	**0.852**	**0.872**	0.493	0.547	0.290	0.290	0.844	**0.852**	0.855	0.812	0.821	0.843	0.288	0.837
		10	0.821	**0.836**	0.419	0.462	0.288	0.307	0.821	**0.834**	0.820	0.727	0.731	0.782	0.313	0.816

and p is the estimated portion of corruptions. The ALS algorithm minimizes J among $(\mathbf{B},\mathbf{C},\mathbf{W}) \in \mathbb{R}^{m\times d} \times \mathbb{R}^{d\times n} \times \mathcal{F}$. Defining, $M_{\mathbf{B}}(\mathbf{C},\mathbf{W}) = \arg\min_{\mathbf{B}} J$, $M_{\mathbf{C}}(\mathbf{B},\mathbf{W}) = \arg\min_{\mathbf{C}} J$ and $M_{\mathbf{W}}(\mathbf{B},\mathbf{C}) = \arg\min_{\mathbf{W}\in\mathcal{F}} J$, we rewrite the ALS Algorithm as

$$\Omega = M_{\mathbf{W}} \circ (M_{\mathbf{B}} \circ M_{\mathbf{C}})^t, \tag{D.1}$$

where t is the number of iterations of the inner loop.

D.3. Conclusion of Theorem 4.1. At each step, the ALS Algorithm is composed of $2t+1$ infimal maps as in (D.1). For fixed $\mathbf{B}$, the product map $\mathbf{C} \mapsto \mathbf{BC}$ is continuous and J is continuous w.r.t. $\mathbf{BC}$. Thus, $J(\mathbf{B},\cdot,\mathbf{W})$ is continuous for fixed $\mathbf{B}$ and $\mathbf{W}$. As a result, Theorem D.6 implies that both $M_{\mathbf{B}}$ and $M_{\mathbf{C}}$ are closed. Meanwhile, $M_{\mathbf{W}}$ is closed because $\mathcal{F}$ is finite. Therefore, the ALS Algorithm is closed. Lemmata D.8 and D.9, which are formulated and proved below, imply that the ALS algorithm is composed by single-valued infimal maps. Therefore, in view of Theorem D.7 the ALS algorithm is strictly monotonic. Consequently, $\lambda_1\|\mathbf{B}^t\|_F^2 + \lambda_2\|\mathbf{C}^t\|_F^2 \le J(\mathbf{B}^t,\mathbf{C}^t,\mathbf{W}^t) < J(\mathbf{B}^0,\mathbf{C}^0,\mathbf{W}^0)$. Thus the iterates of



(a) $\sigma = 0, p_0 = 0\%$ (b) $\sigma = 5, p_0 = 0\%$ (c) $\sigma = 10, p_0 = 0\%$

(d) $\sigma = 0, p_0 = 5\%$ (e) $\sigma = 5, p_0 = 5\%$ (f) $\sigma = 10, p_0 = 5\%$

(g) $\sigma = 0, p_0 = 10\%$ (h) $\sigma = 5, p_0 = 10\%$ (i) $\sigma = 10, p_0 = 10\%$

Figure C.2: Performance of inpainting (in SSIM) for BSD database.

ALS are uniformly bounded. Combining these observations with Theorem D.5, we conclude Theorem 4.1.

D.3.1. Establishing Single Values for $M_{\mathbf{W}}$.

Lemma D.8. *If the elements of $\mathbf{U}$ are i.i.d. samples from a continuous distribution on $[0, 1]$, then the $\lfloor p \cdot m \cdot n \rfloor$-th and $\lfloor p \cdot m \cdot n \rfloor + 1$-th greatest elements of $\{|(\mathbf{B}^t\mathbf{C}^t)_{ij} - \mathbf{X}_{ij}|^2 + \lambda_3 \mathbf{U}_{ij}^2\}_{\substack{1\le i\le m\\ 1\le j\le n}}$ are different with probability 1.*

Proof. Due to the independence and continuity of the distribution of $\mathbf{U}_{ij}$, we have

$$\begin{aligned}
&\mathbb{P}\ \{\ |(\mathbf{B}^t\mathbf{C}^t)_{ij} - \mathbf{X}_{ij}|^2 + \lambda_3 \mathbf{U}_{ij}^2 = |(\mathbf{B}^t\mathbf{C}^t)_{i'j'} - \mathbf{X}_{i'j'}|^2 + \lambda_3 \mathbf{U}_{i'j'}^2\ \} \\
&\quad = \mathbb{P}\{\lambda_3(\mathbf{U}_{i'j'}^2 - \mathbf{U}_{ij}^2) = |(\mathbf{B}^t\mathbf{C}^t)_{ij} - \mathbf{X}_{ij}|^2 - |(\mathbf{B}^t\mathbf{C}^t)_{i'j'} - \mathbf{X}_{i'j'}|^2\} \\
&\quad = 0
\end{aligned} \tag{D.2}$$

■

D.3.2. Establishing Single Values for $M_{\mathbf{B}}$ and $M_{\mathbf{C}}$.

Lemma D.9. *The infimal maps $M_{\mathbf{B}}$ and $M_{\mathbf{C}}$ derived from (4.1) are single-valued.*

Proof.

By the definition of infimal maps, we have

$$\hat{\mathbf{B}} := M_{\mathbf{B}}(\mathbf{C}, \mathbf{W}) = \arg\min_{\mathbf{B}} \|(\mathbf{B}\mathbf{C} - \mathbf{X}) \circ \mathbf{W}\|_F^2 + \lambda_1\|\mathbf{B}\|_F^2 + \lambda_2\|\mathbf{C}\|_F^2 \tag{D.3}$$

Let $\tilde{\mathbf{C}}_{(i)} = \mathbf{C} \circ (\mathbf{1}_{d\times 1}\mathbf{W}_{i\cdot})$ and $\tilde{\mathbf{X}} = \mathbf{X} \circ \mathbf{W}_{i\cdot}$. Then

$$\hat{\mathbf{B}}_{i\cdot} = \tilde{\mathbf{X}}_{i\cdot}\tilde{\mathbf{C}}_{(i)}^T(\tilde{\mathbf{C}}_{(i)}\tilde{\mathbf{C}}_{(i)}^T + \lambda_1\mathbf{I})^{-1}, \tag{D.4}$$

where $i = 1, \ldots, m$. Similarly, let $\hat{\mathbf{C}} = M_{\mathbf{C}}$ and we have

$$\hat{\mathbf{C}}_{\cdot j} = (\tilde{\mathbf{B}}_{(j)}^T\tilde{\mathbf{B}}_{(j)} + \lambda_2\mathbf{I})^{-1}\tilde{\mathbf{B}}_{(j)}^T\tilde{\mathbf{X}}_{\cdot j}, \tag{D.5}$$

where $\tilde{\mathbf{B}}_{(j)} = \mathbf{B} \circ (\mathbf{W}_{\cdot j}\mathbf{1}_{1\times d})$ and $j = 1, \ldots, n$. Then (D.4) and (D.5) complete the proof. ■

D.4. Conclusion of Theorem 4.2.

The K-ALS algorithm can also be viewed as a point-to-set map:

$$\Omega = (M_{\mathbf{W}} \circ (M_{\mathbf{B}} \circ M_{\mathbf{C}})^{t_1})^{t_2} \circ M_\eta \tag{D.6}$$

where $M_\eta = \arg\min_{\eta} J$, i.e.,

$$\eta(j) = \arg\min_k \min_{\mathbf{c}} \|(\mathbf{B}_k\mathbf{c} - \mathbf{X}_{\cdot j}) \circ \mathbf{W}_{\cdot j}\|_2^2 + \lambda_4\mathbf{V}_{kj}^2 \tag{D.7}$$

where $\mathbf{W} = \mathbf{P}[\mathbf{W}_1 \ldots \mathbf{W}_K]$, $\mathbf{P}$ is a permutation matrix such that $\mathbf{X} = \mathbf{P}[\mathbf{X}_1 \ldots \mathbf{X}_K]$.

The proof of Theorem 4.2 is analogous to Theorem 4.1. It suffices to show that (D.7) is single-valued. Because each element of $\mathbf{V}$ is i.i.d. sampled from a continuous distribution on [0, 1], we have:

$$\begin{aligned}
&\mathbb{P}\ \{\ \min_{\mathbf{c}} \|(\mathbf{B}_k\mathbf{c} - \mathbf{X}_{\cdot j}) \circ \mathbf{W}_{\cdot j}\|_2^2 + \lambda_4\mathbf{V}_{kj}^2 = \min_{\mathbf{c}} \|(\mathbf{B}_{k'}\mathbf{c} - \mathbf{X}_{\cdot j}) \circ \mathbf{W}_{\cdot j}\|_2^2 + \lambda_4\mathbf{V}_{k'j}^2\ \} \\
&\quad = \mathbb{P}\{\lambda_4(\mathbf{V}_{k'j}^2 - \mathbf{V}_{kj}^2) = \|(\mathbf{B}_k\mathbf{c} - \mathbf{X}_{\cdot j}) \circ \mathbf{W}_{\cdot j}\|_2^2 - \|(\mathbf{B}_{k'}\mathbf{c} - \mathbf{X}_{\cdot j}) \circ \mathbf{W}_{\cdot j}\|_2^2\} \\
&\quad = 0
\end{aligned} \tag{D.8}$$

Therefore, (D.7) is single-valued with probability 1 and the theorem is concluded.

REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein, k -svd: An algorithm for designing overcomplete dictionaries for sparse representation, Signal Processing, IEEE Transactions on, 54 (2006), pp. 4311 –4322.

[2] L. Balzano, B. Recht, and R. Nowak, *High-dimensional matched subspace detection when data are missing*, in Proceedings of the International Symposium on Information Theory, June 2010.

[3] L. Bottou and Y. Bengio, *Convergence properties of the k-means algorithms*, in Advances in Neural Information Processing Systems 7, MIT Press, 1995, pp. 585–592.

[4] P. Bradley and O. Mangasarian, *k-plane clustering*, J. Global optim., 16 (2000), pp. 23–32.

[5] A. Buades, B. Coll, and J. Morel, *A non-local algorithm for image denoising*, in Computer Vision and Pattern Recognition, 2005, pp. 60–65.

[6] A. M. Buchanan and A. W. Fitzgibbon, *Damped newton algorithms for matrix factorization with missing data*, in in CVPR05, 2005, pp. 316–322.

[7] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, J. ACM, 58 (2011), p. 11.

[8] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, *Rank-sparsity incoherence for matrix decomposition*, Arxiv, 02139 (2009), pp. 1–24.

[9] R. R. Coifman and D. L. Donoho, *Translation-invariant de-noising*, in Wavelets and Statistics, Springer-Verlag, 1995, pp. 125–150.

[10] G. B. Dantzig, J.Folkman, and N. Shapiro, *On the continuity of the minimum set of continuous functions*, J. Math. Anal. Appl., 17 (1967), pp. 519–548.

[11] M. Elad and M. Aharon, *Image denoising via learned dictionaries and sparse representation*, in In CVPR, 2006, pp. 17–22.

[12] P. Favaro, R. Vidal, and A. Ravichandran, *A closed form solution to robust subspace estimation and clustering*, in CVPR, 2011.

[13] J. C. Fiorot and P. Huard, *Composition and union of general algorithms of optimization*, Point-to-Set Maps and Mathematical Programming, Mathematical Programming Studies, 10 (1979), pp. 69–85.

[14] M. Fischler and R. Bolles, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, Comm. of the ACM, 24 (1981), pp. 381–395.

[15] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman, *Clustering appearances of objects under varying illumination conditions*, in Proceedings of International Conference on Computer Vision and Pattern Recognition, vol. 1, 2003, pp. 11–18.

[16] N. Kambhatla and T. K. Leen, *Fast non-linear dimension reduction*, in Advances in Neural Information Processing Systems 6, Morgan Kaufmann, 1994, pp. 152–159.

[17] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*, Information Science and Statistics, Springer, 2007.

[18] G. Lerman and T. Zhang, *Robust recovery of multiple subspaces by geometric ℓ_p minimization.* Submitted April 2011. Available at http://arxiv.org/abs/1104.3770.

[19] 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*, preprint, (2009).

[20] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, *Robust recovery of subspace structures by low-rank representation.* Available at http://arxiv.org/abs/1010.2955.

[21] G. Liu, Z. Lin, and Y. Yu, *Robust subspace segmentation by low-rank representation*, in ICML, 2010.

[22] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in ICCV, vol. 2, July 2001, pp. 416–423.

[23] R. R. Meyer, *Sufficient conditions for the convergence of monotonic mathematical programming algorithms*, J. Comput. System Sci., 12 (1976), pp. 108–121.

[24] T. Okatani and K. Deguchi, *On the wiberg algorithm for matrix factorization in the presence of missing components*, Int. J. Comput. Vision, 72 (2007), pp. 329–337.

[25] S. Roweis, *EM algorithms for pca and sensible pca*, in Neural Information Processing Systems, 1997.

[26] S. Roweis and L. Saul, *Nonlinear dimensionality reduction by locally linear embedding*, Science, 290 (2000), pp. 2323–2326.

[27] Y. Shen, Z. Wen, and Y. Zhang, *Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization*, Tech. Report TR11-02, Rice University, January 2011.

[28] J. B. Tenenbaum, V. de Silva, and J. C. Langford, *A global geometric framework for nonlinear dimensionality reduction*, Science, 290 (2000), pp. 2319–2323.

[29] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle adjustment - a modern synthesis*, in Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, ICCV '99, London, UK, 2000, Springer-Verlag, pp. 298–372.

[30] J. Tropp, I. Dhillon, R. Heath, and T. Strohmer, *Designing structured tight frames via alternating projection*, IEEE Trans. Inform. Theory, (2003), pp. 188–209.

[31] P. Tseng, *Nearest q-flat to m points*, Journal of Optimization Theory and Applications, 105 (2000), pp. 249–252. 10.1023/A:1004678431677.

[32] R. Vidal and R. I. Hartley, *Motion segmentation with missing data using powerfactorization and gpca*, in Computer Vision and Pattern Recognition, 2004, pp. 310–316.

[33] M. B. Wakin, *Manifold-based signal recovery and parameter estimation from compressive measurements*, tech. report, 2008.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *Image quality assessment: From error visibility to structural similarity*, IEEE Trans. Image Process., 13 (2004), pp. 600–612.

[35] T. Wiberg, *Computation of principal components when data are missing*, Proc 2nd Symposium on Computational Statistics Berlin Germany, (1976), pp. 229–326.

[36] G. Yu, G. Sapiro, and S. Mallat, *Image modeling and enhancement via structured sparse model selection*, in IEEE International Conference on Image Processing (ICIP), 2010.

[37] W. I. Zangwill, *Nonlinear Programming: A Unified Approach*, Engle-wood Cliffs, NJ: Prentice-Hall, 1969.

[38] K. Zhao and Z. Zhang, *Successively alternate least square for low-rank matrix factorization with bounded missing data*, Comput. Vis. Image Underst., 114 (2010), pp. 1084–1096.