SIAM J. IMAGING SCIENCES
Vol. 6, No. 1, pp. 526–562

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 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

DOI. 10.1137/110843642

1. Introduction.

We consider the problem of denoising images corrupted with impulsive noise, and the problem of blind inpainting where the images have 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, 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 a method with state-of-the-art performance on 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 [13]. 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. One can learn both $\mathbf{A}$ and $\mathbf{D}$ and create a model for $\mathbf{X}$ that can be used for both denoising and inpainting.

[*]Received by the editors August 8, 2011; accepted for publication (in revised form) October 4, 2012; published electronically March 6, 2013. This work was supported by NSF grants DMS-08-11203, DMS-09-15064, and DMS-09-56072. An earlier version of this manuscript appeared as an IMA preprint in September 2011.
http://www.siam.org/journals/siims/6-1/84364.html

[†]School of Mathematics, University of Minnesota, Minneapolis, MN 55455 (wangx857@umn.edu, lerman@umn.edu). The work of the third author was partially supported by the IMA during 2010-2012.

[‡]Department of Mathematics, The City College of New York, New York, NY 10031 (aszlam@ccny.cuny.edu).

Yu, Sapiro, and Mallat [46] 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 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 [18, 5, 39, 15]. Given parameters q and K, this algorithm attempts to find K q-dimensional subspaces, represented by $m \times q$ (where $q < m$) orthogonal matrices $\mathbf{D}_1, \dots, \mathbf{D}_K$ (i.e., matrices with orthonormal columns), which minimize

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

Finding $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_K]$ can be thought of as a block structured dictionary learning problem, or as finding the best secant subspaces to the given data set. The block structure benefits the denoising by making the coding more stable [46].

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 q 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 [46] to images corrupted by impulsive noise (and perhaps with additional Gaussian noise) as well as to blind inpainting. In this framework, finding $\mathbf{D} = [\mathbf{D}_1, \dots, \mathbf{D}_K]$ with the clusters fixed requires a tool different from the SVD. In particular, we will need a "robust" low rank method, which allows large corruptions of a controlled percentage of the corresponding matrix, in order to obtain a robust sparse coding method. Recently, there has been much interest in developing robust low rank models for matrices corrupted this way [9, 8, 34]. Our goal is to locally use such a low rank model in a framework such as 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 we will use instead a greedy and fast method for the updates of the dictionary $\mathbf{D}$.

The major contributions of this paper are as follows:

1. We describe an 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 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. We also suggest a new algorithm based on the well-known framework of ALS (which was previously used for PCA with missing components).

It is clear that, in general, this is an impossible problem: 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.

2.1. Review of some recent RPCA algorithms. While RPCA in its broadest sense has a long history [17, 26], its intensive study with sparse corruption (as formulated above) started only recently with two parallel groundbreaking works by Candès et al. [8] and Chandrasekaran et al. [9]. Both proposed the following convex minimization for RPCA:

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

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. This minimization for RPCA is referred to by [8] as principal component pursuit (PCP).

Candès et al. [8] proved that if $\lambda = 1/\sqrt{n}$ and the data matrix can be represented as $\mathbf{X} = \mathbf{L}_0 + \mathbf{S}_0$, where $\mathbf{L}_0$ is low rank and "strongly incoherent" (as specified in [8, eqs. (1.2)–(1.3)]) and $\mathbf{S}_0$ is sparse enough with uniformly sampled nonzero elements, then the minimizer of (2.1) is $\mathbf{L}_0$ with overwhelming probability (of the uniform sampling). On the other hand, Chandrasekaran et al. [9] provided a condition for deterministic exact recovery for various distributions of corrupted elements, though it implied a stronger restriction on the fraction of corruption (see also [16] for weaker restrictions than those in [9]); clearly [9] could not fix λ for its more general setting.

Lin et al. [23] have implemented (2.1) via an augmented Lagrangian approach, which alternates between shrinkages of spaces and singular values. 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 to as PCP(capped) and implement in the supplemental material (a similar idea of capping appeared in [43] after the submission of this paper).

The PCP algorithm is designed only to deal with sparse errors and may not be stable to noise in all entries. For this purpose, Zhou et al. [49] proposed a stable PCP, which minimizes

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

A formally equivalent problem is

$$\min_{\mathbf{L},\mathbf{S}\in\mathbb{R}^{m\times n}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \text{ such that } \|\mathbf{X}-\mathbf{L}-\mathbf{S}\|_F \leq \varepsilon. \tag{2.3}$$

They proved a perturbation-type result for the stability of the solution of (2.3), assuming the common condition for robustness of the solution of (2.1) and small values of noise ε (equivalently, large sizes of ν in (2.2); however, there is no simple expression for the lower bound on ν in terms of the upper bound of ε). Other types of stability results appeared in [1] and [16]. However, in our setting of large noise (often larger than the signal when measured in the Frobenius norm), such perturbation results may not be practical. Various methods have recently been proposed for solving this kind of optimization (see e.g., [35]), though we are not aware of any online code for this.

Another approach for RPCA, low-rank matrix fitting (LMaFit) [34], uses only the last two terms of (2.2). That is, it minimizes the objective function

$$\min_{\mathbf{B},\mathbf{C},\mathbf{S}} \|\mathbf{S}+\mathbf{B}\mathbf{C}-\mathbf{X}\|_F^2 + \mu\|\mathbf{S}\|_1, \tag{2.4}$$

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. Similarly to stablePCP, the LMaFit model is expected to handle Gaussian noise well.

2.2. RPCA via an ALS 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 measure errors with some matrix norm. If we choose the Frobenius norm, following the notation above, we may get the problem

$$\min_{\mathbf{B},\mathbf{C},\mathbf{S}} \|\mathbf{S}+\mathbf{B}\mathbf{C}-\mathbf{X}\|_F^2 \quad \text{such that } \|\mathbf{S}\|_0 \le N_0. \tag{2.5}$$

Clearly, the LMAFit minimization of (2.4) is a relaxation of this problem.

We suggest a variant for the minimization of (2.5). We represent the set of corrupted indices by a matrix $\mathbf{W} \in \{0,1\}^{m\times n}$ with N_0 zeros (for indices of corrupted points) and $m\cdot n - N_0$ ones. We denote the Hadamard product by $\circ$, i.e., $(\mathbf{A}\circ\mathbf{B})_{ij} = \mathbf{A}_{ij}\mathbf{B}_{ij}$. Then, instead of minimizing (2.5), we may minimize the objective function $J(\mathbf{B},\mathbf{C},\mathbf{W}) = \|(\mathbf{B}\mathbf{C}-\mathbf{X})\circ\mathbf{W}\|_F^2$.

In order to obtain a well-conditioned optimization problem and thus stable solutions, we modify J by adding some regularization terms (depending on sufficiently small $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 > 0$, and $\mathbf{U} \in \mathbb{R}^{m\times n}$, whose elements are i.i.d. samples from a continuous distribution on $[0, 1]$) as follows:

$$J(\mathbf{B},\mathbf{C},\mathbf{W}) = \|(\mathbf{B}\mathbf{C}-\mathbf{X})\circ\mathbf{W}\|_F^2 + \lambda_1\|\mathbf{B}\|_F^2 + \lambda_2\|\mathbf{C}\|_F^2 + \lambda_3\|\mathbf{U}\circ\mathbf{W}\|_F^2. \tag{2.6}$$

The low rank approximation is $\mathbf{L} = \mathbf{B}\,\mathbf{C}$, and the sparse matrix is $\mathbf{S} = \mathbf{X} - \mathbf{L}$ (which is supported on the zeros of $\mathbf{W}$). This formulation suggests a simple algorithm, which we describe in Algorithm 1 and will henceforth refer to as ALS. It will be easier for us to explain the role of the various regularizations in (2.6) after clarifying this algorithm.

The idea of this algorithm is to iteratively minimize (2.6) alternating between minimization over $\mathbf{B}$ and $\mathbf{C}$ as well as estimating $\mathbf{W}$. The code has n_2 loops for re-evaluating $\mathbf{W}$ and n_1 loops for minimizing over $\mathbf{C}$ and $\mathbf{B}$.

We note that the first two regularization terms for J (that is, $\lambda_1\|\mathbf{B}\|_F^2$ and $\lambda_2\|\mathbf{C}\|_F^2$) ensure that the minimizers $\mathbf{B}$ and $\mathbf{C}$ (estimated throughout the iterates of Algorithm 1) are unique and stable. Formulas for these minimizers are provided later in (G.4) and (G.5) (when proving convergence of Algorithm 1). It is clear from these formulas that such regularization is also useful in improving the numerical stability of these solutions. The regularization parameter λ_3 ensures that the set of indices of the $\lfloor p\cdot m\cdot n\rfloor$ greatest elements

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 iterations; $\lambda_1, \lambda_2, \lambda_3 \geq 0$: regularization parameters.
Output: $\mathbf{B} \in \mathbb{R}^{m\times d}$, $\mathbf{C} \in \mathbb{R}^{d\times n}$, $\mathbf{W} \in \{0,1\}^{m\times n}$ with exactly $\lfloor p \cdot m \cdot n \rfloor$ zeros.
Initialization: $\mathbf{W} = \mathbf{1}$; $\mathbf{B} \in \mathbb{R}^{m\times d}$ with i.i.d. entries: $\mathbf{B}_{ij} \sim \mathcal{N}(0,1)$; $\mathbf{U} \in \mathbb{R}^{m\times n}$ with i.i.d. entries: $\mathbf{U}_{ij} \sim \text{Uniform}(0,1)$.
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
    $\mathbf{W}$ is 0 at the indices of the $\lfloor p \cdot m \cdot n \rfloor$ greatest elements of
    $\{|\mathbf{X}_{ij} - (\mathbf{BC})_{ij}|^2 + \lambda_3 \mathbf{U}_{ij}^2\}_{i=1}^{m}{}_{j=1}^{n}$, and 1 otherwise.
end for
```

of $\{|\mathbf{X}_{ij} - (\mathbf{BC})_{ij}|^2 + \lambda_3 \mathbf{U}_{ij}^2\}_{i=1}^{m}{}_{j=1}^{n}$ (computed in Algorithm 1) is uniquely determined. We later use this property for proving convergence of Algorithm 1 (see e.g., (G.2)). However, in practice, we have found no difference when setting $\lambda_3 = 0$ and arbitrarily choosing any set of $\lfloor p \cdot m \cdot n \rfloor$ greatest elements of $\{|\mathbf{X}_{ij} - (\mathbf{BC})_{ij}|^2\}$. We demonstrate later in Tables 3 and 4 the advantage of the overall regularization. We also demonstrate later in Figures 10 and 11 the stability of the algorithm (in terms of both speed and accuracy) with respect to (w.r.t.) a large logarithmic range of small values of the regularization parameters. We comment though that in all of these experiments we could have set $\lambda_3 = 0$ without changing the results (which is not true for λ_1 and λ_2). Nevertheless, we will use $\lambda_3 > 0$ in our implementation in order to demonstrate in practice the algorithm suggested by our theory.

Throughout all the experiments we fix the parameters of Algorithm 1 as $\lambda_1 = \lambda_2 = \lambda_3 = 10^{-10}$ and form $\mathbf{U}$ by drawing i.i.d. samples from a uniform distribution on [0, 1]. We also fix $n_1 = 1$. For the simulated data in Appendix A, 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 criteria 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}. 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. More specifically, the results of ALS are robust to overestimation of p and d but not robust to underestimation (see Appendix B). In Appendix C, we propose and test an estimation strategy for d. We demonstrate results on artificial data in Appendix A, while in the main text we focus on the application of ALS within a hybrid linear modeling approach to both impulsive denoising and blind painting.

ALS algorithms for PCA have a long history and have been widely used for matrix completion, i.e., when the locations of the corrupted elements are specified, see, for example, [44, 37, 40, 7, 31, 29, 48]. In fact, the ALS algorithm for matrix completion is practically obtained by fixing the matrix $\mathbf{W}$ according to known and missing locations (1's and 0's of this matrix) and performing a restricted version of the ALS algorithm of this paper.

2.3. Why a new RPCA algorithm? The reader may wonder why we use an ALS method rather than the convex method of [9, 8] or its stable version [49], which have theoretical guarantees. We will use RPCA in a K-subspaces framework for modeling the underlying data by multiple subspaces. One cannot expect a fully convex formulation for such a setting. Indeed, Lerman and Zhang [21, section 5.1] have shown that it is impossible to generalize convex strategies as in [9, 8] for recovering multiple subspaces by energy minimization. Moreover, strategies such as [25, 24], which generalize [9, 8] to obtain an affinity matrix via a convex procedure (and then apply the nonconvex spectral clustering), are extremely slow for our application (see section 5).

We were also unable to successfully apply [9, 8] (or its stable version) within a K-subspaces algorithm because it is not obvious how to extend the nuclear norm out-of-sample (see section 3.2). In particular, we find it difficult to prove weak convergence theorems for the naive K-subspaces versions of PCP, unlike ALS (see section 4.4), due to the lack of monotonicity. While [34] naturally generalizes to a K-subspace algorithm, our experimental results using the authors' code in the setting of image denoising have not been successful.

Finally, and in some sense, most importantly, in our image denoising application, we have a good initialization. One of the complaints with nonconvex methods such as ALS or K-subspaces is the dependence on the initialization for good performance. But the initialization of the dictionary as in [46] has shown to be effective for image denoising. Here we follow this initialization for both denoising and blind inpainting (see section 5.2). Because we have such a good idea of where to start, dependence on initialization is actually a feature and not a bug. For example, this initialization (when compared to a random one) gains an improvement up to 4dB for denoising and up to 9dB for blind inpainting when using the images in Tables 1 and 2.

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 [36, 32, 20] 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 K-ALS algorithm aims to minimize the following regularized objective function:

$$\begin{aligned}(3.1)\qquad & J(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathbf{W}_k\}_{k=1}^K, \eta) \\ &= \sum_{k=1}^{K} (\|(\mathbf{B}_k\mathbf{C}_k - \mathbf{X}_k) \circ \mathbf{W}_k\|_F^2 + \lambda_1 \|\mathbf{B}_k\|_F^2 + \lambda_2\|\mathbf{C}_k\|_F^2 + \lambda_3 \|\mathbf{U}_k \circ \mathbf{W}_k\|_F^2) + \lambda_4 \sum_{j=1}^{n} v_{\eta(j)}^2,\end{aligned}$$

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 $\eta(j) = k$, $\mathbf{W}_k \in \{0,1\}^{m\times n_k}$, $\mathbf{U}_k \in \mathbb{R}^{m\times n_k}$, and both $\{\mathbf{U}_k\}_{k=1}^K$ and $(v_1, \ldots, v_K)$ are initialized by i.i.d. samples from a continuous distribution on [0, 1].

In practice, K-ALS uses Lloyd's algorithm as follows. The input parameters of K-ALS are the number of subspaces K, a rank d (though mixed ranks for the different subspaces are also possible), a percentage of corrupted entries p, regularization parameters $\lambda_1, \dots, \lambda_4$ (which we clarify at the end of this section), $\{\mathbf{U}_k\}_{k=1}^K$, $(v_1, \dots, v_K)$, and number of iterations n_1, n_2, and n_3. The algorithm first initializes a list of subspaces (details of the initialization for the imaging problems of this paper are discussed in section 5) and the matrix $\mathbf{W}$ (with indices 0 for estimated corrupted coordinates and 1 for uncorrupted). The algorithm then iteratively repeats the following two steps n_3 times. The first step finds the nearest subspace to each data point, ignoring the entries currently suspected to be corrupted. That is, given a data point $\mathbf{x} \in \mathbb{R}^m$ (i.e., column of the matrix $\mathbf{X}$), let $\overline{\mathbf{x}}$ denote the suspected uncorrupted entries in $\mathbf{x}$. Furthermore, for each $1 \le i \le K$ and each such $\overline{\mathbf{x}}$ (obtained from a column of $\mathbf{X}$), let $\overline{\mathbf{B}}_i \equiv \overline{\mathbf{B}}_i(\overline{\mathbf{x}})$ be the $\dim(\overline{\mathbf{x}}) \times d$ matrix formed 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

$$j = \operatorname*{arg\,min}_{1 \le i \le K} \min_{\mathbf{c} \in \mathbb{R}^d} \left(\|\overline{\mathbf{B}}_i \mathbf{c} - \overline{\mathbf{x}}\|_2^2 + \lambda_4 v_i^2 \right). \tag{3.2}$$

The second step computes a new basis for each cluster of points assigned to a given subspace of the first step by the ALS procedure of section 2.

The regularization parameters λ_1, λ_2, and λ_3 are used for the ALS algorithm within each cluster, and their role has already been discussed in section 2.2. The parameter λ_4 ensures that the cluster memberships (which are determined by (3.2)) are uniquely assigned; this is clarified later in (G.7) and (G.8) when proving convergence of the K-ALS algorithm. In practice, we have found the regularization term $\lambda_4 \sum_{j=1}^n v_{\eta(j)}^2$ unnecessary, though we keep it here so that our experiments correspond to the later theoretical analysis. Throughout all the experiments, $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 10^{-10}$, though the results are hardly different when setting $\lambda_3 = \lambda_4 = 0$.

Our implementation of the K-ALS algorithm (as well as the ALS algorithm) is available at http://math.umn.edu/∼lerman/kals.

3.2. Other possibilities of K-RPCA algorithms.

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 $\mathbf{B}_1, \dots, \mathbf{B}_K$ are the orthonormal bases of the d-dimensional subspaces representing the clusters, then we associate to $\mathbf{x}$ the subspace $\mathbf{B}_j$, where

$$j = \operatorname*{arg\,min}_{1 \le i \le K} \min_{\mathbf{c} \in \mathbb{R}^d} \|\mathbf{x} - \mathbf{B}_i \mathbf{c}\|_1. \tag{3.3}$$

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 $\mathbf{B}_1, \dots, \mathbf{B}_K$ are the orthonormal bases of the d-dimensional subspaces representing the clusters, then we associate to $\mathbf{x}$ the subspace $\mathbf{B}_j$, where

$$j = \operatorname*{arg\,min}_{1 \le i \le K} \min_{\mathbf{c} \in \mathbb{R}^d, \mathbf{e} \in \mathbb{R}^m} \left(\|\mathbf{e} + \mathbf{B}_i \mathbf{c} - \mathbf{x}\|_F^2 + \mu \|\mathbf{e}\|_1 \right). \tag{3.4}$$

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 [10]; in this context, it is simply the idea that the charts of a manifold (which may be part of a union of manifolds) should overlap. If the final goal is denoising, then we average the outputs of all repetitions.

For the image data in section 5 we set n_2 from Algorithm 1 to be 1 and consequently speed up the algorithm.

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 the (regularized) ALS and K-ALS algorithms.

4.1. Preliminaries. We associate to both the ALS algorithm and the K-ALS algorithm mappings from their inputs to their possible outputs. For the ALS algorithm each iteration at time t computes the values of a point $(\mathbf{B}^t, \mathbf{C}^t, \mathbf{W}^t)$. A domain for this point is $\mathcal{X} = \mathbb{R}^{m\times d} \times \mathbb{R}^{d\times n} \times \{0,1\}^{m\times n}$. For the K-ALS algorithm we denote for simplicity a point $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathbf{W}_k\}_{k=1}^K, \eta)$ by Ω. We also denote this point at iteration t by Ω^t. The domain $\mathcal{X}$ for points Ω is $\mathcal{X} = (\mathbb{R}^{m\times d})^K \times (\mathbb{R}^{d\times n})^K \times (\{0,1\}^{m\times n})^K \times \{1,\dots,K\}^n$. We describe each algorithm as a mapping ϕ from $\mathcal{X}$ to the partition set of $\mathcal{X}$, $\mathcal{P}(\mathcal{X})$ (including several possible results as we have no a priori knowledge of convergence). A *fixed point* of ϕ is a point $\mathbf{x} \in \mathcal{X}$ for which $\{\mathbf{x}\} = \phi(\mathbf{x})$. We express convergence properties of both ALS and K-ALS in terms of fixed points of their corresponding mapping ϕ. The convergence theorems are formulated in probability since these algorithms contain randomly initialized regularization components.

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

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

4.3. Main theorem: Convergence of K-ALS. We establish the following result for convergence of the K-ALS algorithm for robust PCA.

Theorem 4.2. *The following statements for the K-ALS algorithm hold with probability* 1: (1) *All accumulation points of the iterates* $\{\Omega^t\}_{t=1}^\infty$ *produced by this algorithm are its fixed points;* (2) $J(\Omega^t) \to J(\Omega^*)$ *as* $t \to \infty$, *where* Ω^* *is a fixed point;* (3) $\|\Omega^t - \Omega^{t+1}\| \to 0$ *as* $t \to \infty$; *and* (4) *either* $\{\Omega^t\}_{t=1}^\infty$ *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 [23] are formulated only in terms of convergence of the objective function, as we have in the second part of our theoretical statements (but the authors of [23] 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 [4], whereas here we prove only 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 our formulation of 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 (BSD) [27]. 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 integers 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) [46], K-SVD [13, 2], low rank representation (LRR) [25, 24], PCP, PCP(capped), LMaFit, K-PCP, K-PCP(capped), K-LMaFit, and two variations (NL-Median1 and NL-Median2) of the nonlocal means method of [6]. When two methods are combined, we use “+” to connect them, e.g., MF+SSMS, which means we apply SSMS after MF.

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 peak signal-to-noise ratio (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 [46] 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 discrete cosine transform basis (our implementation is available online). Choosing K is a tradeoff between accuracy and speed. The larger K is, the more accurate, but slower, the algorithm is. Note that this initialization is important for good results.

In order to account for Gaussian noise, we apply additional steps in the spirit of [46] (to be consistent for all methods). We first recall that Yu, Sapiro, and Mallat [46] use 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 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 for $\mathbf{a} = (a_1, \ldots, a_{d'})$ and $1 \leq j \leq d'$, $(\delta(\mathbf{a}))_j = a_j$ if $|a_j| > 3\sigma$ and $= 0$ otherwise (i.e., SSMS assumes knowledge of the model parameter σ described in section 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 affect 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, med$(\mathbf{Y})$ the image obtained by applying the 4×4 median filter to $\mathbf{Y}$, $\tilde{\mathbf{Y}}$ the image translated from the patches of the current iteration (before thresholding), and 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). This update of $\mathbf{W}$ is very natural in the framework of K-ALS. Indeed, K-ALS has the advantage of using application-dependent information in the estimate of locations of corrupted elements.

For the global low rank model (in particular, PCP), we addressed Gaussian noise by further applying the SSMS algorithm [46] (so that 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. We did not notice an advantage of an implementation of (2.2) (which we later refer to as stablePCP in Appendix A) over PCP and did not report it since it required experimentation with the additional parameter ν.

We describe in section G.5 two robust variants of the nonlocal means algorithm [6]. We have also tested the median variant of the nonlocal means toolbox on MATLAB exchange [30]; however, its performance was not satisfying and is thus not reported here.

For LRR [25, 24], it is not feasible to assign the data matrix itself as the dictionary (as done in [25, 24]). We thus tried the following two options: the first was to use the initial basis used for K-ALS, and the second was to use the output basis learned by K-ALS. We found out that these two ways were comparable when we chose appropriate values for the regularization parameter λ. We thus studied only the first case. More precisely, we take the first $d' = 20$ vectors from each of the bases and form the dictionary by concatenating them in a matrix. Similarly to PCP, we address Gaussian noise by further applying the SSMS algorithm.

We fixed the parameters of K-ALS for all 105 images as follows: $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 10^{-10}$, $n_1 = n_2 = 1$, and $n_3 = 5$ (see section 3.1). For p, we tested both $p = p_0$ and $p = 2p_0$. 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 $\lambda = 10^k$, $k = -3, \dots, 3$. Similarly, for PCP(capped)+SSMS and K-PCP(capped), the parameters (λ, γ) were chosen to be $(0.1, 0.001)$ after testing $(10^k, 10^l)$, $k, l = -3, \dots, 3$. For LMaFit and K-LMaFit, we fine-tuned the best parameters based on PSNRs of the outputs. We first tested the 7 values $\{10^k\}_{k=-3}^3$ and then searched around the best one in a finer scale until the results were not improved.

For the parameter λ in LRR, we first noticed that $\lambda = 10^{-3}$ obtained the best results among $\{10^k\}_{k=-3}^3$. To obtain even better results we tested the following 7 values for λ: 0.0003, 0.0005, 0.0007, 0.001, 0.0015, 0.0022, and 0.0032 (corresponding to k in $[-3.5, -2.5]$). We chose $\lambda = 0.0015$ according to the best performance. As we explain below, we tested LRR only for the 5 common images because it was too slow to run on the whole database.

5.3. Results. For each image, method, and choice of corruption/denoising parameters, we averaged the PSNR over five simulations. Tables 1 and 2 report results of the different methods for denoising impulsive noise and blind inpainting, respectively, for the five popular images. We highlight the best results. Figures 1 and 2 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 slow (as is evident from Table 5); we thus ran PCP+SSMS and K-PCP only for these five common images. 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 reported only MF+SSMS among these. As for the nonlocal medians variants, which are described in section G.5, we report only NL-Median1 for blind inpainting and NL-Median2 for denoising since they are the best for these problems.

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 at the time of submitting the paper is not open source. LRR gives good results on the 5 popular images, but it is slow; for example, it takes about a minute for K-ALS to finish the *House* image and several hours for LRR. We were thus unable to test LRR on the large database.

In addition to PSNR, we also evaluated the image quality by the structural similarity (SSIM) index [42]. We report these results in Appendix D. 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 performances of K-ALS(p_0) and K-ALS($2p_0$) are more closely correlated to each other when using SSIM than PSNR.

Besides PSNR and SSIM, we also demonstrate the visual quality of some denoising and blind inpainting results in Figures 3 and 4, respectively. The visual quality is important. Indeed, Figure 5 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.

We observe that PCP+SSMS and K-PCP are not practical because of long running times. Moreover, PCP+SSMS does not successfully restore images in general, and K-PCP (with additional thresholding) also fails in some cases, e.g., the image *House*. PCP(capped)+SSMS

Table 1

Performance (in PSNR) of denoising for 5 popular images. Each row lists results of different approaches on a specific image corrupted by p_0 impulsive noise and Gaussian noise with standard deviation σ. The number in parentheses after K-ALS specifies the input parameter estimating the portion of corruptions. See section 5.3 for details on the parameters used in all the algorithms. The best results are in bold. K-ALS and LRR outperform other methods for many of the cases; however, LRR is nearly two orders of magnitude slower than K-ALS.

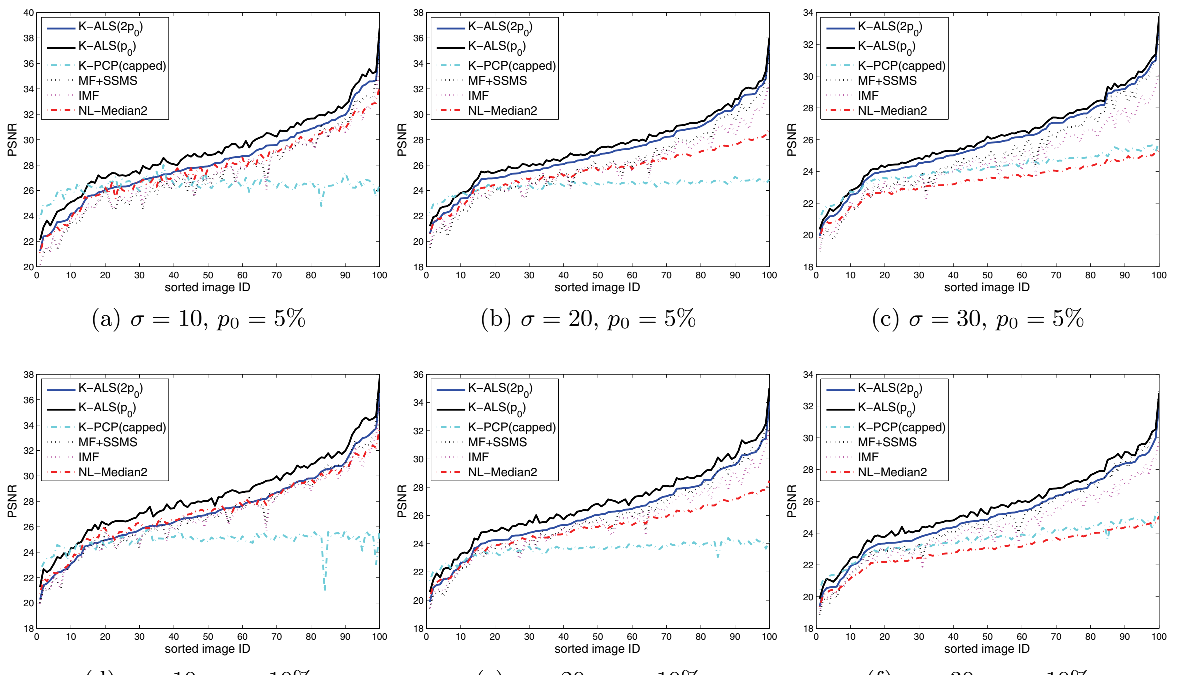
	p_0	σ	K-ALS($2p_0$)	K-ALS(p_0)	PCP(capped)+SSMS	K-PCP(capped)	KSVD	SSMS	MF+KSVD	MF+SSMS	IMF	NL-Median2	PCP+SSMS	K-PCP	LRR
Barbara	5%	10	30.77	**31.84**	26.35	26.61	22.06	22.81	24.74	24.96	24.63	29.73	24.11	29.18	31.59
		20	28.68	**29.22**	24.07	24.61	22.06	24.98	23.81	24.24	23.84	26.66	25.38	27.34	29.15
		30	26.99	**27.38**	23.56	24.55	23.16	26.22	22.93	23.67	23.05	23.96	26.19	25.67	27.38
	10%	10	29.30	**30.84**	24.34	25.31	19.26	19.59	24.59	24.77	24.35	29.20	19.77	28.45	28.35
		20	27.55	**28.64**	23.18	23.75	19.58	21.82	23.71	24.08	23.61	26.09	21.91	26.70	28.27
		30	25.99	26.63	22.99	23.93	21.00	23.85	22.81	23.52	22.89	23.47	23.96	25.20	**27.27**
Boat	5%	10	30.74	31.49	26.11	26.50	22.43	23.32	28.50	29.10	28.95	29.63	24.61	29.05	**31.71**
		20	28.88	**29.30**	24.51	24.55	22.54	25.46	26.55	27.46	27.06	26.69	25.94	27.43	29.15
		30	27.38	**27.69**	24.43	24.63	23.97	26.63	25.05	26.34	25.58	24.10	26.67	26.00	27.64
	10%	10	29.70	**30.84**	24.52	25.30	19.76	20.15	28.29	28.84	28.42	29.13	20.43	28.52	29.39
		20	27.99	**28.69**	23.74	23.79	20.55	22.55	26.39	27.23	26.69	26.21	22.67	26.98	27.86
		30	26.58	**27.09**	23.98	24.10	21.85	24.56	24.86	26.10	25.26	23.66	24.55	25.61	26.57
House	5%	10	34.15	**34.84**	26.11	28.15	22.42	23.51	31.85	32.16	31.09	31.85	33.90	28.26	28.52
		20	31.76	**32.17**	24.32	26.17	22.90	25.87	29.81	30.13	28.60	27.89	31.38	27.14	27.10
		30	29.79	**30.10**	24.80	26.43	24.92	28.11	27.24	28.81	26.93	24.81	30.23	25.99	25.88
	10%	10	32.96	**34.12**	24.75	26.66	19.75	20.25	31.27	31.52	30.42	31.41	28.75	27.74	28.17
		20	30.55	**31.32**	23.67	25.40	20.05	22.83	29.43	29.76	28.12	27.42	28.69	25.73	26.75
		30	28.82	**29.43**	24.35	25.71	22.34	25.55	26.93	28.38	26.47	24.35	28.16	25.61	25.54
Lena	5%	10	33.47	**34.19**	25.63	26.31	22.40	23.36	31.75	32.49	31.71	31.86	25.33	31.23	34.01
		20	31.29	**31.69**	23.92	24.53	22.76	25.62	29.28	30.40	29.17	27.81	26.36	29.39	31.31
		30	29.54	**29.85**	24.64	25.09	24.31	27.95	27.37	28.95	27.53	24.76	27.96	28.02	29.57
	10%	10	32.35	**33.49**	24.16	25.23	19.74	20.15	31.48	32.11	31.22	31.39	20.40	30.74	31.29
		20	30.20	**30.97**	23.39	23.85	20.01	22.61	29.07	30.11	28.82	27.34	22.56	28.96	29.52
		30	28.55	**29.15**	24.17	24.52	21.73	25.30	27.10	28.60	27.18	24.35	25.39	27.54	28.24
Peppers	5%	10	32.81	**33.44**	25.22	26.20	22.05	22.88	32.05	32.55	31.89	31.55	24.85	30.01	32.68
		20	31.01	**31.42**	24.16	24.54	22.32	24.94	30.03	30.64	29.53	27.71	25.87	28.75	30.95
		30	29.48	**29.82**	24.73	25.01	23.73	27.50	28.15	29.25	27.91	24.75	27.68	27.83	29.40
	10%	10	31.66	**32.81**	23.92	25.13	19.36	19.72	31.65	32.08	31.31	30.94	20.01	29.54	30.55
		20	29.88	**30.74**	23.40	23.78	19.57	21.96	29.69	30.24	29.10	27.20	21.99	28.37	29.15
		30	28.41	**29.07**	24.18	24.40	20.97	24.62	27.75	28.81	27.47	24.32	24.76	27.15	28.05

and K-PCP(capped) are fast but do not work well in general. LMaFit and K-LMaFit failed in general. The variants of the nonlocal medians cannot handle Gaussian noise well. The best of them is comparable to K-ALS when the Gaussian noise is low, but is clearly worse as it increases. It is interesting to note that these variants are often effective with texture (even with noise), but do not perform well on smooth regions with noise. On the other hand, methods based on the median filter (e.g., MF+SSMS and IMF) tend to oversmooth the image and thus cannot handle textures well. For some blindly inpainted images, PCP(capped) obtained high PSNR, but its perceptual quality was not good; that is, it was not able to remove many

Table 2

Performance (in PSNR) of blind inpainting for 5 popular images. Each line lists results of different approaches on a specific image corrupted by σ Gaussian noise, p_0 impulsive noise, and some scratches. The number in parentheses after K-ALS specifies the input parameter estimating the portion of corruptions. See section 5.3 for details on the parameters used in all the algorithms. The best results are in bold. K-ALS and LRR outperform other methods for most images except Peppers, which is relatively smooth (K-ALS results improve when we reduce the dimension and the number of subspaces); however, LRR is nearly two orders of magnitude slower than K-ALS.

	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	PCP+SSMS	K-PCP	LRR
		0	32.26	**32.97**	29.41	29.00	27.33	27.33	25.22	25.22	25.22	26.75	28.53	29.99	31.99
	0%	5	31.32	**32.23**	28.49	28.31	27.09	27.10	25.15	25.23	25.06	26.60	27.95	29.25	31.09
		10	30.41	**30.90**	26.70	26.42	26.85	26.97	24.72	24.96	24.69	26.04	27.40	28.51	29.95
		0	29.67	**31.60**	27.84	27.11	20.77	20.76	24.91	24.91	24.91	26.21	21.53	29.15	30.51
Barbara	5%	5	29.30	**31.02**	27.23	26.68	20.96	21.00	24.90	24.99	24.77	26.05	21.72	28.56	30.06
		10	28.98	**29.97**	25.53	25.16	21.07	21.66	24.58	24.78	24.42	25.53	22.37	27.92	29.19
		0	25.43	**30.45**	25.74	25.32	18.33	18.33	24.56	24.54	24.55	25.68	18.44	28.40	29.12
	10%	5	26.05	**30.03**	25.34	25.12	18.44	18.43	24.62	24.70	24.46	25.53	18.53	27.93	28.93
		10	26.81	**28.95**	24.06	24.21	18.68	18.98	24.34	24.53	24.18	25.05	19.08	27.28	28.32
		0	30.67	32.80	29.76	28.63	26.96	26.96	30.31	30.31	30.31	29.54	28.88	30.65	**33.27**
	0%	5	30.25	**31.84**	28.93	28.07	26.67	26.66	29.73	29.95	29.87	29.25	27.91	29.46	31.49
		10	29.59	**30.45**	26.48	26.15	26.40	26.51	28.41	29.01	28.89	28.57	27.08	28.58	30.17
		0	29.66	31.65	27.59	26.93	21.06	21.05	29.74	29.72	29.72	29.24	21.92	29.37	**32.26**
Boat	5%	5	29.53	30.70	26.95	26.52	21.16	21.19	29.32	29.55	29.32	29.02	22.01	28.55	**31.01**
		10	28.92	29.65	25.32	25.01	21.35	21.98	28.14	28.72	28.41	28.26	22.69	28.00	**29.76**
		0	28.00	30.88	25.57	25.42	18.66	18.66	28.84	28.94	28.95	28.90	18.83	28.75	**31.07**
	10%	5	28.11	30.02	25.06	25.11	18.82	18.81	28.77	28.98	28.69	28.59	18.95	28.05	**30.25**
		10	27.99	**29.30**	23.94	24.11	19.08	19.42	27.74	28.31	28.00	27.92	19.63	27.50	29.09
		0	**36.44**	35.89	27.35	25.66	23.67	23.67	31.11	31.10	32.05	31.44	33.64	29.44	30.05
	0%	5	**33.93**	33.63	27.02	25.49	23.66	23.66	30.89	31.05	31.57	30.37	31.38	29.10	29.28
		10	**32.19**	31.42	25.71	24.84	23.71	23.83	30.08	30.52	30.54	29.51	29.63	28.38	28.55
		0	35.41	**36.58**	26.38	24.65	20.00	20.00	29.98	29.98	31.40	30.81	28.95	28.39	29.53
House	5%	5	33.29	**33.90**	25.81	24.62	20.08	20.12	29.99	30.06	30.92	30.14	27.59	28.52	28.84
		10	31.79	**31.54**	24.54	24.26	20.35	20.89	29.16	29.75	30.00	29.03	27.15	28.21	28.24
		0	32.67	**35.80**	24.58	23.63	18.02	18.02	28.43	28.53	30.48	30.19	22.67	27.93	29.19
	10%	5	32.15	**33.67**	24.26	23.69	18.16	18.16	28.94	28.74	30.23	29.71	22.36	28.85	28.39
		10	31.20	**32.23**	23.60	23.56	18.47	18.80	28.60	28.70	29.39	28.54	23.16	27.94	27.92
		0	35.32	**36.65**	30.94	29.23	27.72	27.72	34.38	34.38	34.38	33.11	30.90	32.81	36.48
	0%	5	34.15	**34.97**	29.72	28.69	27.49	27.50	33.24	33.54	33.33	32.64	29.73	31.76	34.36
		10	32.84	**33.03**	26.26	26.62	27.36	27.47	31.49	32.29	31.70	31.30	28.75	30.82	32.88
		0	33.71	**35.46**	27.99	27.00	21.19	21.19	33.50	33.44	33.44	32.78	22.46	31.71	35.31
Lena	5%	5	33.10	**34.29**	26.87	26.94	21.37	21.43	32.73	32.94	32.55	32.33	22.54	30.94	33.81
		10	32.15	**32.77**	24.80	25.39	21.45	22.12	31.14	31.89	31.28	30.90	23.23	30.25	32.43
		0	31.24	**34.81**	25.45	25.55	18.72	18.72	32.00	32.18	32.56	32.44	18.90	31.15	34.20
	10%	5	31.73	**33.61**	24.92	25.57	18.84	18.85	31.91	32.19	32.02	31.93	19.05	30.50	33.09
		10	31.17	**32.22**	23.76	24.45	19.17	19.51	30.57	31.29	30.75	30.49	19.75	29.75	31.69
		0	29.22	31.26	31.36	29.23	27.68	27.67	**34.22**	**34.22**	**34.22**	32.93	30.97	32.08	32.58
	0%	5	28.66	30.24	29.19	28.68	27.38	27.37	33.22	**33.46**	33.34	32.58	29.78	30.98	31.51
		10	29.50	30.85	27.62	26.80	20.93	20.93	31.89	**32.40**	32.00	31.28	28.74	30.23	31.12
		0	29.50	30.85	27.62	26.80	20.93	20.93	33.31	33.25	**33.45**	32.48	22.21	31.31	32.08
Peppers	5%	5	29.07	30.64	26.49	26.85	21.04	21.09	32.68	**32.89**	32.80	32.07	22.42	30.37	31.34
		10	29.09	30.40	24.60	25.38	21.16	21.74	31.45	**31.92**	31.50	30.79	23.02	29.75	30.86
		0	28.54	31.09	25.12	25.32	18.43	18.42	31.85	31.99	**32.69**	32.12	18.69	30.65	31.41
	10%	5	29.07	30.14	24.60	25.35	18.55	18.55	31.74	31.96	**32.17**	31.62	18.69	29.86	30.98
		10	28.70	29.98	23.62	24.40	18.80	19.11	30.74	**31.26**	30.95	30.37	19.36	29.14	30.40



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

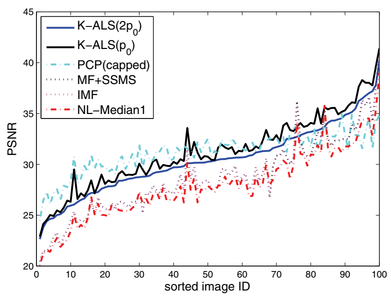
(d) $\sigma = 10$, $p_0 = 10\%$ (e) $\sigma = 20$, $p_0 = 10\%$ (f) $\sigma = 30$, $p_0 = 10\%$

Figure 1. *Performance (in PSNR) of denoising for the BSD database* [27]. *In each subfigure, PSNR of a specific corrupted scenario (see the subcaptions) is plotted versus the sorted image IDs. The number in parentheses after K-ALS specifies the input estimating the portion of corruptions. Other methods use the best values from a broad range of parameters; see section* 5.3. *For most of the cases, K-ALS outperforms the other algorithms, especially when the Gaussian noise is high.*

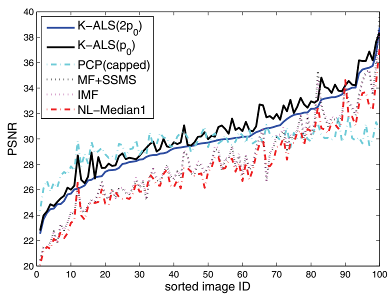
scratches. Of all the methods we tested, LRR was the only one that could compete with K-ALS in terms of quality. However, it is very slow, costing two orders of magnitude more computation time. Even when significantly overestimating its main parameter p, K-ALS performs very well; in particular, it is good for various scenarios including texture, noisy smooth regions, and scratches.

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 an ALS 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 [41].

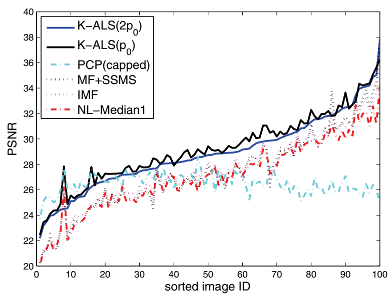
Appendix A. Numerical simulations with a single subspace. To evaluate the performance of ALS in comparison to other RPCA methods, several simulations are conducted. We distinguish here and in what follows between p_0, the ground truth percentage of sparse corruptions (i.e., percentage of impulsive noise), and p, the input parameter of ALS estimating



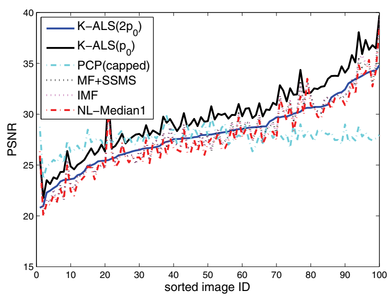
(a) $\sigma = 0$, $p_0 = 0\%$



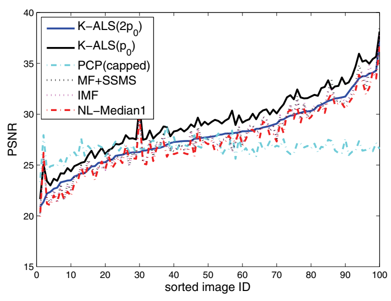
(b) $\sigma = 5$, $p_0 = 0\%$



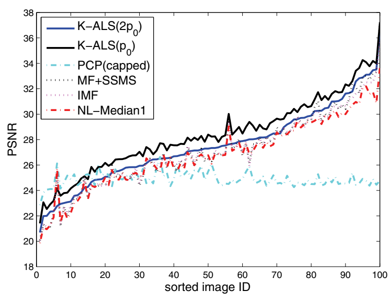
(c) $\sigma = 10$, $p_0 = 0\%$



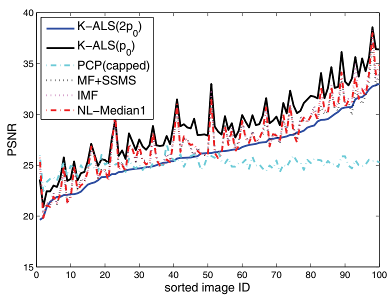
(d) $\sigma = 0$, $p_0 = 5\%$



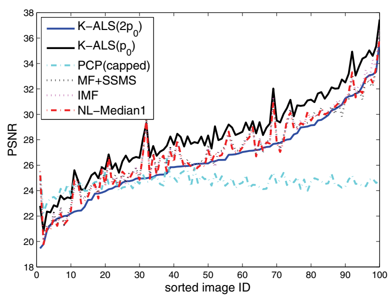
(e) $\sigma = 5$, $p_0 = 5\%$



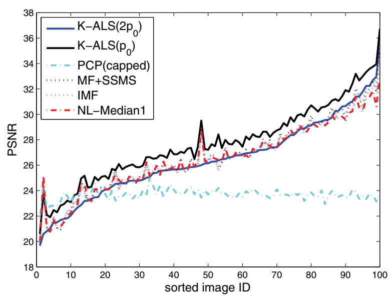
(f) $\sigma = 10$, $p_0 = 5\%$



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



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



(i) $\sigma = 10$, $p_0 = 10\%$

Figure 2. *Performance (in PSNR) of blind inpainting for the BSD database* [27]. *In each subfigure, PSNR of a specific corrupted scenario is plotted versus the sorted image IDs (the impulsive and Gaussian noise parameters are in the subcaptions, and the scratch used throughout this paper is also added). The number in parentheses after K-ALS specifies the input estimating the portion of corruptions. Other methods use the best values from a broad range of parameters; see section* 5.3. *In most cases K-ALS(p_0) outperforms other algorithms. However, K-ALS($2p_0$) is advantageous only with low levels of corruption. For a few images the PSNR values of PCP(capped) are high, but their perceptual quality is not as satisfying.*

the percentage of corruption, and between 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. Due to the randomness, we repeat the experiment 100 times. 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 there is no Gaussian noise). Finally, p_0 randomly chosen pixels

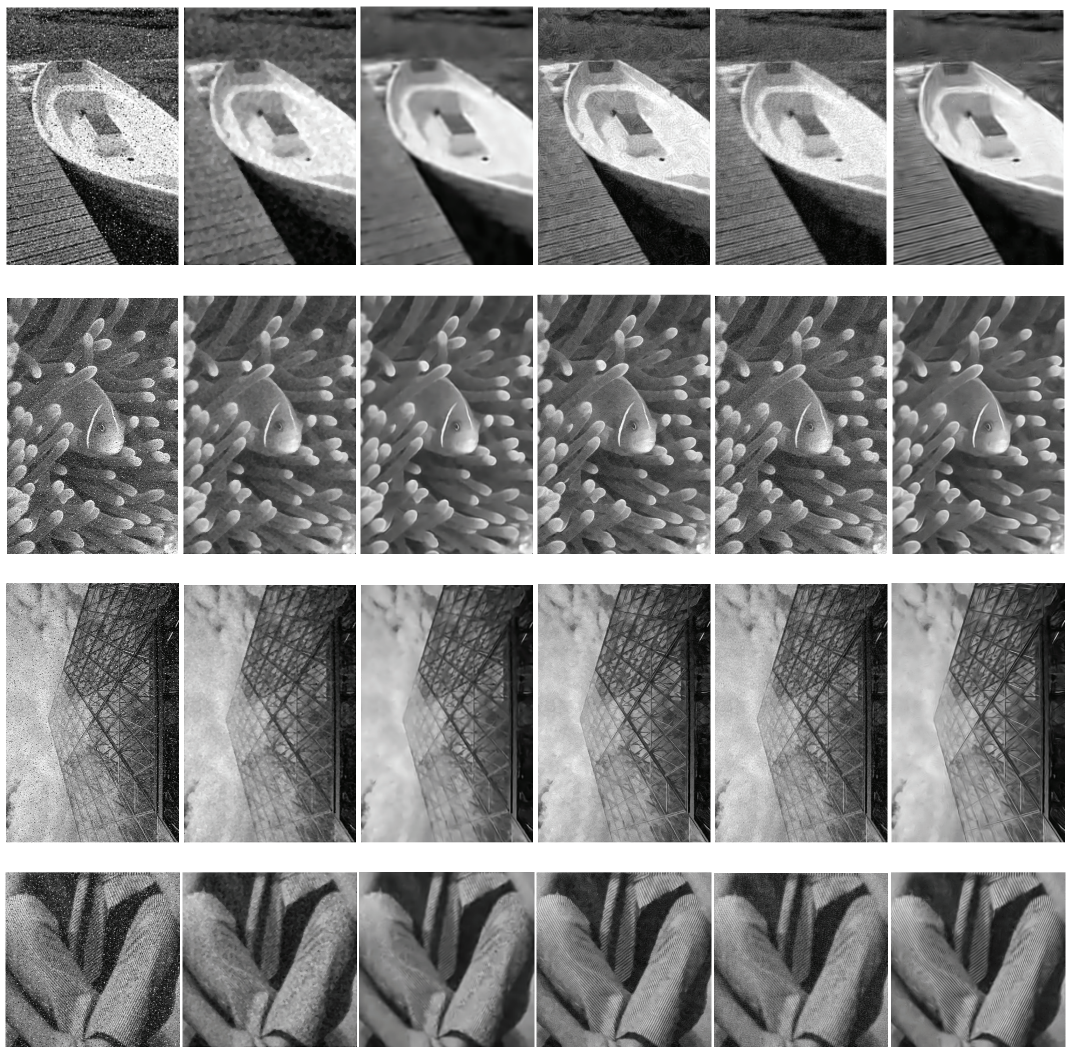


Figure 3. *From left to right: noisy images, IMF, MF+SSMS, NL-Median2, K-PCP(capped), and K-ALS($2p_0$). The additive Gaussian noise has standard deviation $\sigma = 30$ for the first image and $\sigma = 20$ for the other three. The first three images have 5% impulsive noise, and the last one has 10%. K-ALS, even with overestimated $p = 2p_0$, performs very well perceptually.*

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), stablePCP, stablePCP(capped), LMaFit, ALS, and $\text{ALS}_{\text{unreg}}$, where $\text{ALS}_{\text{unreg}}$ is the ALS algorithm without regularization, i.e., $\lambda_1 = \lambda_2 = \lambda_3 = 0$. All of our implementations appear in supplementary material available at http://math.umn.edu/~lerman/kals. For PCP, we use the fast implementation suggested in [23] with $\lambda = (mn)^{-0.25}$. We recall that PCP(capped) is the version of PCP mentioned in section 2, which requires an upper bound, i.e., a cap, on the intrinsic dimension. Since we are not aware of any available code for minimizing (2.2), we have

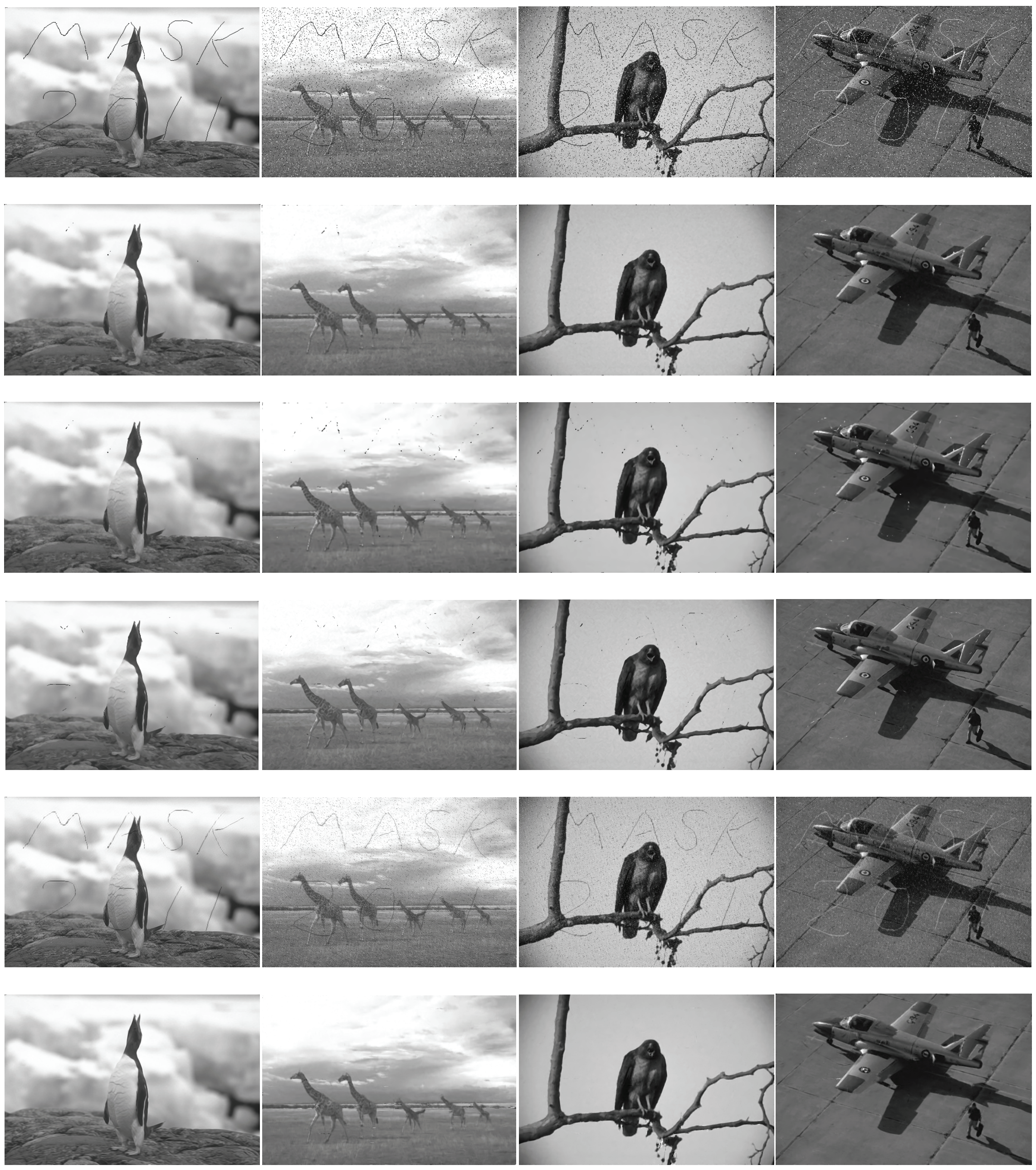


Figure 4. *From top to bottom: corrupted images, IMF, MF+SSMS, NL-Median1, PCP(capped)+SSMS, and K-ALS($2p_0$). The first image is corrupted only by scratches. The rest also have 5% impulsive noise and $\sigma = 10$ Gaussian noise. K-ALS, even with overestimated $p = 2p_0$, completely removes the scratches and still preserves image quality.*

implemented it by simple alternation (solving for each variable separately) and refer to this implementation as stablePCP. Its capped version is similar to that of PCP (but does not require the parameter γ).

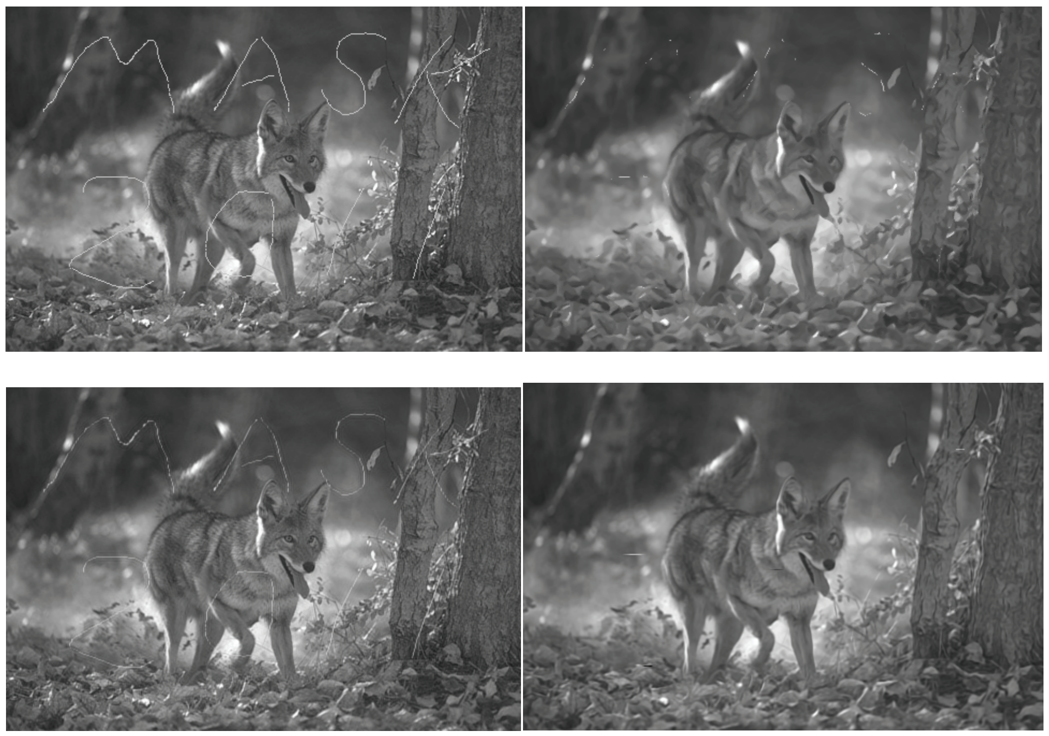


Figure 5. *Demonstration of the importance of visual quality. From left to right, top to bottom: image corrupted by scratches only (no Gaussian noise or impulsive noise), outputs of NL-Median1 (PSNR* $= 29.51$*, SSIM* $= 0.8637$*), PCP(capped)+SSMS (PSNR* $= 32.66$*, SSIM* $= 0.9425$*), and ALS(*$2p_0$*) (PSNR* $= 31.70$*, SSIM* $= 0.9425$*), where ALS(*$2p_0$*) uses double the true percentage of corruptions, while the other methods use the best values for their parameters among a fairly broad range. Although the PSNR and SSIM are high for the third image, its perceptual quality is not as good as the second and the fourth images. The second image is not as good as the fourth one due to unremoved scratches and oversmoothing.*

For ALS, PCP(capped), stablePCP(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 Appendix B). For PCP(capped) we input in addition to λ of (2.1) a parameter γ, which is present in every ADMM (alternating direction method of multipliers) implementation for PCP (see supplemental code at http://math.umn.edu/∼lerman/kals). We tested $(\lambda, \gamma) = (10^k, 10^l)$ for $k, l = -3, \dots, 3$ 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 stable(PCP) and stablePCP(capped), we tested $(\lambda, \nu) = (10^k, 10^l)$ for $k, l = -3, \dots, 3$ and selected $(\lambda, \nu) = (0.1, 100)$ (thus stable(PCP) and stablePCP(capped) are also of oracle type). For LMaFit, we chose $\mu = 10^3$ by minimizing the true fitting error among $\mu = 10^k$, $k = -3, \dots, 5$. We thus refer to the method as LMaFit(oracle). The parameters for ALS were specified in section 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:

$$e = \frac{\|\tilde{\mathbf{X}} - \mathbf{X}_0\|_F}{\|\mathbf{X}_0\|_F}, \tag{A.1}$$

Table 3

Mean relative fitting error (in percentage) on simulations without Gaussian noise. The random data matrix (generated 100 *times) is* $m \times 1000$*, and its underlying rank is* d_0*. It is corrupted by* p_0 *sparse noise (random values).*

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
stablePCP	1.48	1.96	5.84	1.58	1.99	13.83	1.81	3.52	79.37	2.70	53.66	132.42
stablePCP(capped	1.44	1.87	4.88	1.59	2.00	3.41	1.80	2.51	5.16	2.18	3.30	6.43
ALS$_{\text{unreg}}$($2p_0$)	1.08	4.86	29.26	0.16	2.30	19.25	0.08	0.31	6.89	**0.00**	0.20	1.74
ALS$_{\text{unreg}}$(p_0)	2.71	7.06	31.89	0.14	0.68	9.14	**0.00**	0.03	0.76	**0.00**	**0.00**	0.05

Table 4

Mean relative fitting error (in percentage) on simulations with Gaussian noise ($\sigma = 0.05$)*. The data matrix is* $m \times 1000$*, and its underlying rank is* d_0*. It is corrupted by additive Gaussian noise and* p_0 *sparse noise (random values).*

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
stablePCP	8.32	9.76	17.24	10.60	13.26	31.06	13.64	22.85	84.08	20.29	61.08	133.21
stablePCP(capped	7.52	8.52	14.17	6.65	7.53	10.49	6.10	7.03	9.98	5.81	6.74	10.01
ALS$_{\text{unreg}}$($2p_0$)	7.78	11.24	34.52	6.66	7.92	18.50	6.20	7.30	11.65	5.95	7.02	10.29
ALS$_{\text{unreg}}$(p_0)	7.35	9.88	23.36	5.87	6.20	9.13	5.41	5.70	6.42	5.17	5.44	6.08

where $\tilde{\mathbf{X}}$ is the low rank estimation for X. We report the mean of this error (for the 100 different times the data was randomly generated) and for fixed values of m, d_0, and p_0 in Tables 3 and 4 when $\sigma = 0$ and $\sigma = 0.05$, respectively. Note that in these 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 $p = 2p_0$.

We observe that PCP performs the best for clean data, but it could not handle Gaussian noise well. 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 that 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). We also observe that the regularization for ALS improves the results, and experiments show this improvement is due to the terms $\lambda_1\|\mathbf{B}\|_F^2 + \lambda_2\|\mathbf{C}\|_F^2$.

Table 5

The mean running times (in seconds) of the simulations with Gaussian noise ($\sigma = 0.05$). The data matrix is $m \times 1000$, and its underlying rank is d_0. It is corrupted by additive Gaussian noise and p_0 sparse noise (random values).

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
stablePCP	1.32	1.36	1.50	4.05	4.42	5.24	26.04	30.20	91.69	172.03	539.39	775.11
stablePCP(capped	0.71	0.74	0.83	2.10	2.24	2.68	11.18	11.93	14.74	61.04	64.75	81.20

Table 5 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.

Appendix B. Effect of the parameters of ALS. We experimentally demonstrate the dependence of the ALS algorithm on the modeling parameters p and d and the regularization parameters λ_1, λ_2, and λ_3. Figures 6 and 7 plot the relative fitting error versus p for different experimental settings with fixed $d = d_0 + 3$ ($\lambda_1 = \lambda_2 = \lambda_3 = 10^{-10}$ here and below, unless specified). Each subfigure has a different value of m, d_0, and p_0, where $\sigma = 0$ in Figure 6 and $\sigma = 0.05$ in Figure 7. Figures 8 and 9 plot the relative fitting error versus d for ALS($2p_0$), ALS(p_0), PCP(capped), and LMaFit(oracle) with the same choice of subfigure parameters as in Figures 6 and 7. 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 ALS algorithm is less robust to the underestimation of these two parameters; however, it can still perform well under very mild underestimation.

In order to test the effect of the regularization parameters in ALS, we set $\lambda_1 = \lambda_2 = \lambda_3 = 10^k$, where $k = -4, \ldots, -16$. We found that ALS is stable w.r.t. both accuracy and speed when λ_1, λ_2, and λ_3 are sufficiently small. The accuracy and speed for one scenario are displayed in Figures 10 and 11, respectively.

Appendix C. 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 an increasing sequence of dimensions $d_1 < \cdots < d_t$ and denote the corresponding low rank estimators of $\mathbf{X}$ by $\tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_t$. We then estimate the ranks of these estimators and denote them by $\tilde{d}_1, \ldots, \tilde{d}_t$, respectively. This estimated 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.

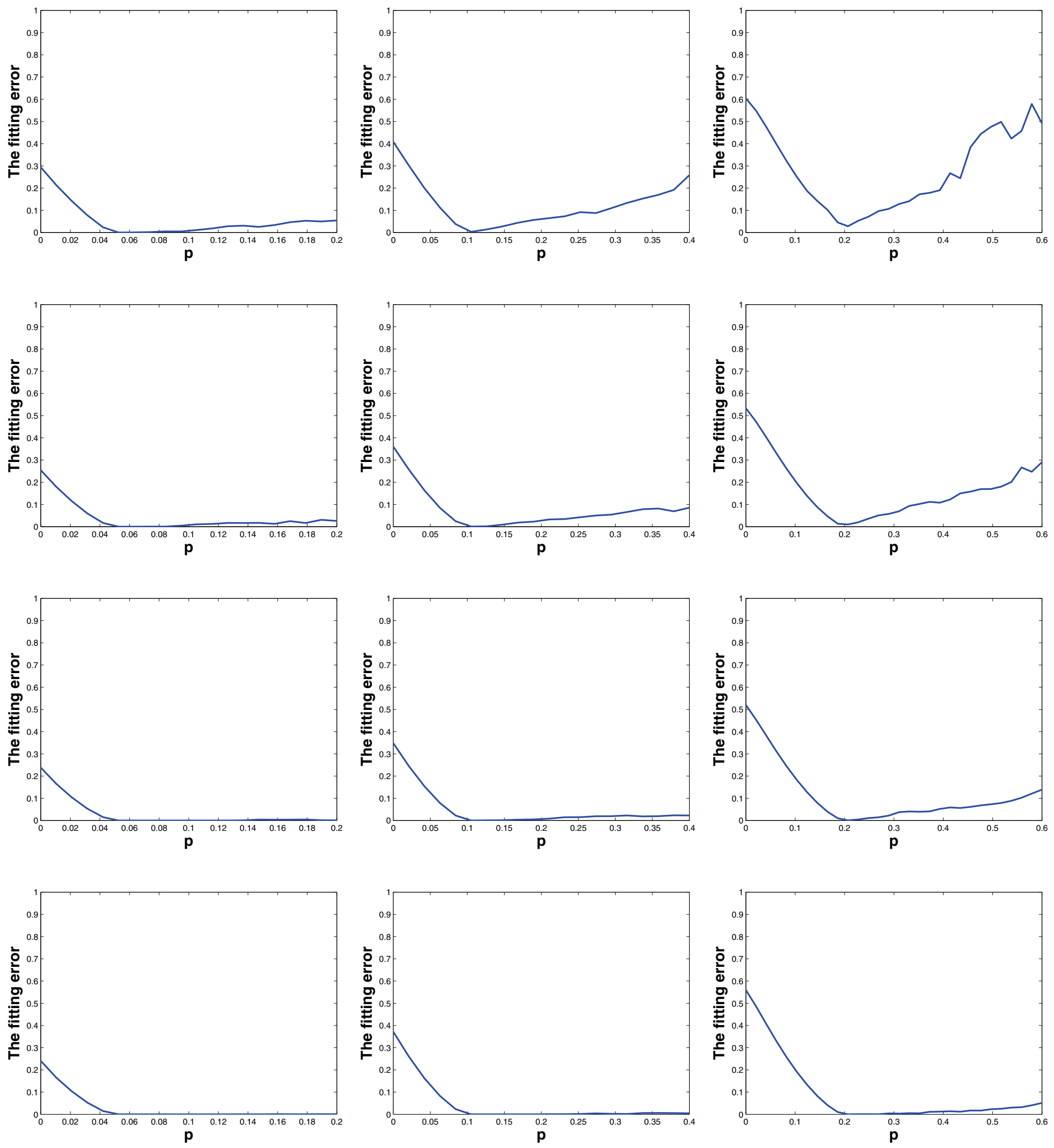


Figure 6. *The performance of the ALS algorithm for different values of the parameter* p *without Gaussian noise. In each figure, the mean relative fitting errors are plotted versus* p *for different settings, where the data matrix is* $m \times 1000$*, its underlying rank is* $m/20$*, and the data matrix is corrupted by* p_0 *sparse (or impulsive) noise. From top to bottom,* $m = 100$, 200, 400, *and* 800, *respectively. From left to right,* $p_0 = 0.05$, 0.1, *and* 0.2, *respectively. The ALS algorithm seems to be robust to overestimation of* p*, especially when the ambient dimension* (m) *is large.*

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 maximal value of the SODs of the singular values is thus expected to occur at location $r+1$. If $\sigma_1, \ldots, \sigma_m$ are the singular values

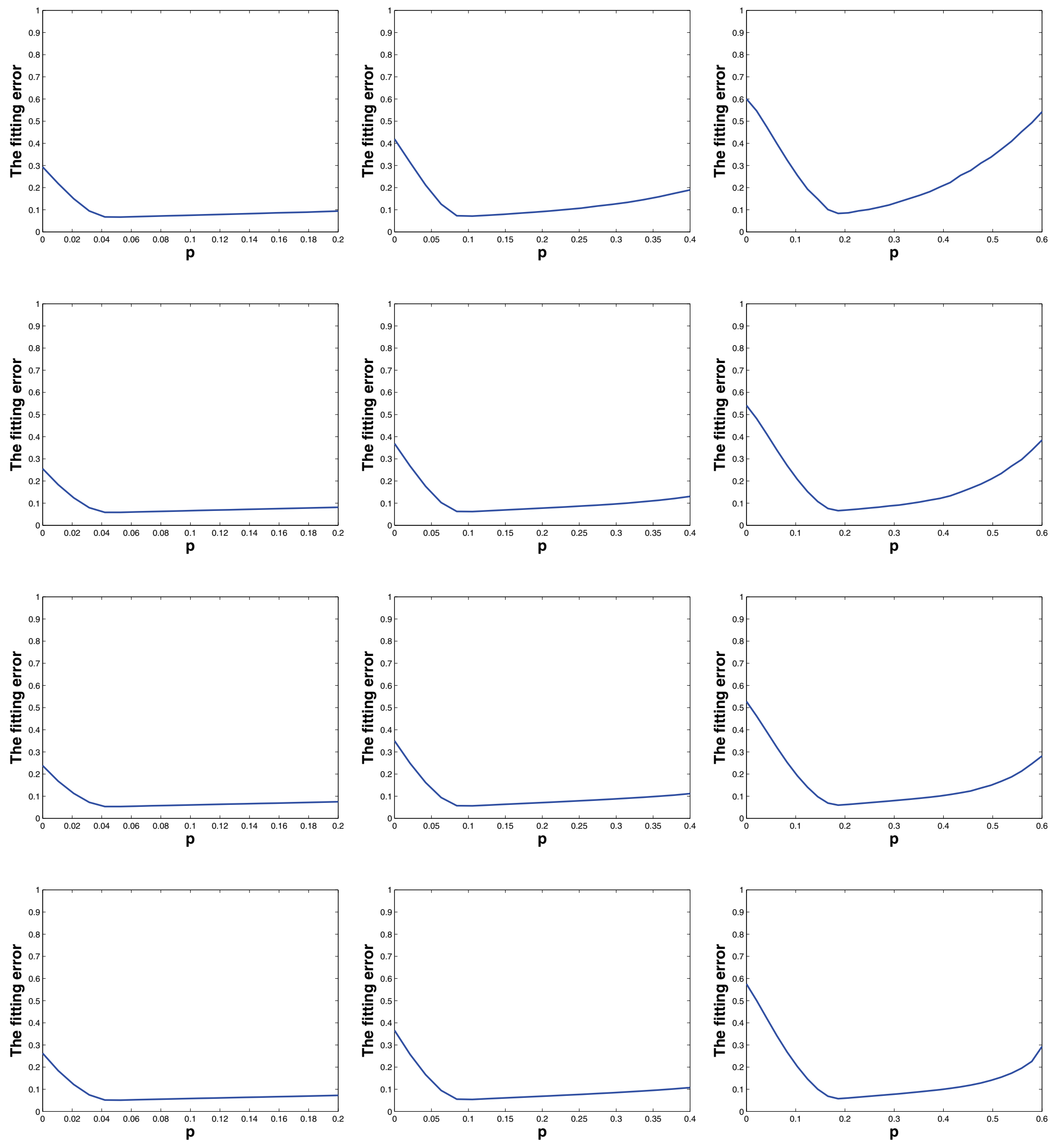


Figure 7. *The performance of the ALS algorithm for different values of the parameter* p *with Gaussian noise* ($\sigma = 0.05$)*. In each figure, the mean relative fitting errors are plotted versus* p *for different settings, where the data matrix is* $m \times 1000$*, its underlying rank is* $m/20$*, and the data matrix is corrupted by* p_0 *sparse (or impulsive) noise and additive Gaussian noise. From top to bottom,* $m = 100$, 200, 400*, and* 800*, respectively. From left to right,* $p_0 = 0.05$, 0.1*, and* 0.2*, respectively. The ALS algorithm seems to be robust to overestimation of* p*, especially when the ambient dimension* (m) *is large.*

of an $m \times n$ estimator (assume $m < n$ without loss of generality), then the SODs obtain the form

$$S(i) = \sigma_{i-1} + \sigma_{i+1} - 2\sigma_i, \quad i = 2, \dots, m-1, \tag{C.1}$$

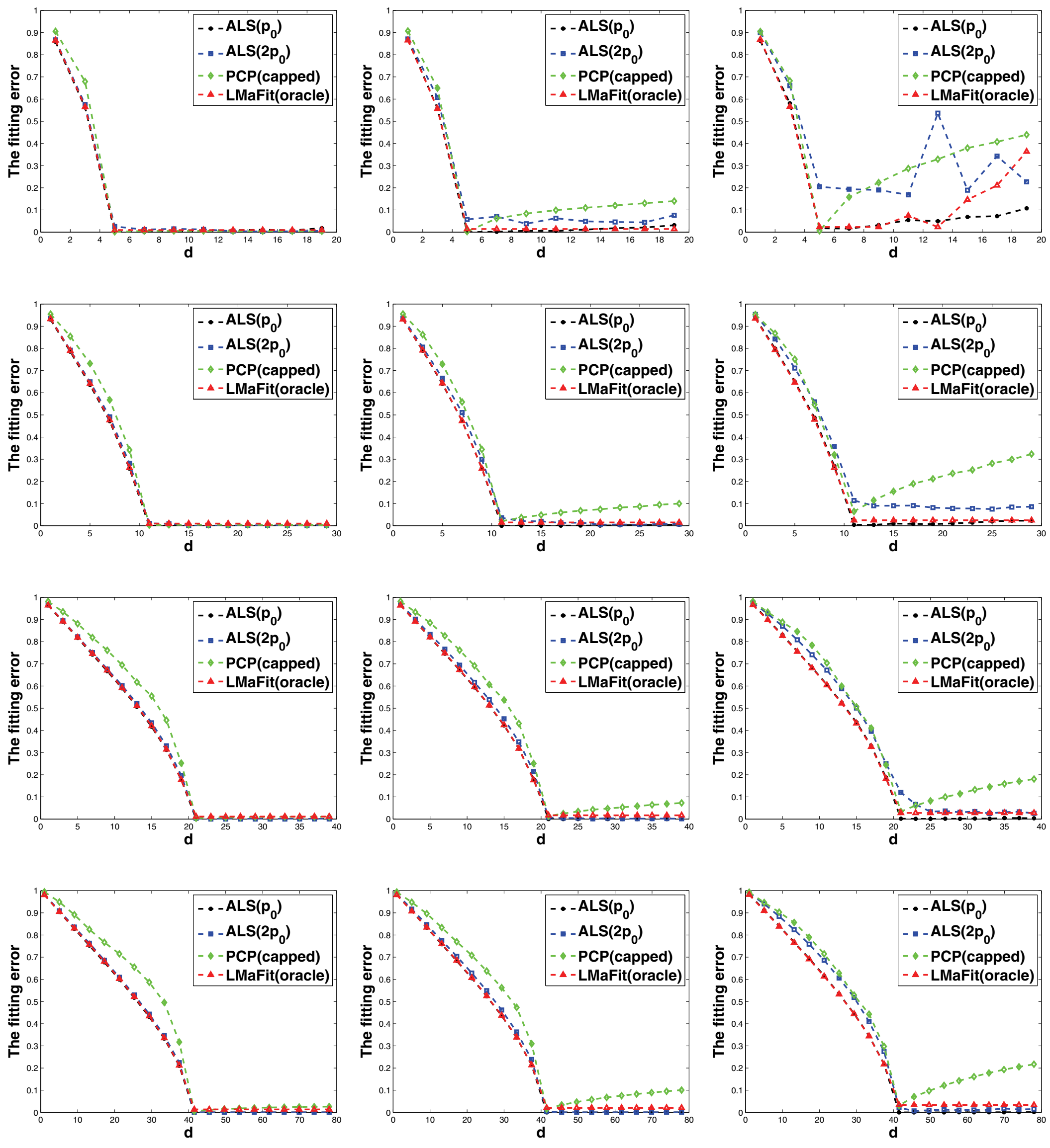


Figure 8. *The effect of the parameter d on the performance of the algorithms ALS($2p_0$), ALS(p_0), PCP(capped), and LMaFit(oracle) without Gaussian noise. In each figure, the relative fitting errors are plotted versus d for different settings, where the data matrix is $m \times 1000$, its underlying rank is $d_0 = m/20$, and the data matrix is corrupted by p_0 sparse (or impulsive) noise. From top to bottom, $m = 100, 200, 400,$ and 800, respectively. From left to right, $p_0 = 0.05, 0.1,$ and 0.2, respectively. The ALS algorithm is robust to overestimation of d, especially when the ambient dimension (m) is large.*

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

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

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

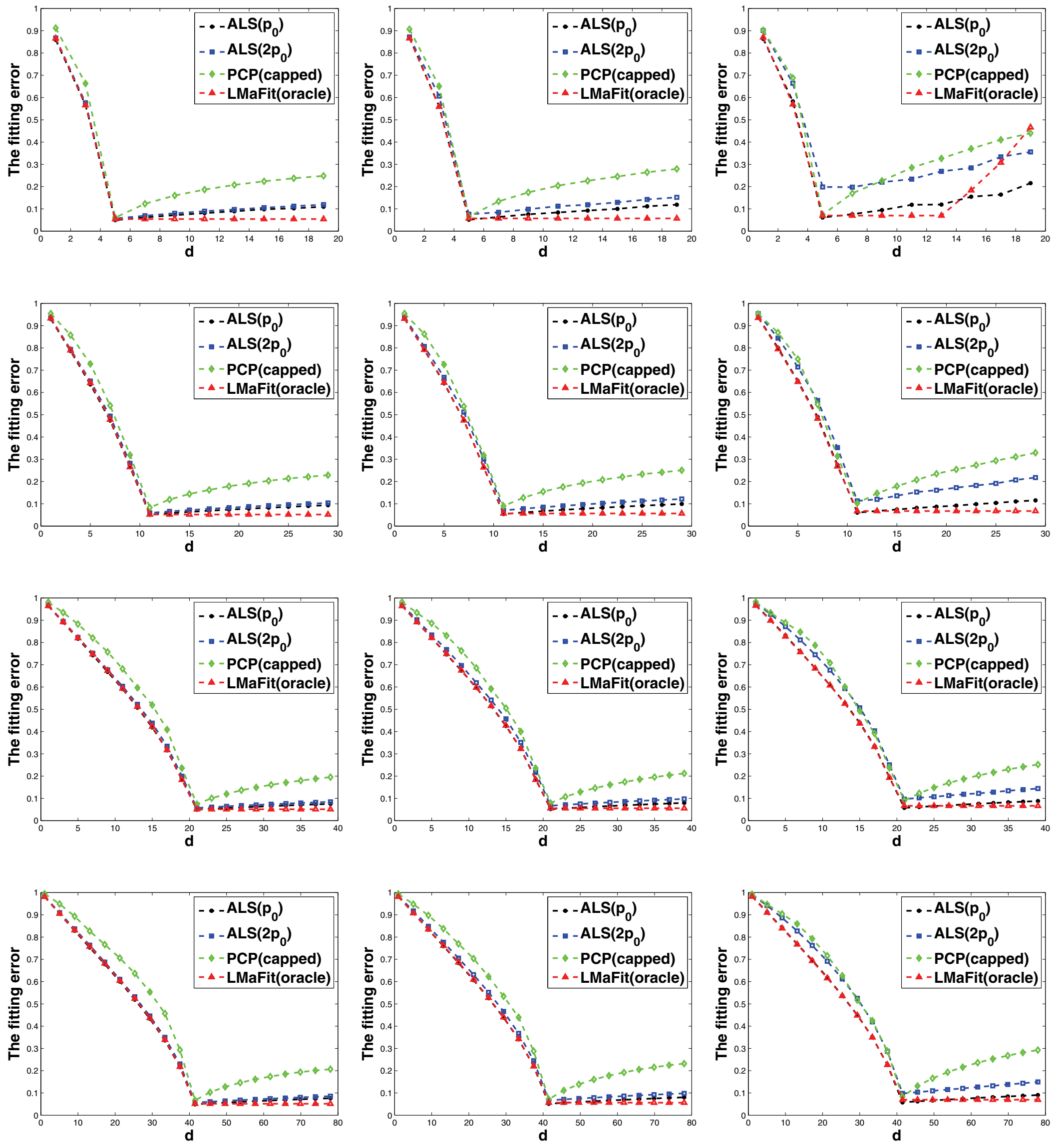


Figure 9. *The effect of the parameter d on the performance of the algorithms ALS(*$2p_0$*), ALS(*p_0*), PCP(capped), and LMaFit(oracle) with Gaussian noise (*$\sigma = 0.05$*). In each figure, the relative fitting errors are plotted versus d for different settings, where the data matrix is* $m \times 1000$*, its underlying rank is* $d_0 = m/20$*, and the data matrix is corrupted by* p_0 *sparse (or impulsive) noise and additive Gaussian noise. From top to bottom,* $m = 100$*, 200, 400, and 800, respectively. From left to right,* $p_0 = 0.05$*, 0.1, and 0.2, respectively. The ALS algorithm is robust to overestimation of d, especially when the ambient dimension (m) is large.*

most problems where the rank is relatively low, this is not the case.

We estimated the rank for the synthetic setting of Appendix A. We chose an arithmetic sequence $\{d_i\}_{i=1}^{20}$ such that $d_1 = 1$ and $d_{20} = 0.2 \cdot m$. We compared the rank estimated by ALS with that by PCP and PCP(capped), where the cap for PCP(capped) was $0.2 \cdot m$. We considered both output values of d_0 and $d_0 + 1$ as successes (after all, ALS is not sensitive

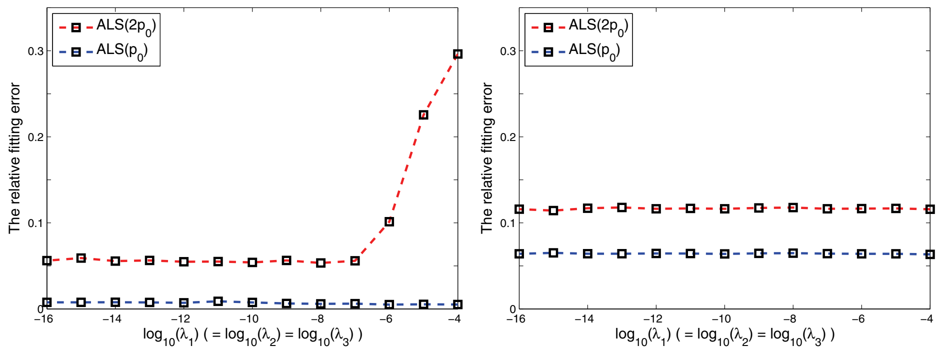


Figure 10. *The relative fitting error for the ALS algorithm with different values of the regularization parameters λ_1, λ_2, and λ_3. The left figure is without Gaussian noise, and the right one is with Gaussian noise ($\sigma = 0.05$). The ALS algorithm seems to be robust to the choice of the regularization parameters if they are sufficiently small.*

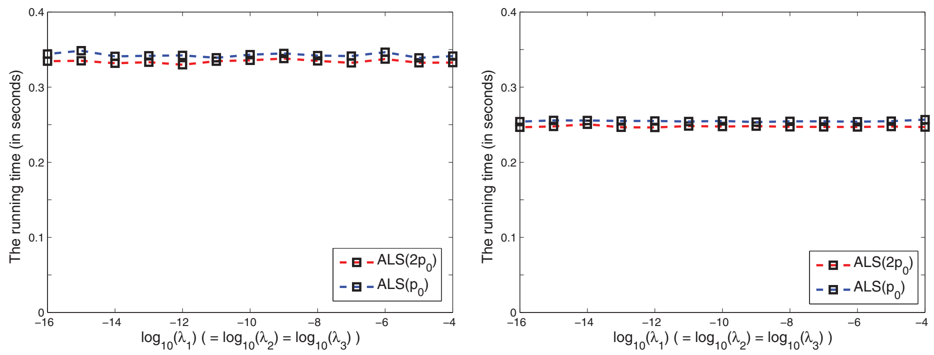


Figure 11. *The running time for the ALS algorithm with different values of the regularization parameters λ_1, λ_2, and λ_3. The left figure is without Gaussian noise, and the right one is with Gaussian noise ($\sigma = 0.05$). The choice of regularization parameters does not affect the speed of the algorithm.*

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. We found out that these three methods worked well: PCP worked perfectly; ALS made only one mistake (out of 100) when $m = 100$, $p = 20\%$, and Gaussian noise was added; PCP(capped) was correct for about 75% of the random samples, when $m = 100$, $p = 20\%$, and no Gaussian noise was added and when $m = 100, 200$ and $p = 20\%$ with Gaussian noise and was 100% correct otherwise.

Appendix D. Restoration of corrupted images in SSIM. We report the results of the experiments in section 5.3 using SSIM [46] instead of PSNR. For the five common images, SSIM for impulsive denoising is shown in Table 6 and for blind inpainting in Table 7. For the 100 images from the BSD database, SSIM is demonstrated in Figure 12 for denoising and in Figure 13 for blind inpainting.

Appendix E. Testing detection of corrupted locations. We test the detection of corrupted locations by K-ALS on the 5 popular images (used earlier) in comparison with other

Table 6

Performance (in SSIM) of denoising for 5 popular images. Each line lists results of different approaches on a specific image corrupted by σ Gaussian noise and p_0 impulsive noise. The best results are in bold. K-ALS outperforms other methods for most images except for Peppers.

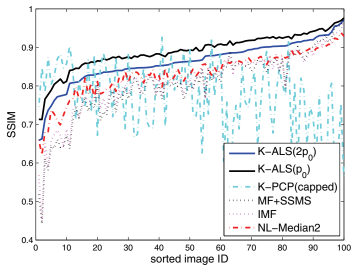
	p_0	σ	K-ALS($2p_0$)	K-ALS(p_0)	PCP(capped)+SSMS	K-PCP(capped)	KSVD	SSMS	MF+KSVD	MF+SSMS	IMF	NL-Median2	PCP+SSMS	K-PCP	LRR
Barbara	5%	10	0.897	**0.912**	0.654	0.656	0.582	0.616	0.726	0.754	0.727	0.846	0.641	0.874	0.895
		20	0.836	**0.849**	0.538	0.557	0.578	0.672	0.654	0.692	0.642	0.679	0.681	0.812	0.827
		30	0.777	**0.791**	0.509	0.554	0.625	0.710	0.604	0.653	0.595	0.535	0.708	0.750	0.768
	10%	10	0.867	**0.896**	0.566	0.607	0.425	0.442	0.723	0.749	0.717	0.832	0.444	0.859	0.875
		20	0.799	**0.826**	0.485	0.521	0.421	0.516	0.651	0.686	0.634	0.655	0.517	0.797	0.806
		30	0.733	**0.763**	0.473	0.529	0.495	0.600	0.601	0.648	0.586	0.509	0.603	0.739	0.748
Boat	5%	10	0.839	**0.854**	0.610	0.618	0.525	0.571	0.750	0.792	0.787	0.794	0.601	0.807	0.844
		20	0.770	**0.783**	0.496	0.500	0.523	0.633	0.673	0.727	0.700	0.619	0.640	0.742	0.772
		30	0.715	**0.727**	0.489	0.496	0.581	0.663	0.622	0.682	0.641	0.473	0.662	0.688	0.714
	10%	10	0.812	**0.841**	0.529	0.565	0.383	0.404	0.746	0.787	0.777	0.777	0.404	0.795	0.833
		20	0.740	**0.765**	0.465	0.464	0.368	0.485	0.668	0.723	0.690	0.596	0.491	0.728	0.760
		30	0.679	**0.704**	0.472	0.471	0.461	0.567	0.618	0.678	0.630	0.448	0.572	0.679	**0.704**
House	5%	10	0.878	**0.888**	0.508	0.600	0.482	0.537	0.848	0.856	0.834	0.804	0.874	0.821	0.815
		20	0.836	**0.844**	0.410	0.497	0.519	0.604	0.812	0.824	0.776	0.598	0.816	0.794	0.783
		30	0.797	**0.805**	0.435	0.527	0.612	0.684	0.759	0.800	0.712	0.442	0.790	0.767	0.752
	10%	10	0.861	**0.879**	0.452	0.547	0.323	0.350	0.845	0.853	0.792	0.790	0.681	0.816	0.811
		20	0.810	**0.828**	0.391	0.469	0.338	0.447	0.811	0.821	0.767	0.578	0.715	0.794	0.775
		30	0.765	0.784	0.416	0.498	0.449	0.559	0.761	**0.794**	0.702	0.422	0.698	0.777	0.747
Lena	5%	10	0.884	**0.894**	0.498	0.530	0.477	0.531	0.849	0.871	0.846	0.811	0.574	0.868	0.890
		20	0.837	**0.847**	0.402	0.429	0.516	0.616	0.799	0.831	0.781	0.603	0.632	0.826	0.846
		30	0.794	0.805	0.427	0.451	0.605	0.700	0.759	0.800	0.719	0.442	0.703	0.793	**0.810**
	10%	10	0.864	**0.884**	0.433	0.485	0.318	0.342	0.848	0.868	0.839	0.796	0.351	0.863	0.881
		20	0.808	0.828	0.383	0.400	0.337	0.447	0.797	0.828	0.773	0.582	0.441	0.819	**0.833**
		30	0.758	0.781	0.414	0.428	0.441	0.576	0.755	0.795	0.709	0.422	0.578	0.784	**0.796**
Peppers	5%	10	0.850	**0.860**	0.479	0.523	0.453	0.503	0.834	0.847	0.830	0.794	0.549	0.838	0.856
		20	0.812	0.822	0.401	0.425	0.488	0.578	0.802	0.818	0.776	0.596	0.600	0.810	**0.823**
		30	0.776	0.786	0.437	0.444	0.575	0.673	0.771	0.792	0.715	0.439	0.676	0.783	**0.795**
	10%	10	0.833	**0.851**	0.425	0.478	0.302	0.324	0.831	0.844	0.825	0.780	0.332	0.834	0.848
		20	0.780	0.802	0.380	0.391	0.313	0.416	0.798	**0.813**	0.768	0.575	0.414	0.803	0.810
		30	0.740	0.763	0.417	0.422	0.407	0.538	0.766	**0.787**	0.706	0.419	0.546	0.777	0.781

appropriate methods (K-PCP(capped), MF+SSMS, IMF, NL-Median2, K-LMAFit, LRR). For each method, we compute $r = \text{abs}(\mathbf{Y} - \tilde{\mathbf{Y}})$, where $\mathbf{Y}$ is the input image and $\tilde{\mathbf{Y}}$ is the output of that method, and identify the corrupted pixels of the true percentage (p_0) as the ones with the largest entries in r. We also test K-ALS($2p_0$), where the true percentage of corruption is overestimated by a factor of two (unlike all other methods). The detection errors are displayed in Figures 14 and 15. The K-ALS algorithm (with p_0 percentage) outperforms other methods (with the same percentage p_0) in terms of detecting the corrupted locations in most cases, especially in the presence of impulsive noise. We believe that the good performance of K-ALS in terms of corruption detection is what helps it in obtaining good denoising and inpainting results.

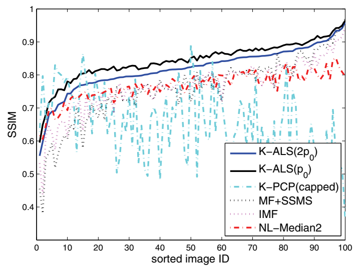
Table 7

Performance (in SSIM) of blind inpainting for 5 popular images. Each line lists results of different approaches on a specific image corrupted by σ Gaussian noise, p_0 impulsive noise, and some scratches. The best results are in bold. We note that for most cases K-ALS outperforms other methods.

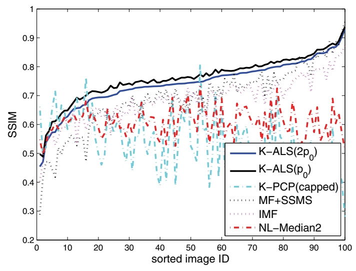
	p_0	σ	K-ALS($2p_0$)	K-ALS(p_0)	PCP(capped)+SSMS	K-PCP(capped)	KSVD	SSMS	MF+KSVD	MF+SSMS	IMF	NL-Median1	PCP+SSMS	K-PCP	LRR
		0	0.956	0.959	0.938	0.944	0.945	0.945	0.809	0.809	0.809	0.860	0.949	0.937	**0.960**
	0%	5	0.932	**0.936**	0.873	0.891	0.912	0.911	0.780	0.789	0.784	0.839	0.915	0.898	0.921
		10	0.904	**0.908**	0.730	0.717	0.886	0.887	0.725	0.755	0.732	0.788	0.888	0.866	0.890
		0	0.936	**0.950**	0.851	0.808	0.564	0.5664	0.798	0.798	0.798	0.846	0.587	0.920	0.935
Barbara	5%	5	0.913	**0.927**	0.791	0.765	0.562	0.567	0.773	0.782	0.772	0.824	0.583	0.883	0.907
		10	0.882	**0.895**	0.657	0.644	0.557	0.589	0.721	0.749	0.722	0.770	0.610	0.851	0.877
		0	0.863	**0.938**	0.710	0.690	0.408	0.407	0.784	0.784	0.784	0.831	0.406	0.903	0.893
	10%	5	0.848	**0.914**	0.670	0.667	0.406	0.407	0.764	0.772	0.756	0.807	0.407	0.868	0.879
		10	0.830	**0.877**	0.574	0.594	0.406	0.423	0.715	0.741	0.711	0.751	0.431	0.837	0.852
		0	0.920	0.930	0.930	0.921	0.932	0.932	0.847	0.847	0.847	0.851	0.941	0.929	**0.959**
	0%	5	0.884	**0.894**	0.853	0.853	0.878	0.871	0.812	0.830	0.825	0.832	0.877	0.852	0.883
		10	0.843	**0.851**	0.664	0.673	0.826	0.826	0.748	0.791	0.789	0.787	0.828	0.809	0.837
		0	0.900	0.918	0.793	0.784	0.538	0.538	0.838	0.838	0.838	0.845	0.566	0.902	**0.937**
Boat	5%	5	0.864	**0.882**	0.730	0.737	0.526	0.528	0.808	0.824	0.818	0.825	0.553	0.831	0.876
		10	0.821	**0.837**	0.598	0.602	0.508	0.548	0.744	0.786	0.781	0.776	0.572	0.792	0.829
		0	0.862	**0.907**	0.656	0.665	0.369	0.369	0.824	0.824	0.827	0.837	0.372	0.886	0.902
	10%	5	0.829	**0.868**	0.613	0.635	0.369	0.369	0.800	0.815	0.810	0.814	0.371	0.821	0.860
		10	0.790	**0.824**	0.520	0.548	0.367	0.386	0.737	0.779	0.770	0.764	0.390	0.777	0.813
		0	**0.955**	0.954	0.866	0.855	0.849	0.849	0.868	0.868	0.883	0.881	0.929	0.899	0.894
	0%	5	0.914	**0.915**	0.775	0.810	0.809	0.805	0852	0.860	0.865	0.846	0.867	0.856	0.839
		10	**0.870**	0.866	0.567	0.629	0.767	0.770	0.831	0.841	0.831	0.788	0.817	0.815	0.815
		0	0.953	**0.961**	0.703	0.692	0.457	0.457	0.854	0.855	0.877	0.872	0.755	0.871	0.883
House	5%	5	0.907	**0.915**	0.614	0.677	0.438	0.443	0.845	0.848	0.858	0.841	0.744	0.837	0.834
		10	0.863	**0.867**	0.480	0.563	0.435	0.474	0.820	0.831	0.826	0.776	0.740	0.809	0.811
		0	0.925	**0.956**	0.540	0.566	0.300	0.300	0.829	0.833	0.867	0.867	0.454	0.861	0.874
	10%	5	0.881	**0.910**	0.498	0.564	0.295	0.297	0.829	0.831	0.848	0.832	0.435	0.839	0.827
		10	0.844	**0.862**	0.433	0.510	0.301	0.323	0.812	0.818	0.819	0.766	0.504	0.809	0.806
		0	0.950	0.955	0.932	0.924	0.933	0.933	0.918	0.918	0.918	0.909	0.943	0.940	**0.970**
	0%	5	0.916	**0.921**	0.809	0.837	0.883	0.880	0.883	0.894	0.889	0.885	0.888	0.887	0.911
		10	0.886	**0.888**	0.549	0.590	0.851	0.852	0.846	0.869	0.847	0.829	0.856	0.861	0.881
		0	0.936	0.949	0.722	0.710	0.480	0.480	0.909	0.910	0.910	0.905	0.513	0.916	**0.954**
Lena	5%	5	0.903	**0.914**	0.629	0.672	0.469	0.476	0.879	0.889	0.884	0.880	0.512	0.874	0.906
		10	0.872	**0.881**	0.474	0.518	0.458	0.501	0.843	0.865	0.841	0.818	0.540	0.850	0.875
		0	0.907	**0.942**	0.553	0.582	0.305	0.305	0.897	0.897	0.900	0.900	0.306	0.908	0.930
	10%	5	0.883	**0.906**	0.510	0.563	0.302	0.304	0.872	0.882	0.878	0.872	0.308	0.867	0.894
		10	0.851	**0.871**	0.428	0.471	0.308	0.329	0.837	0.859	0.833	0.806	0.335	0.842	0.861
		0	0.928	0.924	0.912	0.920	0.928	0.928	0.886	0.886	0.886	0.875	0.938	0.916	**0.959**
	0%	5	**0.878**	0.875	0.753	0.810	0.860	0.851	0.856	0.864	0.866	0.855	0.860	0.849	0.875
		10	**0.847**	0.846	0.512	0.571	0.818	0.818	0.832	0.846	0.831	0.803	0.822	0.827	**0.847**
		0	0.916	0.925	0.689	0.688	0.466	0.466	0.878	0.878	0.879	0.872	0.497	0.911	**0.941**
Peppers	5%	5	0.869	**0.877**	0.603	0.657	0.449	0.452	0.852	0.860	0.861	0.849	0.491	0.844	0.872
		10	0.840	**0.844**	0.465	0.507	0.437	0.477	0.828	0.841	0.826	0.793	0.514	0.823	0.842
		0	0.889	**0.919**	0.533	0.561	0.291	0.291	0.865	0.866	0.874	0.868	0.296	0.903	0.911
	10%	5	0.852	**0.872**	0.493	0.547	0.290	0.290	0.844	0.852	0.855	0.843	0.288	0.837	0.862
		10	0.821	**0.836**	0.419	0.462	0.288	0.307	0.821	0.834	0.820	0.782	0.313	0.816	0.831



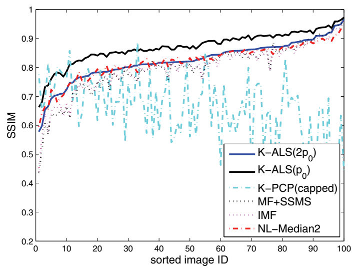
(a) $\sigma = 10$, $p_0 = 5\%$



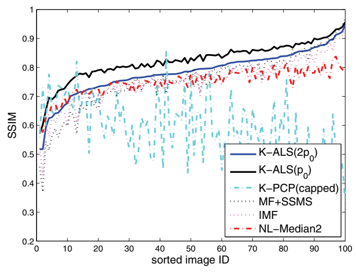
(b) $\sigma = 20$, $p_0 = 5\%$



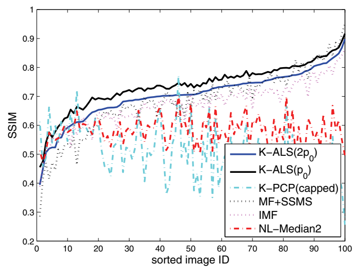
(c) $\sigma = 30$, $p_0 = 5\%$



(d) $\sigma = 10$, $p_0 = 10\%$



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



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

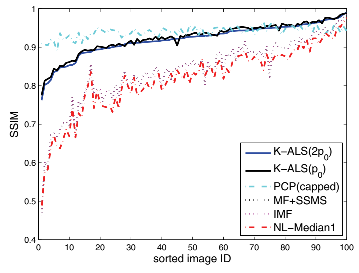
Figure 12. *Performance (in SSIM) of denoising for the BSD database* [27]. *In each subfigure, SSIM indices of several approaches (see the legends) are plotted versus image IDs for the case when the input image is corrupted by σ Gaussian noise and p_0 impulsive noise. K-ALS has the best performance for most of the cases, especially when the Gaussian noise is high.*

Appendix F. Sensitivity of the K-ALS algorithm w.r.t. the dimension. In this section we study the sensitivity of the K-ALS algorithm w.r.t. the choice of the dimension d' in the two imaging problems. Intuitively, we observe that the SVD of the clusters of natural image patches can be well approximated with about rank 15, and slight overestimation cannot affect the results. Thus we tested K-ALS($2p_0$) on 2 of the popular images (*House* and *Lena*) with different values of d' (from 10 to 40) for both denoising and blind inpainting. The results are shown in Figure 16, where in each subfigure performances (in PSNR) of different scenarios (with different p_0 and σ) are displayed. By these experiments, we conclude that K-ALS is robust to the choice of the dimension. It tends to improve with larger d' when the image is mildly corrupted; however, increasing d' could be worse if the image were largely degraded.

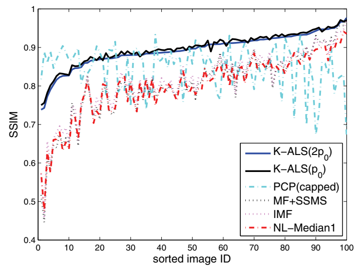
Appendix G. Mathematical analysis of the ALS algorithm. We analyze the performance of the ALS algorithm following the strategies of [38, 3].

G.1. Preliminaries.

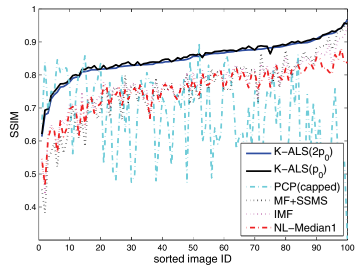
G.1.1. Notation. Let $\mathbf{1}_{m\times n}$ denote the $m \times n$ matrix whose elements are all equal to 1, let $\mathbf{I}$ denote the identity matrix (whose dimension will be clear from the context), and let $|\mathbf{A}|_0$ denote the number of nonzero elements of the matrix $\mathbf{A}$. We let $\mathbf{A}_{\cdot j}$ and $\mathbf{A}_{i\cdot}$ denote the jth column and ith row, respectively, of the matrix $\mathbf{A}$.



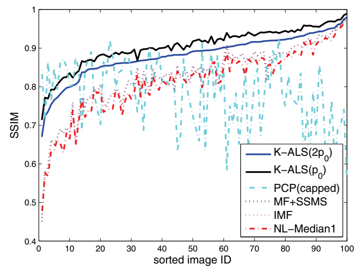
(a) $\sigma = 0$, $p_0 = 0\%$



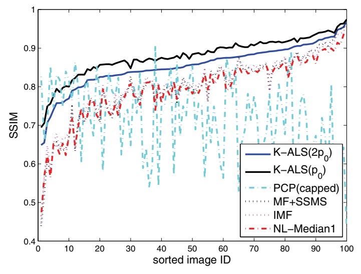
(b) $\sigma = 5$, $p_0 = 0\%$



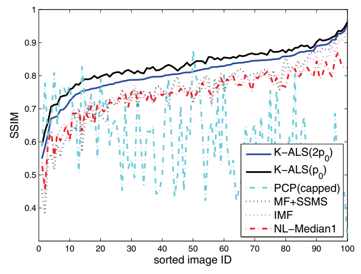
(c) $\sigma = 10$, $p_0 = 0\%$



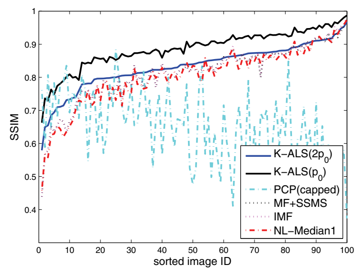
(d) $\sigma = 0$, $p_0 = 5\%$



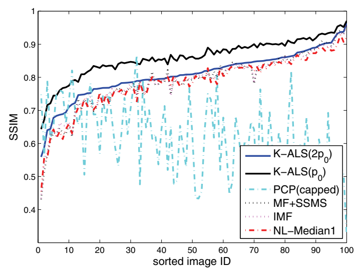
(e) $\sigma = 5$, $p_0 = 5\%$



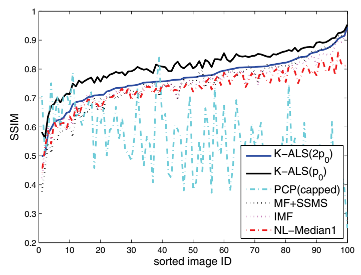
(f) $\sigma = 10$, $p_0 = 5\%$



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



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



(i) $\sigma = 10$, $p_0 = 10\%$

Figure 13. *Performance (in SSIM) of blind inpainting for the BSD database* [27]*. In each subfigure, SSIM indices of several approaches (see the legends) are plotted versus image IDs for the case when the input image is corrupted by σ Gaussian noise, p_0 impulsive noise, and some scratches. K-ALS has the best performance for most of the cases, especially when the Gaussian noise is high. For a few images the SSIM values of PCP(capped) are high, but their perceptual quality is not as satisfying.*

G.1.2. Point-to-set maps.

Definition G.1 (point-to-set map). *Given two sets $\mathcal{X}$,$\mathcal{Y}$, a point-to-set map Ω is a function Ω: $\mathcal{X} \to \mathcal{P}(\mathcal{Y})$. The composition of two point-to-set maps Ω_1: $\mathcal{X} \to \mathcal{P}(\mathcal{Y})$ and Ω_2: $\mathcal{Y} \to \mathcal{P}(\mathcal{Z})$ is defined by $(\Omega_2 \circ \Omega_1)(\mathbf{x}) = \bigcup_{\mathbf{y} \in \Omega_1(\mathbf{x})} \Omega_2(\mathbf{y})$.*

Definition G.2 (closed map). *A point-to-set map Ω is closed at $\hat{\mathbf{x}} \in \mathcal{X}$ if, for any $\hat{\mathbf{y}} \in \mathcal{P}(\mathcal{Y})$ created by a sequence $\{\mathbf{x}_k\} \subset \mathcal{X}$ such that $\mathbf{x}_k \to \hat{\mathbf{x}}$ and by a sequence $\mathbf{y}_k \in \Omega(\mathbf{x}_k)$ such that $\mathbf{y}_k \to \hat{\mathbf{y}}$, $\hat{\mathbf{y}} \in \Omega(\hat{\mathbf{x}})$.*

Definition G.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 fixed point of Ω is a point $\mathbf{x}$ for which $\mathbf{x} \in \Omega(\mathbf{x})$.*

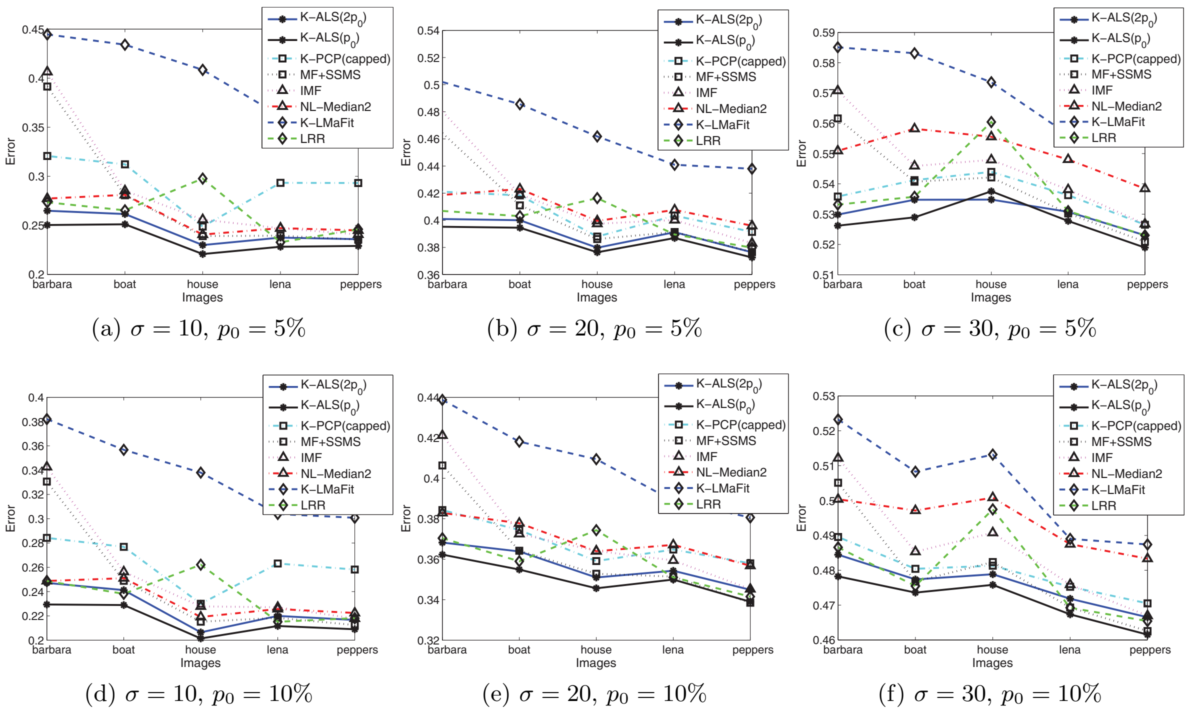


Figure 14. *Errors of detecting the corrupted locations for the 5 popular images. In each subfigure, errors of a specific corrupted scenario (see the subcaptions) are plotted versus the images. The number in parentheses after K-ALS specifies the input estimating the portion of corruptions. Other methods use the best values from a broad range of parameters; see section 5.3. For most of the cases, K-ALS outperforms the other algorithms.*

G.1.3. Iterative algorithms. An iterative algorithm is a point-to-set map $\Omega : \mathcal{X} \to \mathcal{P}(\mathcal{X})$. It generates a sequence of points via the rule $\mathbf{x}_{k+1} \in \Omega(\mathbf{x}_k)$, where $\mathbf{x}_0$ is a given initial point. Now, suppose that $\phi : \mathcal{X} \to \mathbb{R}_+$ is a continuous, nonnegative function. An algorithm Ω is *monotonic* w.r.t. ϕ whenever $\mathbf{y} \in \Omega(\mathbf{x})$ implies that $\phi(\mathbf{y}) \le \phi(\mathbf{x})$. 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.*

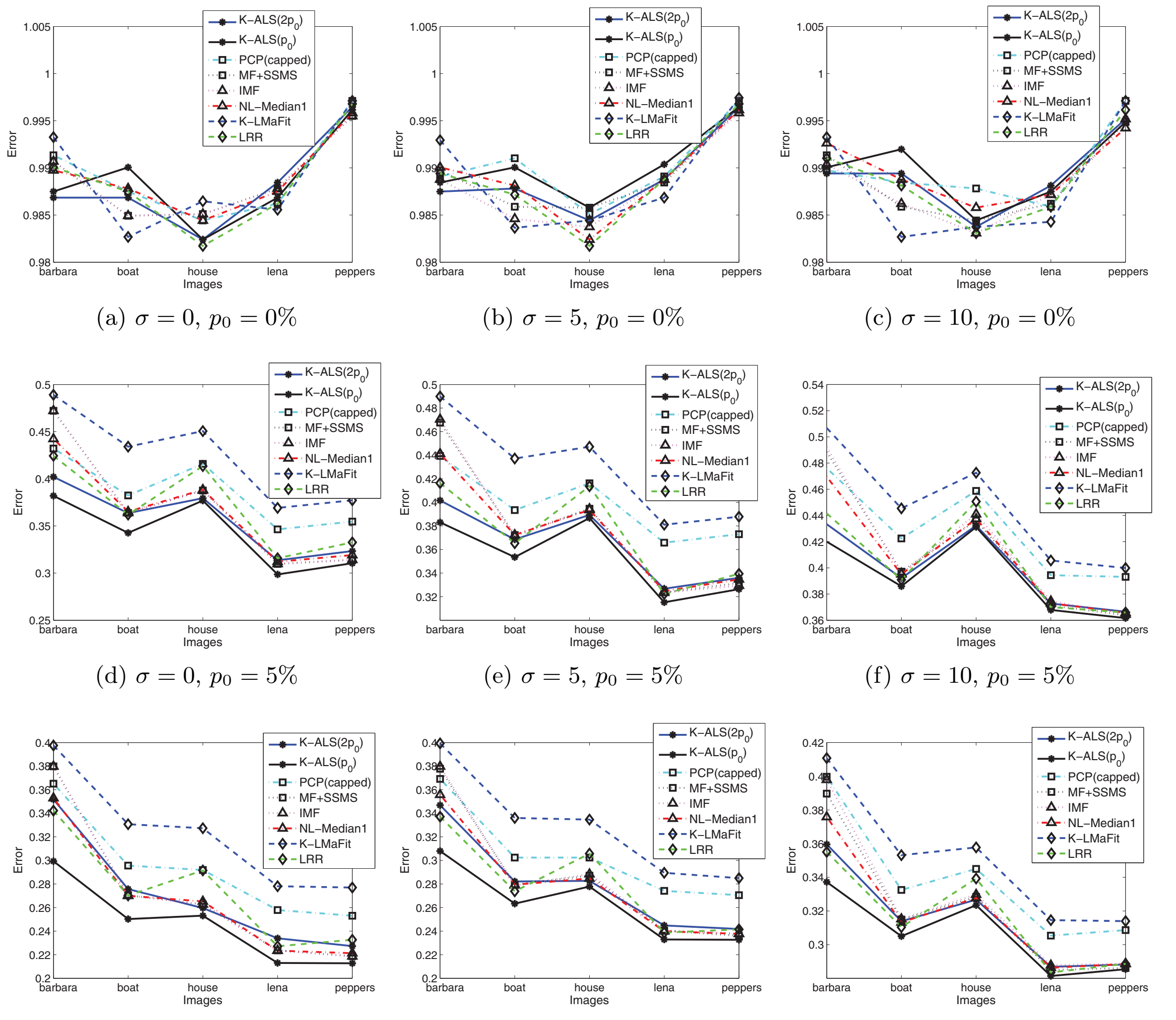
We review the following theorems on convergence of iterative algorithms.

Theorem G.4 (see Zangwill [47]). *Let Ω be an algorithm that is monotonic w.r.t. 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 G.5 (see Meyer [28]). *Assume that the algorithm Ω is strictly monotonic w.r.t. 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, then $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \to 0$. 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.*

G.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 $M_{\mathbf{x}} : \mathcal{Y} \to \mathcal{P}(\mathcal{X})$.

We will decompose our algorithms by infimal maps and use the following auxiliary lemmas in order to apply Theorems G.4 and G.5.



(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) $\sigma = 10$, $p_0 = 10\%$

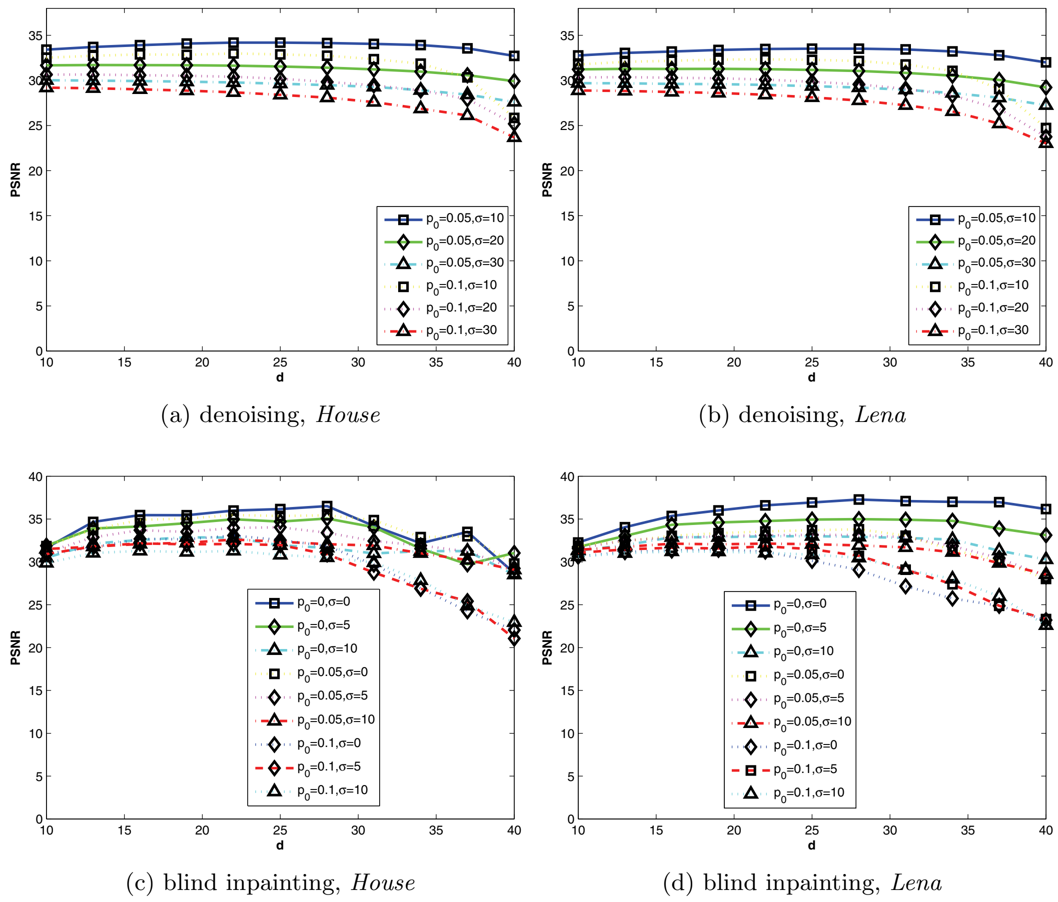
Figure 15. *Errors of detecting the corrupted locations for the 5 popular images. In each subfigure, errors of a specific corrupted scenario are plotted versus the images (the impulsive and Gaussian noise parameters are in the subcaptions, and the scratch used throughout this paper is also added). The number in parentheses after K-ALS specifies the input estimating the portion of corruptions. Other methods use the best values from a broad range of parameters; see section 5.3. When the impulsive noise is present, K-ALS(p_0) outperforms other algorithms.*

Theorem G.6 (see Dantzig, Folkman, and Shapiro [11]). *If $\phi(\hat{\mathbf{x}}, \cdot)$ is continuous on $\mathcal{Y}$, then $M_{\mathbf{y}}$ is closed at $\hat{\mathbf{x}}$.*

Theorem G.7 (see Fiorot and Huard [14]). *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 w.r.t. ϕ.*

G.2. The ALS algorithm as a point-to-set map.

In ALS the energy function J of (2.6) serves as the function ϕ of section G.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 $d < \min(m,n)$



(a) denoising, *House* (b) denoising, *Lena*

(c) blind inpainting, *House* (d) blind inpainting, *Lena*

Figure 16. *The performance (in PSNR) of K-ALS($2p_0$) versus different values of the dimension d' for House (left) and Lena (right) for denoising (top) and blind inpainting (bottom). The different curves in the figures correspond to different values of σ and p_0 as specified. The K-ALS($2p_0$) is robust to the choice of d', and it favors small values for large corruptions and large values for small corruptions.*

and that 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

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

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

G.3. Conclusion of Theorem 4.1.

At each step, the ALS algorithm is composed of $2t+1$ infimal maps as in (G.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 G.6 implies that both $M_{\mathbf{B}}$ and $M_{\mathbf{C}}$ are closed. Furthermore, $M_{\mathbf{W}}$ is closed because $\mathcal{F}$ is finite. Therefore, the ALS algorithm is closed. Lemmas G.8 and G.9, which are formulated and proved below, imply that the ALS algorithm can be decomposed by single-valued infimal maps. Therefore, in view of Theorem G.7, the ALS algorithm is strictly monotonic. Consequently, $\lambda_1\|\mathbf{B}^t\|_F^2 + \lambda_2\|\mathbf{C}^t\|_F^2 \leq J(\mathbf{B}^t, \mathbf{C}^t, \mathbf{W}^t) < J(\mathbf{B}^0, \mathbf{C}^0, \mathbf{W}^0)$. Thus the iterates of ALS are uniformly bounded. Combining these observations with Theorem G.5, we conclude Theorem 4.1.

G.3.1. Establishing single values for $M_{\mathbf{W}}$.

Lemma G.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*

$$\{|(\mathbf{B}^t\mathbf{C}^t)_{ij} - \mathbf{X}_{ij}|^2 + \lambda_3 \mathbf{U}_{ij}^2\}_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}$$

are different with probability 1.

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

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

G.3.2. Establishing single values for $M_{\mathbf{B}}$ and $M_{\mathbf{C}}$.

Lemma G.9. *The infimal maps* $M_{\mathbf{B}}$ *and* $M_{\mathbf{C}}$ *derived from* (2.6) *are single-valued.*

Proof. By the definition of infimal maps,

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

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

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

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

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

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

G.4. Conclusion of Theorem 4.2. The K-ALS algorithm can also be viewed as a point-to-set map

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

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

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

where $\mathbf{W} = \mathbf{P}[\mathbf{W}_1 \ldots \mathbf{W}_K]$ and $\mathbf{P}$ 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 that of Theorem 4.1. It suffices to show that (G.7) is single-valued. Because each element of $\mathbf{V}$ is i.i.d. sampled from a continuous distribution on [0, 1], we have

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

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

G.5. Nonlocal medians. Many of the best performing image denoising methods (in the additive Gaussian noise setting) are versions of the nonlocal means algorithm (NLM) [6]. 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 two robust versions of NLM, which we refer to as the nonlocal medians. First, we define two different similarity functions between patches $\mathbf{x}_j$ and $\mathbf{x}_i$ (of width $\sqrt{m}$) as follows: (1) first apply a median filter to the entire image, and then use the regular Euclidean distance (d_2); (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.

To find the nonlocal medians, we define a square neighborhood of length l for the ith pixel and denote the indices of the pixels in this neighborhood by S_i. We compute the distances (d_2 after median filtering or $\tilde{d}_r$) 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 $\tilde{S}_i$. We refer to these two methods as NL-Median1 and NL-Median2, respectively.

In our experiments testing these two nonlocal median methods, we let $m = 16$, $s = 12$, $l = 7$, 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., $\sigma \geq 20$ (although this improvement with known σ and p_0 was still not better than K-ALS(p_0)).

Acknowledgments. We thank the reviewers for their careful reading of the paper and their valuable comments; Fatih Porikli for the professional handling of this manuscript; Guoshen Yu for carefully explaining details of [46]; Sathiya Keerthi for referring us to the initial technical report version of [38] and providing some other references for alternating least squares; Joel Tropp for corresponding with YW regarding [38]; Rene Vidal for some feedback on GL's short presentation of this work and some useful references; and both Teng Zhang and John Wright for commenting on an earlier version of this manuscript (John also contributed several references, in particular, ones related to the stablePCP algorithm).

REFERENCES

[1] A. Agarwal, S. Negahban, and M. Wainwright, *Noisy matrix decomposition via convex relaxation: Optimal rates in high dimensions*, in Proceedings of the 28th International Conference on Machine Learning (ICML), Omnipress, Madison, WI, 2011, pp. 1129–1136.

[2] M. Aharon, M. Elad, and A. Bruckstein, *K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Trans. Signal Process., 54 (2006), pp. 4311–4322.

[3] 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, IEEE, New York, 2010.

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

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

[6] A. Buades, B. Coll, and J.-M. Morel, *A non-local algorithm for image denoising*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 60–65.

[7] A. M. Buchanan and A. W. Fitzgibbon, *Damped Newton algorithms for matrix factorization with missing data*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 316–322.

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

[9] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, *Rank-sparsity incoherence for matrix decomposition*, SIAM J. Optim., 21 (2011), pp. 572–596.

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

[11] 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.

[12] B. Dong, H. Ji, J. Li, Z. Shen, and Y. Xu, *Wavelet frame based blind image inpainting*, Appl. Comput. Harmon. Anal., 32 (2011), pp. 268–279.

[13] M. Elad and M. Aharon, *Image denoising via learned dictionaries and sparse representation*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 17–22.

[14] J. C. Fiorot and P. Huard, *Composition and union of general algorithms of optimization. Point-to-set maps and mathematical programming*, Math. Programming Studies, No. 10 (1979), pp. 69–85.

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

[16] D. Hsu, S. M. Kakade, and T. Zhang, *Robust matrix decomposition with sparse corruptions*, IEEE Trans. Inform. Theory, 57 (2011), pp. 7221–7234.

[17] P. J. Huber and E. M. Ronchetti, *Robust Statistics*, Wiley Ser. Probab. Stat., John Wiley & Sons, Hoboken, NJ, 2009.

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

[19] A. T. Kyrillidis and V. Cevher, *Matrix ALPS: Accelerated Low Rank and Sparse Matrix Reconstruction*, CoRR, abs/1203.3864, 2012.

[20] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*, Inf. Sci. Stat., Springer, New York, 2007.

[21] G. Lerman and T. Zhang, *Robust recovery of multiple subspaces by geometric l_p minimization*, Ann. Statist., 39 (2011), pp. 2686–2715.

[22] X. Liang, X. Ren, Z. Zhang, and Y. Ma, *Repairing sparse low-rank texture*, in Proceedings of the 12th European Conference on Computer Vision, Part V, Lecture Notes in Comput. Sci. 7572, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 482–495.

[23] 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*, Technical report UILU-ENG-09-2214, University of Illinois at Urbana-Champaign, Urbana, IL, 2009.

[24] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, *Robust recovery of subspace structures by low-rank representation*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), pp. 171–184.

[25] G. Liu, Z. Lin, and Y. Yu, *Robust subspace segmentation by low-rank representation*, in Proceedings of the 27th International Conference on Machine Learning (ICML), Omnipress, Madison, WI, 2010.

[26] R. A. Maronna, R. D. Martin, and V. J. Yohai, *Robust Statistics: Theory and Methods*, Wiley Ser. Probab. Stat., John Wiley & Sons, Chichester, UK, 2006.

[27] 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 Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV), Vol. 2, 2001, pp. 416–423.

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

[29] 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.

[30] G. Peyre, *Toolbox Non-Local Means*, http://www.mathworks.com/matlabcentral/fileexchange/13619 (27 June 2009).

[31] S. Roweis, *EM algorithms for PCA and sensible PCA*, in Proceedings of Neural Information Processing Systems, 1997.

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

[33] J. Salmon, Z. T. Harmany, C.-A. Deledalle, and R. Willett, *Poisson Noise Reduction with Non-Local PCA*, CoRR, abs/1206.0338, 2012.

[34] Y. Shen, Z. Wen, and Y. Zhang, *Augmented Lagrangian Alternating Direction Method for Matrix Separation Based on Low-Rank Factorization*, Technical Report TR11-02, Rice University, Houston, TX, 2011.

[35] M. Tao and X. Yuan, *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM J. Optim., 21 (2011), pp. 57–81.

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

[37] 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), Springer-Verlag, London, 2000, pp. 298–372.

[38] J. Tropp, I. Dhillon, R. Heath, and T. Strohmer, *Designing structured tight frames via an alternating projection method*, IEEE Trans. Inform. Theory, 51 (2005), pp. 188–209.

[39] P. Tseng, *Nearest q-flat to m points*, J. Optim. Theory Appl., 105 (2000), pp. 249–252.

[40] R. Vidal and R. I. Hartley, *Motion segmentation with missing data using power factorization and GPCA*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. 310–316.

[41] M. B. Wakin, *Manifold-Based Signal Recovery and Parameter Estimation from Compressive Measurements*, preprint, arXiv:1002.1247v1 [stat.ML], 2010.

[42] 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.

[43] A. E. Waters, A. C. Sankaranarayanan, and R. G. Baraniuk, *SpaRCS: Recovering low-rank and sparse matrices from compressive measurements*, in Proceedings of Neural Information Processing Systems (NIPS), Granada, Spain, 2011.

[44] T. Wiberg, *Computation of principal components when data are missing*, in Proceedings of the Second Symposium on Computational Statistics, Berlin, Physica-Verlag, Heidelberg, 1976, pp. 229–326.

[45] M. Yan, *Restoration of Images Corrupted by Impulse Noise Using Blind Inpainting and l_0 Norm*, UCLA CAM report 11-72, Department of Mathematics, University of California, Los Angeles, CA, 2011.

[46] G. Yu, G. Sapiro, and S. Mallat, *Image modeling and enhancement via structured sparse model selection*, in Proceedings of the 17th IEEE International Conference on Image Processing (ICIP), 2010.

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

[48] 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.

[49] Z. Zhou, X. Li, J. Wright, E. J. Candès, and Y. Ma, *Stable principal component pursuit*, in Proceedings of the International Symposium on Information Theory (ISIT), IEEE, New York, 2010.