Outlier Detection in the Framework of Dimensionality Reduction

Qiang Ye[a], Weifeng Zhi[b,*]

[a]*Department of Mathematics, University of Kentucky, Lexington, KY 40506, USA*
[b]*Department of Computer Science, University of California-Davis, Davis, CA 95616, USA*

Abstract

We propose an effective outlier detection algorithm for high-dimensional data. We consider manifold models of data as is typically assumed in dimensionality reduction/manifold learning. Namely, we consider a noisy data set sampled from a low-dimensional manifold in a high-dimensional data space. Our algorithm uses local geometric structure to determine inliers, from which the outliers are identified. The algorithm is applicable to both linear and nonlinear models of data. We also discuss various implementation issues and we present several examples to demonstrate the effectiveness of the new approach.

Keywords: Outlier detection, High dimensional data, manifold learning

1. Introduction

Outlier detection is a classical problem in data analysis that arises in a variety of applications. Outliers in a data set may exist because of abnormality of the subjects underlying the data, inclusion of data from different categories, or contamination/corruption of the data. Their identification is important and often a major task in data analysis. High-dimensional data is typically analyzed with a dimensionality reduction technique, but most existing dimensionality reduction methods are very sensitive to the existence of outliers; namely existence of a single outlier point could significantly distort

*Corresponding author
Email addresses: qye3@uky.edu (Qiang Ye), wzhi@cs.ucdavis.edu (Weifeng Zhi)
[1]Research supported in part by NSF under Grant DMS-1317424.

the result of dimensionality reduction. Thus, while dimensionality reduction can usually detect some underlying patterns/structures of data, they are not apt to deal with outliers. In this paper, we are concerned with outlier detection for high-dimensional data that admits a certain low-dimensional structure and is well suited for analysis through dimensionality reduction.

There have been many techniques developed in the literature for the task of outlier detection; see the recent book [1] for a comprehensive survey. Many of these methods identify outliers as a by-product of another data mining task such as clustering or dimensionality reduction. Proximity-based methods [1, Chapter 4] use relative distances among data points or density to detect outliers. For high dimensional data, they may suffer from *curse of dimensionality*, where points with a few outlying components are masked by the noise effects of high dimensions; see [2]. Various methods have been developed to address this issue such as the approach of selecting and projecting on abnormal lower dimensional spaces [3, 4, 5, 6, 7]; see [2, 8, 9] for some others.

Another class of outlier detection methods is based on linear models of data where data points are assumed to lie approximately on a low dimensional plane. Principal component analysis (PCA) is a major dimensionality reduction algorithm that uncovers linear relations among data points, and there is a large body of literature on robust PCA that aims at overcoming the sensitivity of PCA to the existence of outliers; see [1, 10, 11, 12, 13, 14]. One approach is to estimate the robust covariance matrix of a data set first and then perform classical PCA on the covariance matrix estimator; see [15] for example. Another approach is to compute the robust principal directions by maximizing the covariance of the projected data set; see [16]. Such approaches do not attempt to detect the outliers of the data set but rather to robustly compute the principal component space and principal components in the presence of outliers. One difficulty associated with such robust PCA methods is that their computational efficiency deteriorates drastically as the size of data set and the dimension of data increase. A recent approach, called outlier pursuit, is to first detect outliers by minimizing the so-called nuclear norm of the data matrix subject to perturbations in the columns corresponding to outliers [17]. This is adapted for outlier detection from another robust PCA method called PCA pursuit [12, 18] which minimizes the nuclear norm of a data matrix subject to a sparse perturbation. Such methods have the effect of minimizing the rank of the data matrix and, as a convex optimization problem, they can be more efficiently solved [12, 17, 18].

For data that does not admit a linear relation, nonlinear dimensionality reduction algorithms [19, 20, 21, 22, 23] introduced in the last decade are effective alternatives but they are also very sensitive to the presence of outliers. This is the problem discussed in [24, 25] in the context of the Local Linear Embedding (LLE) method [19]. In [24], the standard construction of local linear structure in LLE is replaced by a robust PCA that is based on iteratively weighted least squares. In [25], outliers are detected by testing density (or gaps in pairwise distances) in local neighborhoods. There are other works dealing with small random noise in nonlinear dimensionality reduction problems; see [26, 23]. We note that all these methods appear to be intended for some specific nonlinear dimensionality reduction methods or problems.

In this paper, we present a new approach towards outlier detection for high-dimensional data. We use the prevailing data model in dimensionality reduction/manifold learning. Namely, we assume that the given data set is sampled from a low-dimensional manifold (linear or nonlinear) $\mathcal{M}$ embedded in a high-dimensional space $\mathbb{R}^m$ as defined by a smooth mapping $f : \mathcal{C} \subset \mathbb{R}^d \to \mathcal{M} \subset \mathbb{R}^m$, where $\mathcal{C}$ is a compact and connected subset of $\mathbb{R}^d$. Let $\{x_1, \cdots, x_N\} \subset \mathbb{R}^m$ be the data set. Then, inlier points of the set are those that can be written as

$$x_i = f(\tau_i), +\epsilon_i, \quad i = 1, \ldots, N, \tag{1}$$

where $\tau_i \in \mathcal{C}$ and ϵ_i represents small noise (i.e. a small number relative to $f(\tau_i)$)). If f is sufficiently smooth and sample points are sufficiently dense, then neighboring inlier points lie approximately on a d-dimensional plane. Thus, instead of detecting outliers directly, we attempt to identify inliers through this low-dimensional structure, i.e whether it is in a small neighborhood lying approximately on a d-dimensional plane. Specifically, we construct a collection of subsets of data, typically small local neighborhoods such as k-nearest point neighborhoods, and examine their geometric structure as characterized by the intrinsic dimension. Then neighborhoods with outlying geometric structures are deemed to contain outliers whereas the remaining contain inliers only. In this way, the inliers are determined, from which the outliers are detected. Our method applies to both linear and nonlinear models of data and can be easily combined with a dimensionality reduction method to further process the data.

The paper is organized as follows. We present our outlier detection method in Section 2. We then discuss its combination with a dimension-

ality reduction algorithm for further analysis in section 3. We finally present several examples to demonstrate the effectiveness of the method in Section 4 and conclude with some remarks in Section 5.

2. Outlier Detection

Consider a given data set $\boldsymbol{X} = \{x_1, \cdots, x_N\}$ modeled by (1) but contaminated with some outliers. Then, the inliers are points satisfying (1) and they can be characterized by the local low-dimensional structure. Namely, if points in a small neighborhood are all inliers satisfying (1), then they approximately span a d-dimensional plane (i.e. after centering, they approximately span a d-dimensional subspace). Indeed, if a local neighborhood approximately lies on a d-dimensional plane, then every point can be defined though (1) with a suitably small noise term and smooth f and hence can be considered an inlier point.

For each $x_i \in \boldsymbol{X}$, let $\boldsymbol{X}_i = \{x_{i_1}, \ldots, x_{i_k}\}$ be the k-nearest point neighborhood of x_i including itself, i.e. $x_{i_1}, \ldots, x_{i_k}$ are the nearest k data points from $\boldsymbol{X}$ to x_i in Euclidean distance. We can also use the ϵ-ball neighborhood that consists of x_j such that $\|x_j - x_i\| \leq \epsilon$ (where $\epsilon > 0$ is a given parameter) but we shall restrict our discussion to k-nearest point neighborhoods. Let $X_i = [x_{i_1}, \ldots, x_{i_k}]$ and let $\bar{x}_i = X_i e/k$ be the mean of the points in $\boldsymbol{X}_i$, where $e \in \mathbb{R}^k$ is the column vector of all ones. Let the SVD decomposition of $X_i - \bar{x}_i e^T$ be

$$X_i - \bar{x}_i e^T = U_i \Sigma_i V_i^T, \tag{2}$$

where $U_i \in \mathbb{R}^{m \times m}$ and $V_i \in \mathbb{R}^{k \times k}$ are orthogonal and $\Sigma_i = \operatorname{diag}\left(\sigma_1^{(i)}, \ldots, \sigma_k^{(i)}\right) \in \mathbb{R}^{m \times k}$ with $\sigma_1^{(i)} \geq \sigma_2^{(i)} \geq \cdots \geq \sigma_k^{(i)} \geq 0$. Then separating out the first d columns of U_i on the right hand side of (2), we have

$$\|X_i - \bar{x}_i e^T - U_i^{(d)} \Sigma_i^{(d)} (V_i^{(d)})^T\|_2 = \sigma_{d+1}^{(i)},$$

where $U_i^{(d)}$ and $V_i^{(d)}$ are the matrices consisting of the first d columns of U_i and V_i respectively and $\Sigma_i^{(d)} = \operatorname{diag}\left(\sigma_1^{(i)}, \ldots, \sigma_d^{(i)}\right)$. Therefore, if $\sigma_{d+1}^{(i)}$ is small, each column of X_i is approximately $\bar{x}_i$ plus a linear combination of the columns of $U_i^{(d)}$, i.e. it lies approximately in the plane through $\bar{x}_i$ and spanned by the columns of $U_i^{(d)}$. So, $\sigma_{d+1}^{(i)}$ measures whether $\boldsymbol{X}_i$ approximately lies on a d-dimensional plane and hence whether it consists entirely of inliers.

However, it is difficult to detect this by examining the magnitude of $\sigma_{d+1}^{(i)}$, as placing a right threshold depends on the distribution of $\sigma_{d+1}^{(i)}$ with respect to i.

We consider instead the set of neighborhoods $\{\boldsymbol{X}_i : 1 \leq i \leq N\}$. We assume that most of $\boldsymbol{X}_i$ contains inliers only. This happens when the number of outliers is not too large or when the outliers are distributed so that most k-nearest neighborhoods of inliers do not contain outliers. Then a neighborhood containing outliers will be an outlier of the set of neighborhoods $\{\boldsymbol{X}_i\}$ in the sense that it has an outlying geometric structure. To determine which neighborhood has an outlying structure, we consider the set of $(d+1)$-st largest singular values $\{\sigma_{d+1}^{(i)}, i = 1, \ldots, N\}$. Then the outlying neighborhoods $\boldsymbol{X}_i$ correspond to outlying singular values $\sigma_{d+1}^{(i)}$ in $\{\sigma_{d+1}^{(i)}, i = 1, \ldots, N\}$.

Outliers in $\{\sigma_{d+1}^{(i)}, i = 1, \ldots, N\}$ can be detected using a well-established statistical criterion. Since the set of singular values are nonnegative and most values are near 0, we find Hampel's method [27] work well for this task. Using the median and the median absolute deviation (MAD) as robust estimates of the location and the spread of the data set. The Hampel identifier is often found to be practically very effective. We apply Hampel's method to $\{\sigma_{d+1}^{(i)}, i = 1, \ldots, N\}$ as follows (see [27] for details). First we compute the median and the median absolute deviation (MAD) as

$$M_\sigma = \text{med}_i\{\sigma_{d+1}^{(i)}\} \tag{3}$$

and

$$\hat{\sigma}_{MAD} = \text{med}_i\{|\sigma_{d+1}^{(i)} - M_\sigma|\}. \tag{4}$$

Then the standard deviation σ is estimated as $\sigma \approx c\hat{\sigma}_{MAD}$, where $c = 1.4826$. We consider $\sigma_{d+1}^{(j)}$ an outlier of $\{\sigma_{d+1}^{(i)}, i = 1, \ldots, N\}$ if

$$\sigma_{d+1}^{(j)} > M_\sigma + 3c\hat{\sigma}_{MAD}. \tag{5}$$

This in turn determines the set of outlying neighborhoods, namely those neighborhoods containing at least one outlier.

The set of inlying neighborhoods is $\mathcal{M} = \{\boldsymbol{X}_i : \sigma_{d+1}^{(i)} \leq (M_\sigma + 3c\hat{\sigma}_{MAD})\}$. Its union $\bigcup_{\boldsymbol{X}_i \in \mathcal{M}} \boldsymbol{X}_i$ consists of inliers only. If every inlier is contained in a neighborhood consisting entirely of inliers, then the remaining points $\boldsymbol{X} \setminus \left(\bigcup_{\boldsymbol{X}_i \in \mathcal{M}} \boldsymbol{X}_i\right)$ are outliers. We present the outlier detection method in Algorithm 1.

Algorithm 1 Outliers Detection Algorithm

Input: $\boldsymbol{X} = \{x_1, \cdots, x_N\} \subset \mathbb{R}^m$; intrinsic dimension d, and $k \geq d+1$

1. For each x_i, construct its k-nearest neighborhood $\boldsymbol{X}_i = \{x_{i_1}, \ldots, x_{i_k}\}$;
2. For each $\boldsymbol{X}_i$, compute $\sigma_{d+1}^{(i)}$ through (2);
3. For $\{\sigma_{d+1}^{(i)}, i = 1, \ldots, N\}$, compute M_σ by (3) and $\hat{\sigma}_{MAD}$ by (4).
4. Output: outliers are contained in $\boldsymbol{X} \setminus \left(\bigcup_{\sigma_{d+1}^{(i)} \leq M_\sigma + 3c\hat{\sigma}_{MAD}} \boldsymbol{X}_i \right)$;

The algorithm takes the size of local neighborhoods k as an input. This need to be at least $d+1$. We have found that $k = d+5$ is usually sufficient. If we also want to find a low dimensional parametrization using a nonlinear dimensionality reduction algorithm, then the local neighborhoods also need to have sufficient overlaps [28], in which case a slightly bigger k may be needed. Also, the algorithm requires the intrinsic dimension d of the manifold. If this is not available, we can determine it from the local geometric structures. We present the details in Subsection 2.1 below.

The most computationally intensive part of of our algorithm is to form local neighborhoods, which requires computing pairwise distances and sorting. However, in the context of nonlinear dimensionality reduction, this is a step required by most methods such as the Local Linear Embedding (LLE) method and the Local Tangent Space Alignment (LTSA) method. We note that for certain very high dimensional data, the Euclidean distance between points may be dominated by insignificant components of data points in a phenomenon known as *curse of dimensionality* [2]. In that case, the k-nearest neighborhoods may be more or less random. However, our algorithm does not critically depend on the k-nearest neighborhood property as long as the points selected for the neighborhoods span approximately a d-dimensional plane, which we will assume to be the case for the present work. For example, if the underlying manifold is linear, then neighborhoods we construct need not be local. In this case, we can avoid the construction of k-nearest neighborhoods by using random neighborhood (or subset). We discuss such a method in Subsection 2.2.

2.1. Determination of the Intrinsic Dimension

Algorithm 1 requires an input parameter of the intrinsic dimension d. In many problems, d or a guess is available. When d is not available, we propose to determine it from the set of local neighborhoods $\{\boldsymbol{X}_i : 1 \leq i \leq N\}$ as follows. If $\boldsymbol{X}_i$ does not contain any outliers, then there is a large gap between the singular values $\sigma_d^{(i)}$ and $\sigma_{d+1}^{(i)}$, unless the points in $\boldsymbol{X}_i$ happen to be lying on a lower dimensional plane. However, if there are outliers in $\boldsymbol{X}_i$, then this gap disappears. To overcome the difficulty brought by the presence of outliers, we can use the medians of the singular values to exclude those neighborhoods containing outliers as well as those that happen to lie on a lower dimensional plane. Then, if we assume that at least half of the neighborhoods do not contain any outlier, then the median singular values $\mu_\ell = \text{med}_i\{\sigma_\ell^{(i)}\}$ will be given by some neighborhoods consisting of inliers only. In particular, μ_d is expected to be large relative to μ_{d+1}. We use this criterion to determine d and we present the process as the following algorithm, where *gap* is an input parameter for the gap.

Algorithm 2 Computing intrinsic dimension d

Input: $\boldsymbol{X} = \{x_1, \cdots, x_N\} \subset \mathbb{R}^m$; k and *gap*;
Output: d

1. For each x_i, construct its k-nearest neighborhood $\boldsymbol{X}_i = \{x_{i_1}, \ldots, x_{i_k}\}$;
2. For each $\boldsymbol{X}_i$, compute $\sigma_1^{(i)}, \cdots, \sigma_k^{(i)}$ through (2);
3. Compute $\mu_\ell = \text{med}_i\{\sigma_\ell^{(i)}\}$ for $\ell = 1, \ldots, m$.
4. If $\mu_\ell/\mu_{\ell+1} \leq gap$ for all ℓ, $k = k + 5$ and repeat steps 1-4; otherwise, d is the smallest ℓ that satisfies $\mu_\ell/\mu_{\ell+1} > gap$;

Algorithm 2 may be used preceding Algorithm 1 to find the input parameter d. Algorithm 2 requires k and the threshold *gap* as inputs. The initial input of k is not critical as it is adaptively updates at Step 4. For the choice of *gap*, we find that $gap = 10^6$ or a value around that works well for data that clearly admits a low-dimensional structure. The choice may be affected by the size of noise in the data. For data from real applications that may not have a clear low-dimensional structure, a good choice of threshold is less obvious and one may need to invoke trial-and-error.

2.2. Random Neighborhood for Linear Manifolds

One main computational cost of Algorithm 1 is to form local neighborhoods (subsets) $\boldsymbol{X}_i = \{x_{i_1}, \ldots, x_{i_{k_i}}\}$. If f is a linear map, then a low-dimensional structure exists in any subset of $\boldsymbol{X}$, as long as it does not contain outliers. Therefore, the composition of $\boldsymbol{X}_i$ needs not be constrained to local points. With this observation, we propose to construct $\boldsymbol{X}_i$ in step 1 of Algorithm 1 (and Algorithm 2 as well) by selecting x_i and $k-1$ other distinct points randomly from $\boldsymbol{X}$. We call them k-random neighborhoods.

This construction of k-random neighbors takes negligible amount of computing time and significantly improves the computational efficiency of our algorithm. One drawback of this method is that it can only deal with data with a small percentage of outliers as the probability that a k-random neighbor contains an outlier increases very quickly as the density of the outliers increases. Additionally, as d increases, $k \geq d+1$ needs to increase which in turn also increases the probability that a k-random neighbor constructed contains an outlier. However, for data with a small percentage of outliers and a modest intrinsic dimension d, the construction of k-random neighbors provides an efficient way for linear models of data.

2.3. Performance

Finally in this section, we comment on the performance of Algorithm 1. First, the success of our algorithm critically depends on the set of neighborhoods constructed. Indeed, the assumption underlying our algorithm is that the neighborhoods containing outliers constitute outlying ones in the set of neighborhoods $\{\boldsymbol{X}_i\}$. This certainly holds if only a small percentage of data points are outliers. This can still be true even when there is a large percentage of outliers, as long as they are clustered together.

Another factor affecting the quality of the neighborhoods is its size k. With larger k (i.e. more points in a neighborhood), the probability that a neighborhood containing outliers increases. Thus, the intrinsic dimension of the problem also affects the performance of our method through k. This is even more pronounced for k-random neighborhoods, as the probability that a neighborhood containing outliers is entirely determined by k and the percentage of outliers in the data set (see also previous subsection), where for the k-nearest neighborhood, it also depends on the distribution of outliers.

A less critical assumption we make is that each inlier is contained in a neighborhood consisting entirely of inliers (inlying neighborhood). This

implies that $\boldsymbol{X} \setminus \left(\bigcup_{\boldsymbol{X}_i \in \mathcal{M}} \boldsymbol{X}_i \right)$ is the set of outliers. If this assumption is not true, then some inlier points may be in $\boldsymbol{X} \setminus \left(\bigcup_{\boldsymbol{X}_i \in \mathcal{M}} \boldsymbol{X}_i \right)$ and misclassified as outliers. However, this can be remedied by checking for every point in $\boldsymbol{X} \setminus \left(\bigcup_{\boldsymbol{X}_i \in \mathcal{M}} \boldsymbol{X}_i \right)$ whether it approximately lies on the plane spanned by its k-nearest inliers. We discuss the details in the next section.

3. Dimensionality Reduction

For the dimensionality reduction problem where a low-dimensional representation of the data is desired, we can first apply the outlier detection algorithm (Algorithm 1) to identify and remove the outliers. Then we can apply one of the standard dimensionality reduction algorithms to the set of inliers to obtain a low-dimensional parametrization of inliers. Indeed, we can combine this with Algorithm 1 to utilize most of the computations there. Consider, for example, the LTSA method [23] for nonlinear dimensionality reduction. After outliers are detected by Algorithm 1, we can collect those neighborhoods constructed in step 1 consisting entirely of inliers and use the SVD computed in step 2 to construct their local coordinates. Then we can obtain a global parametrization by aligning the local coordinates as discussed in [23, 28].

For each of the outliers, we can also construct a parametrization through a projection on a local neighborhood of inliers as follows. For each outlier x_j, let $\boldsymbol{X}_j = \{x_{j_1}, \ldots, x_{j_k}\}$ be the k-nearest point neighborhood of x_j consisting of inliers, i.e. $x_{j_1}, \ldots, x_{j_k}$ are the nearest k inlier points to x_j. Let $\tau_{j_1}, \ldots, \tau_{j_k} \in \mathbb{R}^d$ be the low-dimensional parameters constructed for them. We find the projection of x_j on the plane spanned by $\boldsymbol{X}_j$ through solving the linear least squares problem

$$\min_{v \in \mathbb{R}^k} \|x_j - [x_{j_1}, \ldots, x_{j_k}]v\|_2. \tag{6}$$

Then the projected parameter for x_j is $\tau_j = [\tau_{j_1}, \ldots, \tau_{j_k}]v$.

The recovered parametrization for the outlier is useful if the outlier arises from a large noise. The projection may also identify an inlier that may be mistakenly identified as an outlier, as discussed in Subsection 2.3. Specifically, if the residual value of (6) is small (i.e. in the order of the diameter of the neighborhood and noise), then x_j approximately lies on the plane spanned by $\boldsymbol{X}_j$ and should be considered an inlier. However, for all of our tests in Section 4, all outliers are detected without this step.

4. Numerical Examples

In this section, we present several examples to illustrate the performance of our algorithm. All tests were performed in MATLAB. We first consider an example of linearly structured random data, which is taken from [17].

Example 4.1. We generated three matrices $A \in \mathbb{R}^{m\times d}$, $B \in \mathbb{R}^{(N-q)\times d}$, and $C \in \mathbb{R}^{m\times q}$, where each entry of the matrices is a pseudo random number with standard normal distribution ($\mathcal{N}(0,1)$). Let $L = AB^T$ be the matrix whose columns form the data set of interest. Assume L is contaminated with C and let $X = [L, C]$ be the data matrix given. The goal of this test is to identify C as outliers. We use $N = 600$ and $m = 400$, and test our algorithm with d (intrinsic dimension) ranging from 1 to 30 and q (the number of outliers) from 1 to 300.

For this problem, we use Algorithm 2 to first determine an intrinsic dimension d and then apply Algorithm 1 with $k = d+5$ to detect outliers. For each fixed d and q, we generate 5 test matrices and we present the rate of success out of the 5 tests in Figure 4.1(a) where it is plotted against (q, d) in the gray scale with white denoting the 100% success rate and black denoting the 0% success rate. A test is successful when both the intrinsic dimension d is correctly determined by Algorithm 2 and the outliers are correctly identified by Algorithm 1. We see that our algorithms have 100% success rate for $q < 300$ (up to half of sample points) when $d \leq 5$. The success rate is still 100% for $q < 200$ when $d \leq 10$. As d further increases, we note that the size of neighborhoods ($k = d + 5$) increases, which increases the probability that a local neighborhood contains an outlier. Eventually, the algorithms break down when $d > 15$. We have also implemented the algorithms with k-random neighborhoods discussed in Section 2.2. In that case, we have 100% success rate for $d \leq 5$ and $q \leq 25$ but the results deteriorate quickly as d increases (the full results not plotted). This is attributed to the fact that, with larger d, more than half of k-random neighborhoods contain outliers; see Subsection 2.3 for some discussions on the performance.

We have compared our method with the robust PCA in [12, 17] which is based on minimizing the nuclear norm of the data matrix subject to a sparse perturbation. We have tested the version presented in [17] using column block-wise sparse perturbations to detect outlier columns and we present the corresponding result in Figure 4.1(b). This method requires choosing the Lagrange multiplier λ and we have found $\lambda = 0.4$ works best for this problem.

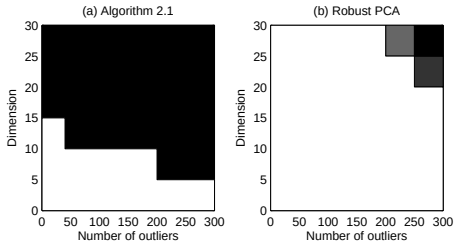

Figure 4.1: Example 4.1: Success rate out of 5 trials (white - 100%; black - 0%)

With this choice, the performance of the robust PCA is less dependent on the intrinsic dimension and can deal with problems with larger d. However, as might be expected, it is more computationally intensive. As an indicator, we compare the CPU time required by Algorithm 1 with k-nearest neighborhoods, Algorithm 1 with k-random neighborhoods, and the robust PCA algorithm of [17] in Figure 4.2. We fix $d = 5$, $m = 400$, $q = 20$, and increase the number of data points N from 500 to 2000. In Figure 4.2 (a), we plot the CPU time required by each method against N. It is clear that the computational time required by the robust PCA is significantly more than others and increase dramatically as N increases. The k-random neighborhoods method is most computationally efficient, as might be expected, but more importantly, its cost scales nearly linearly with N. We also test the algorithms with fixed $d = 5$, $N = 1000$, and $q = 20$ but varying m (the dimension of data points) from 200 to 800. We plot the CPU time used by each method in Figure 4.2 (b). A similar performance is observed although the k-random neighborhoods method does not offer much advantage over the k-nearest neighborhoods method.

From this synthetic data, we see that our methods work well within the constraints with respect to the size of neighborhoods and the density of outliers as discussed in Subsection 2.3. It compares favorably with the nuclear norm minimization based robust PCA for problems with small or modest intrinsic dimension d. The robust PCA developed specifically for linear problems, on the other hand, appears capable of handling data of very high intrinsic dimension. However, our algorithm is computationally more efficient and scale well with the problem size. In addition, our algorithm is applicable to nonlinearly modeled data.

We next consider noisy image data sets that can be modeled as 2-d nonlinear manifolds. We shall apply our algorithm to first detect outliers and then

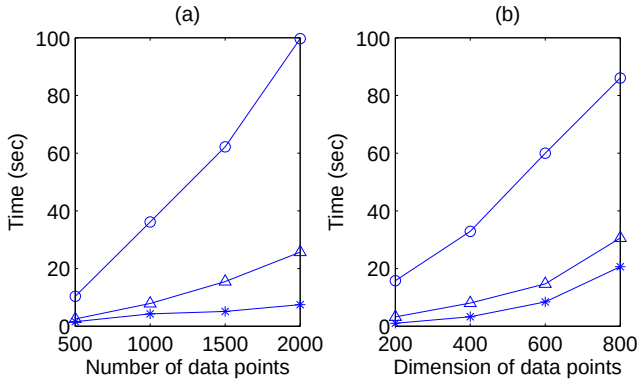

Figure 4.2: a) Computation time vs. N (number of data points); b) Computation time vs. m (dimension of data points). $\triangle$-line - Algorithm 1 with k-nearest neighbors; $*$-line - Algorithm 1 with k-random neighbors; $\circ$-line - robust PCA algorithm.

use LTSA [23] to find a low-dimensional parametrization. Our goal is to identify those images with large noise and find low dimensional parametrization for the data set. We compare our method with the robust nonlinear dimensionality reduction algorithm of [25], which uses a density-based approach to identify outliers and then apply a so-called linear EIV (error-in-variables) model together with a fusion step to the LLE method to derive a global estimate of noise-free coordinates.

Example 4.2. Consider a data set consisting of $N = 2000$ face images generated using the 3D face model in [29]. The 64×64 images are taken from the same face with varying pan and tilt angles for the observer. Figure 4.3(a) plots for each of the 2000 images its pan angle in the x-axis (from -30 to 30 degrees) and its tilt angle in the y-axis (from -10 to 10 degrees). We artificially introduce noise to each image by adding a random number in standard normal distribution to each pixel value. For 100 images, we also add salt-and-pepper noise (either 0 or 255) to 40% of the image pixels, resulting much distorted images. We show some selected images from each group in Figure 4.4, where the first row contains 10 noise-free images, the second row contains 10 images with small Gaussian noise, and the third row contains 10 images with large salt-and-pepper noise which come from the 10 clean images in the first row. In Figure 4.3(a), the coordinates of the 100 images perturbed by salt-and-pepper noise are labeled by red circles.

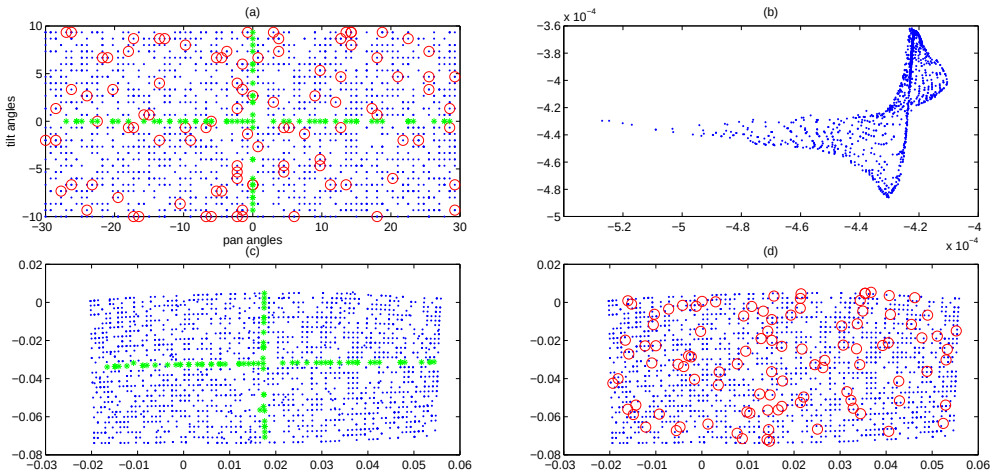

Figure 4.3: Example 4.2: (a) - original parameters; (b) - parameters recovered by LTSA; (c) - parameters recovered for inliers using Algorithm 1 and LTSA; (d) - parameters recovered for both inliers and outliers. Blue dot: 2-d parameters of images; Red circle: parameters for images with Salt-and-pepper noise (outliers); Green $*$: parameters for images with the original coordinates on the x-axis or the y-axis.

We are interested in identifying those with salt-and-pepper noise (as in the third row of Figure 4.4), considered outliers in this setting, and also recovering a 2-dimensional parametrization for these face images. We apply Algorithm 1 with $k = 10$ and $d = 2$ (known in this case) to detect outliers and then apply LTSA [23] to the inliers to find a parametrization. (The use of slightly larger k is to ensure sufficient overlaps among local neighborhoods that is required for LTSA [28].) Our algorithm exactly detects all the images with salt-and-pepper noise and the recovered parametrization for the remaining images are shown in Figure 4.3(c). To further visualize how Figure 4.3(c) correctly recovers the original parametrization Figure 4.3(a), we plot those points in the x-axis or the y-axis in 4.3(a) with a green $*$ and also plot in 4.3(c) the recovered coordinates for the corresponding images with a green $*$, demonstrating the affine transformation property of the reconstructed parametrization [28]. For the 100 outliers, we also construct an approximate parametrization as discussed in Sec. 3, which is shown with red circles in Figure 4.3(d). The parametrization constructed for the 100 images with salt-and-pepper noise is not expected to be accurate, but it is

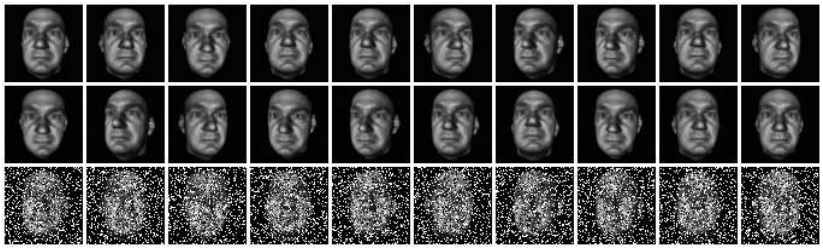

Figure 4.4: Example 4.2: First row - 10 typical original images; Second row - 10 typical images with the Gaussian noise $\mathcal{N}(0,1)$; Third row - images from row one perturbed by salt-and-pepper noise.

still within a small range of the original parametrization.

As a comparison, we also implement the LTSA algorithm using 10-nearest neighborhoods and $d = 2$ to find a 2-dimensional parametrization for these face images. The result is shown in Figure 4.3(b), where some coordinates with salt-and-pepper noise are out of range and not plotted. Without detecting outliers first, it is clear that the large outlier noise significantly distorts the results of LTSA.

We have also carried out a comparison with the robust nonlinear dimensionality reduction method of [25] and we have implemented it using 10-nearest neighborhoods and $d = 2$ to find a 2-dimensional parametrization for the face images. The robust nonlinear dimensionality reduction method identifies all 100 outliers but it also misidentifies 11 inliers as outliers. The identified outliers are shown in Figure 4.5(a) (in black +). In Figure 4.5(b), we plot the outliers identified by the robust nonlinear dimensionality reduction method [25] (in black +) and the outliers (in red circles), showing the 11 inliers misidentified as outliers by the former.

The robust nonlinear dimensionality reduction method [25] also applies a linear EIV model with fusion to the entire data set to recover a noise-free global parametrization. Figure 4.5(c) presents the result for this data set, with those corresponding to outliers marked in red circles. For comparison with our approach, we also consider first removing the outliers detected and then applying the linear EIV model with fusion to the remaining data set and we present the result in Figure 4.5(d). For this data set of images with salt-and-pepper noise, it is clear that our method outperforms the robust nonlinear dimensionality reduction method [25] in both outlier identification and parametrization recovery.

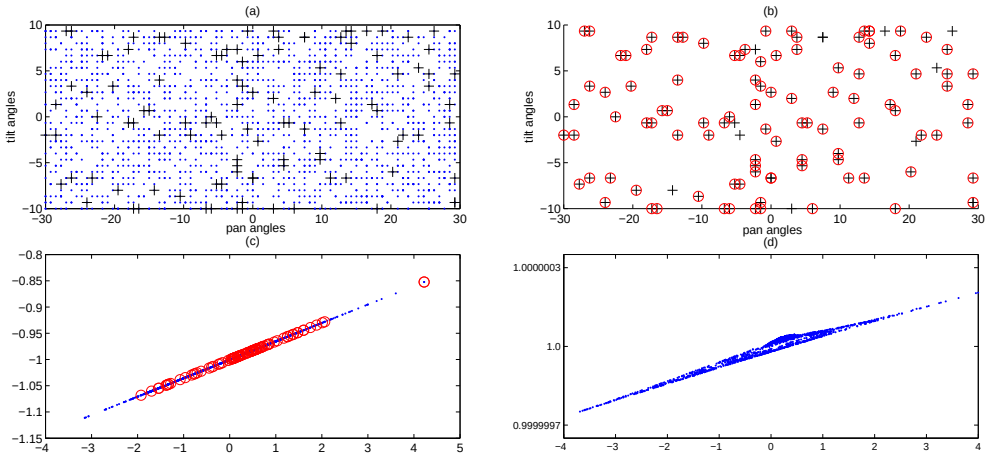

Figure 4.5: Example 4.2 (robust nonlinear dimensionality reduction method): (a) - original parameters and those identified as outliers; (b) - outliers identified vs. true outliers; (c) - parameters recovered using the linear EIV model with fusion for all images; (d) - parameters recovered using the linear EIV model with fusion with outliers removed. Blue dot: 2-d parameters of images; Red circle: parameters for images with Salt-and-pepper noise (outliers); Black +: parameters for outliers identified.

Example 4.3. Consider the same clean data set as in Example 4.2. We also artificially introduce Gaussian noise to each image by adding a random number in standard normal distribution to each pixel value but for 100 images, we also add large Gaussian noise ($\mathcal{N}(0, \sigma)$) to each pixel, where $\sigma = 20$ for our test. Our goal is to identify the 100 images with relatively large noise and then find the low-dimensional parametrization for all images. The original camera angles of these images are shown in Figure 4.6(a). The camera angles of images perturbed by the large Gaussian noise are labeled by red circles. We also show some selected images from each group in Figure 4.7.

Again, we apply Algorithm 1 with $k = 10$ and $d = 2$ to detect outliers and then apply LTSA [23] to the inliers to find a parametrization. Our algorithm correctly detects all the images with large Gaussian noise as outliers and correctly recovers the parametrization of the images with small Gaussian noise, which is shown in Figure 4.6(c). In Figure 4.6(a) and (c), the original points on the x-axis or the y-axis and the corresponding points respectively are plotted in green to demonstrate the recovered coordinates being an affine transformation of the original ones. Figure 4.6(d) includes the coordinates

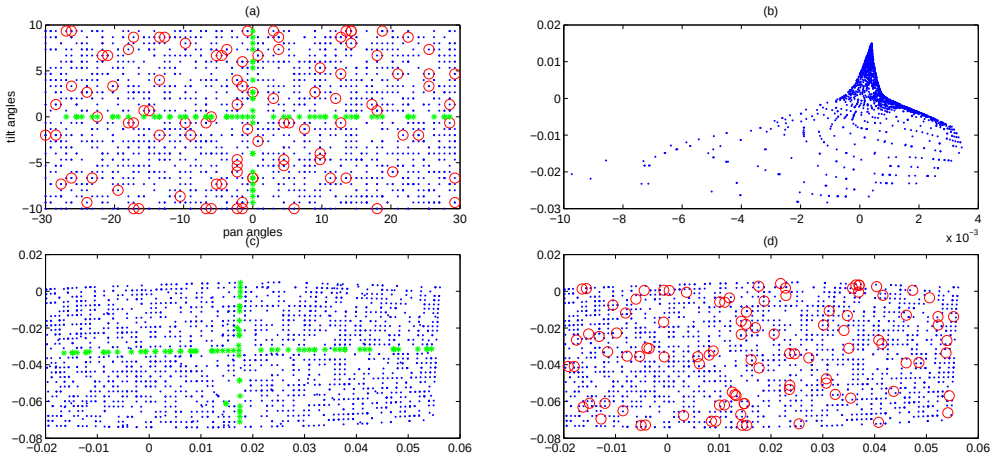

Figure 4.6: Example 4.3: (a) - original parameters; (b) - parameters recovered by LTSA; (c) - parameters recovered for inliers using Algorithm 1 and LTSA; (d) - parameters recovered for both inliers and outliers. Blue dot: 2-d parameters of images; Red circle: parameters for images with Salt-and-pepper noise (outliers); Green $*$: parameters for images with the original coordinates on the x-axis or the y-axis.

recovered for the images with large Gaussian noise and are labeled with red circles.

For comparison, we first implement the LTSA algorithm also using 10-nearest neighborhoods and $d = 2$. The result is shown in Figure 4.6(b), where some coordinates for large Gaussian noise are out of range and not plotted. Again, the large Gaussian noise significantly distorts the results of LTSA, even though the noise in these images is far less visible.

We also apply the robust nonlinear dimensionality reduction method in [25] to this dataset using 10-nearest neighborhoods and $d = 2$ to find 2-dimensional parametrization for the face images. Again, the robust nonlinear dimensionality reduction method identifies all 100 outliers but it also misidentifies 11 inliers as outliers. The identified outliers are shown in Figure 4.8(a) (in black $+$) as well as in Figure 4.8(b) where the coordinates of the outliers are plotted in red circles. The recovered parametrization using robust nonlinear dimensionality reduction method [25] in linear EIV model with fusion is shown in Figure 4.8(c) and the corresponding result obtained by removing the identified outliers first is shown in Figure 4.8(d). Again, our method outperforms robust nonlinear dimensionality reduction method

Figure 4.7: Example 4.3: First row - 10 typical original images; Second row - 10 typical images with the Gaussian noise $\mathcal{N}(0, 1)$; Third row - images from row one with the Gaussian noise $\mathcal{N}(0, 20)$.

in [25] in both outlier identification and parametrization recovery for this data set of images with large Gaussian noise.

Finally, we report the performance of Algorithm 1 on a data set from real applications that has no clear low dimensional geometric structure. Nevertheless, with a proper choice of parameters, our method detects the outliers correctly.

Example 4.4. We use the data from the USPS handwritten digits data set [30]. The experimental data set contains the first 140 samples of digit "0" and the first 10 samples of digit "4" in the USPS Handwritten digits database. The first 30 digit "0" and the 10 digit "4" are shown in Figure 4.9. The objective of this focused experiment is to identify all the "4" digits. With 10 samples out of a dataset of 150, the digit "4" can be considered as outliers of this data set. An underlying assumption of the experiment is that handwritten variations of the digit "0" approximately lie in a low-dimensional manifold, but one difficulty is that there is no clear intrinsic dimensionality. However, a projection to a 2-dimensional manifold ($d = 2$) has been commonly used in the literature (see [24]), which may also be a good starting point for our choice. Indeed, with $k = 12$ and either $d = 2$ or $d = 3$, Algorithm 1 successfully identifies the 10 samples of digit "4" from the digit "0" as outliers in this experiment.

We note that, for a difficult problem like this one, the performance of our method can vary with the choice of parameters. For example, with $d = 2$ and $k = 10$, all 10 samples of digits "4" are identified but 4 samples of digits "0" are also mislabeled as outliers. The 4 misidentified "0" are shown in the top row (panel (a)) of Figure 4.10. With $d = 3$ and $k = 10$, all 10 samples of

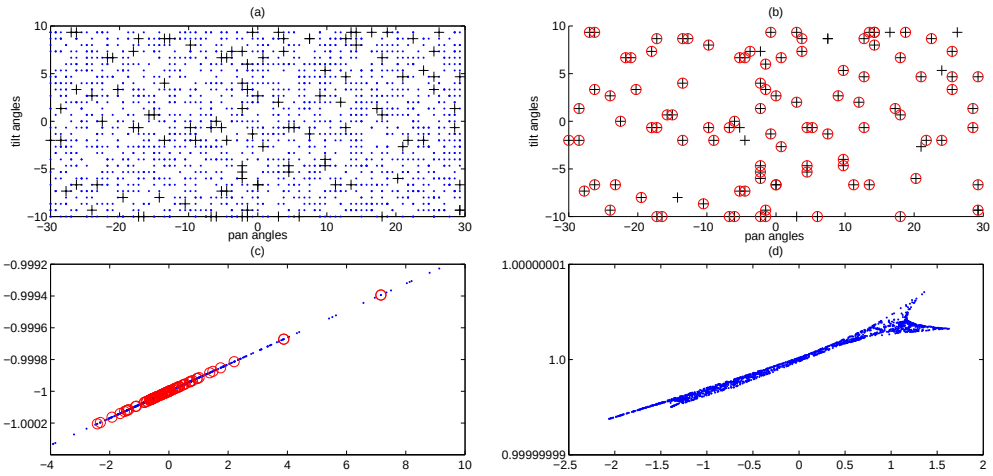

Figure 4.8: Example 4.3 (robust nonlinear dimensionality reduction method): (a) - original parameters and those identified as outliers; (b) - outliers identified vs. true outliers; (c) - parameters recovered using the linear EIV model with fusion for all images; (d) - parameters recovered using the linear EIV model with fusion with outliers removed. Blue dot: 2-d parameters of images; Red circle: parameters for images with Salt-and-pepper noise (outliers); Black +: parameters for outliers identified.

digit "4" are also identified but 2 samples of digits "0" are also mislabeled as outliers. The 2 misidentified "0" are shown in the bottom row (panel (b)) of Figure 4.10.

For comparison, we have also applied the robust nonlinear dimensionality reduction method [25] to this dataset with different dimensions ($d = 2, 3$) and different numbers of neighborhoods ($k = 10, 12$). In all cases, only 4 samples of digits "4" are identified and one sample of digits "0" is also mislabeled as outliers. Thus, our method has the advantage that for some commonly used parametric values, it is capable of successfully labeling all outlying samples, albeit it may misidentify some digits "0". We attribute this difficulty of misidentifying additional outliers to the problem itself where, while the digits "4" can clearly be considered outliers, some handwritten digits "0" (e.g. those in the bottom row of Figure 4.10) may significantly deviate from other samples of "0" in the dataset that their proper classification may be questionable.

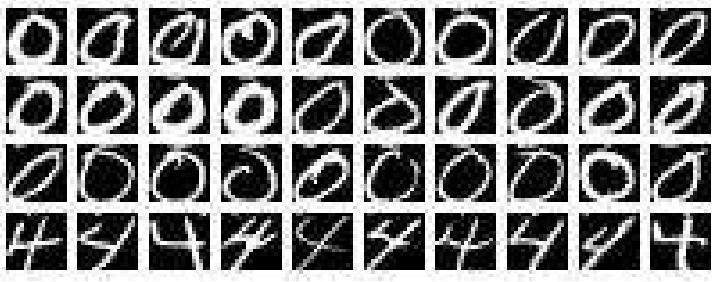

Figure 4.9: Example 4.4: first 30 digit "0" and the 10 digit "4".

5. Conclusion

We have proposed a new and effective algorithm to detect outliers of a data set that is modeled as lying on a low dimensional manifold. It is applicable to both linear and nonlinear models of data. In particular, the algorithms are computationally efficient and scales well with the problem size in terms of the dimension of date points and the number of data points. Our numerical tests demonstrate effectiveness of the algorithms.

One main feature of our approach is to indirectly characterize inliers or outliers through the set of neighborhoods. Thus, the success of our algorithm critically depends on the construction of neighborhoods. It will be our future work to carry out a theoretical study on performance of the algorithm and on characterization of inliers or outliers.

[1] C. C. Aggarwal, Outlier Analysis, Springer, New York, 2013.

[2] C. C. Aggarwal, P. S. Yu, Outlier detection for high dimensional data,

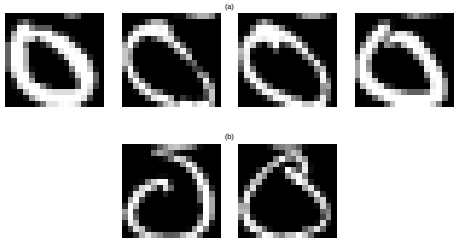

Figure 4.10: Example 4.4: misidentified digit "0". top (a): $d = 2$ and $k = 10$; bottom (b): $d = 3$ and $k = 10$

in: Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, 2001, pp. 37–46.

[3] H. Huang, H. Qin, S. Yoo, D. Yu, Local anomaly descriptor: a robust unsupervised algorithm for anomaly detection based on diffusion space., in: X. wen Chen, G. Lebanon, H. Wang, M. J. Zaki (Eds.), CIKM, ACM, 2012, pp. 405–414.

[4] F. Keller, E. Mller, K. Bhm, Hics: High contrast subspaces for density-based outlier ranking., in: A. Kementsietsidis, M. A. V. Salles (Eds.), ICDE, IEEE Computer Society, 2012, pp. 1037–1048.

[5] E. Mller, I. Assent, P. Iglesias, Y. Mlle, K. Bhm, Outlier ranking via subspace analysis in multiple views of the data., in: M. J. Zaki, A. Siebes, J. X. Yu, B. Goethals, G. I. Webb, X. Wu (Eds.), ICDM, IEEE Computer Society, 2012, pp. 529–538.

[6] T. S. E. Muller, M. Schiffer, Statistical selection of relevant subspace projections for outlier ranking, 2011.

[7] N. H. Vu, H. H. Ang, V. Gopalkrishnan, Mining outliers with ensemble of heterogeneous detectors on random subspaces., in: H. Kitagawa, Y. Ishikawa, Q. Li, C. Watanabe (Eds.), DASFAA (1), Vol. 5981 of Lecture Notes in Computer Science, Springer, 2010, pp. 368–383.

[8] C. C. Aggarwal, Outlier ensembles, SIGKDD Explor. Newsl. 14 (2) (2013) 49–58.

[9] A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD '05, ACM, New York, NY, USA, 2005, pp. 157–166.

[10] V. Hodge, J. Austin, A survey of outlier detection methodologies, Artificial Inetelligence Review 22 (2) (2004) 85–126.

[11] P. J. Rousseeuw, A. M. Leroy, Robust Regression and Outlier Detection, Wiley, New York, 1987.

[12] E. Candes, X. Li, Y. Ma, J. Wright, Robust principal component analysis?, Journal of the ACM 58 (1) (2011) 1–37.

[13] C. Croux, A. Ruiz-Gazen, High breakdown estimators for principal components: the projection-pursuit approach revisited, Journal of Multivariate Analysis 95 (2005) 206–226.

[14] L.Xu, A. Yuille, Robust principal component analysis by self-organizing rules based on statistical physics approach, IEEE Transactions on Neural Networks 6 (1) (1995) 131–143.

[15] C. Croux, G. Hasebroeck, Principal component analysis based on robust estimators of the covariance or correlation matrix: Influence functions and efficiencies, Biometrika 87 (3) (2000) 603–618.

[16] G. Li, Z. Chen, Projection-pursuit approach to robust dispersion matrices and principal components: Primary theory and monte carlo, Journal of the American Statistical Association 391 (80) (1985) 759–766.

[17] H.Xu, C. Caramanis, S. Sanghavi, Robust pca via outlier pursuit, IEEE Transactions on Information Theory 58 (5) (2012) 3047–3064.

[18] V. Chandrasekaran, S. Sanghavi, P. Parrilo, A. Willsky, Rank-sparsity incoherence for matrix decomposition, SIAM J. Optim. 21 (2) (2011) 572–596.

[19] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.

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

[21] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15 (2002) 1373–1396.

[22] D. Donoho, C. Grimes, Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data, Proceedings of National Academy of Sciences 100 (2003) 5591–5596.

[23] Z. Zhang, H. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, SIAM J. Sci. Comp. 26 (2005) 313–338.

[24] H. Chang, D.-Y. Yeung, Robust locally linear embedding, Pattern Recognition 39 (6) (2006) 1053–1065.

[25] H. Chen, G. Jiang, K. Yoshihira, Robust nonlinear dimensionality reduction by manifold learning, in: Proceedings of the International Conference on Pattern Recognition (ICPR), 2006, pp. 447–450.

[26] W. Wang, M. Carreira-Perpinan, Manifold blurring mean shift algorithms for manifold denoising, CVPR (2010) 1759 – 1766.

[27] F. R. Hampel, The influence curve and its role in robust estimation, Journal of the American Statistical Association 69 (1974) 383–393.

[28] Q. Ye, H. Zha, R.-C. Li, Analysis of an alignment algorithm for nonlinear dimensionality reduction, BIT-Numerical Mathematics 47 (2007) 873–885.

[29] V. Blanz, T. Vetter, A morphable model for the synthesis of 3d faces, in: SIGGRAPH, 1999, pp. 187–194.

[30] J. Hull, A database for handwriting text recognition research, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (5) (1994) 550–554.