Fast Low-Rank Subspace Segmentation

Xin Zhang, Fuchun Sun, *Member, IEEE,* Guangcan Liu, *Member, IEEE,* Yi Ma, *Fellow, IEEE,*

Abstract—Subspace segmentation is the problem of segmenting (or grouping) a set of n data points into a number of clusters, with each cluster being a (linear) subspace. The recently established algorithms such as Sparse Subspace Clustering (SSC), Low-Rank Representation (LRR) and Low-Rank Subspace Segmentation (LRSS) are effective in terms of segmentation accuracy, but computationally inefficient as it possesses a complexity of $O(n^3)$, which is too high to afford for the case where n is very large. In this paper we devise a fast subspace segmentation algorithm with complexity of $O(n\log(n))$. This is achieved by firstly using partial Singular Value Decomposition (SVD) to approximate the solution of LRSS, secondly utilizing Locality Sensitive Hashing (LSH) to build a sparse affinity graph that encodes the subspace memberships, and finally adopting a fast Normalized Cut (NCut) algorithm to produce the final segmentation results. Besides of high efficiency, our algorithm also has comparable effectiveness as the original LRSS method.

1 Introduction

Clustering is a basic step in data analysis and engineering. To segment (or cluster) a given set of data points, a common approach is the K-Means algorithm that aims at minimizing the sum of squared errors between data points and their nearest centers. K-Means could work well when each cluster possesses a "ball-like" structure. However, in many domains such as visual data, the data points of a cluster usually form a (linear) *subspace* that is an infinite set [2], [3], [4], [5], [6]. In this case, the algorithms (e.g., K-Means) based on (Euclidean) distances will fail, because: for two data points belonging to the same cluster, their distance can be arbitrarily large; for two data points that belong to different clusters, their distance can be arbitrarily small (see Figure 1). So, it is critical to investigate the challenging problem of *subspace segmentation* [1], whose goal is to segment data points into their respective clusters, with each cluster being a subspace. As a special clustering problem, subspace segmentation has its own specifics and requires professional tools to solve.

Subspace segmentation has been being an active direction in several areas such as machine learning, data mining, computer vision, image processing and system identification [2], [4], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20]. Most existing methods perform subspace segmentation by two steps: firstly learning an affinity matrix (i.e., an undirected graph) from the given data, and then obtaining the segmentation results by using spectral clustering algorithms such as the Normalized Cuts (NCut) [21]. The main difference among various methods is the approach for learning the affinity matrix that encodes the subspace memberships among data points. To ensure the success of subspace segmentation, ones need to obtain an affinity matrix S that satisfies the following two properties.

1. *Block-Diagonal*: If the i-th and j-th data points are from different subspaces, then $(S)_{i,j}=0$, where $(S)_{i,j}$ measures the similarity between the i-th and j-th data points[1].
2. *Strongly-Connected*: The data points belonging to the same subspace form a strongly connected graph. That is to say, if the i-th and j-th data points are from the same subspace, then there exist a series of data points with indices $\{i_1, i_2, \cdots, i_p\}$ such that $(S)_{i,i_1}>0$, $(S)_{i_1,i_2}>0$, $\cdots$, $(S)_{i_p,j}>0$.

Under the assumption that the data is noiseless and the subspaces are independent, Elhamifar and Vidal [7] show that the solution produced by Sparse Representation (SR) [22] satisfies the block-diagonal property. However, SR may not achieve the strongly-connected property, as analyzed in [23]. The recently established Low-Rank Representation (LRR) [16], [24] method and its variations (e.g., [5], [25]) guarantee to

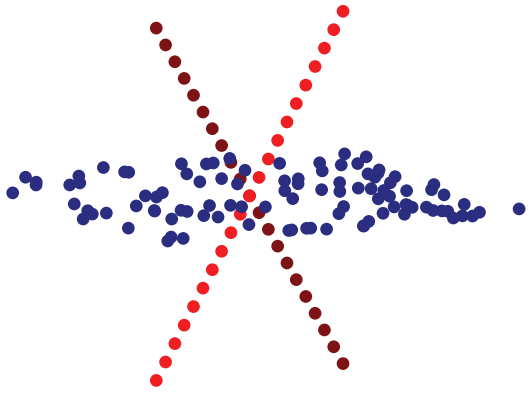

Fig. 1: **The specifics of subspace segmentation:** the within-subspace distances can be arbitrarily large, and the between-subspace distances can be arbitrarily small.

- *X. Zhang is with the Department of Computer Science and Technology, Tsinghua University, Beijing, China, 100084. E-mail: xinzhang1111@gmail.com*
- *F. Sun is with Tsinghua University.*
- *G. Liu is with University of Illinois at Urbana-Champaign.*
- *Y. Ma is with Microsoft Research Asia.*

1. This properties is called as "block-diagonal" because the affinity matrix S will be block-diagonal if the indices of the data points have been arranged to satisfy the true subspace memberships.

produce block-diagonal affinity matrices under the independent assumption. What is more, the affinity matrix produced by low-rankness based methods should also satisfy the strongly-connected property, although there is no proof in theory. Nevertheless, existing methods based on low-rankness modeling essentially need $O(n^3)$ computational complexity to segment a set of n data points, and thus it is not very competitive for dealing with large datasets.

In this paper, we present an efficient subspace segmentation algorithm based on Low-Rank Subspace Segmentation (LRSS) [25], which is designed for dealing with the data contaminated by Gaussian noise. According to the results in [25], LRSS has a closed form solution, which implies that the solution of LRSS can be approximated by using partial Singular Value Decomposition (SVD) [26] techniques. To process n data points of d-dimensional, the complexity of partial SVD is $O(nd)$. So, the desired affinity matrix can be obtained in an efficient way. However, the spectral clustering step is still time consuming, as the affinity matrix is often dense. To resolve this issue, we use a Locality Sensitive Hashing (LSH) mechanism [27] to build a k-sparse affinity matrix based on the dense affinity matrix produced by LRSS. This step spends $O(kn\log(n))$ time, where k is the number of non-zero entries in each row of the affinity matrix. Given a k-sparse affinity matrix, the NCut procedure has a complexity of $O(nk)$. Hence, the overall complexity of our algorithm is $O(n\log(n))$ with regard to the data points number n. Besides the high efficiency, the effectiveness of our algorithm is comparable to the original LRSS method, as demonstrated by our experiments.

2 Our Algorithm

In this section, we introduce our algorithm that consists of three procedures, including a partial SVD procedure to find the solution of LRSS, an LSH procedure for obtaining a k-sparse affinity matrix (i.e., building an undirected graph with sparse edges), and an NCut procedure for producing the final results.

2.1 Solving LRSS by Partial SVD

We consider the following convex optimization problem, termed LRSS, which is designed for the data contaminated by dense Gaussian noise:

$$\min_{Z,E} \|Z\|_* + \frac{\lambda}{2}\|E\|_F^2, \text{ s.t. } X = XZ + E, \tag{1}$$

where $\|\cdot\|_*$ denotes the *nuclear norm* [28] of a matrix, $\|\cdot\|_F$ is the Frobenius norm, and $\lambda > 0$ is a parameter. Let the SVD of X as $U\Sigma V^T$, then the optimal solution Z^* (with regard to the variable Z) to above problem has the following closed form [25]:

$$Z^* = V\mathcal{S}_\lambda(\Sigma)V^T,$$

where $\mathcal{S}_\lambda(\cdot)$ is a shrinkage operator defined by

$$\mathcal{S}_\lambda(t) = \begin{cases} 1 - \frac{1}{\lambda t^2}, & \text{if } t \geq \frac{1}{\sqrt{\lambda}}, \\ 0, & \text{otherwise.} \end{cases}$$

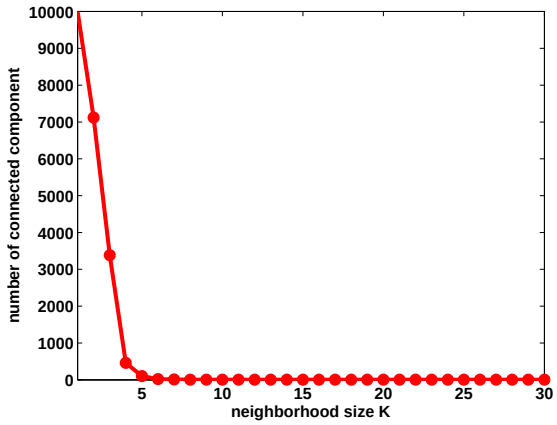

Fig. 2: **Illustrating that the strongly-connected property can be satisfied by k-sparse affinity matrices.** We generate a dataset containing 10000 data points ("dataset 1" of Section 3.1). There are 5 subspaces in total, and thus the number of connected components of the graph should be 5 in idea case. While $k \geq 6$, it is calculated that the number of connected components of the k-sparse graph is exactly 5.

The above discussions illustrate that the LRSS problem (1) can be efficiently solved by calculating a partial SVD [26] of X. Namely, suppose the rank-r SVD of X is $U_r\Sigma_r V_r^T$, then we may calculate the optimal solution of (1) by

$$\tilde{Z}^* = V_r\mathcal{S}_\lambda(\Sigma_r)V_r^T. \tag{2}$$

When $\lambda \leq 1/\sigma_{r+1}^2$ (σ_{r+1} denotes the $(r+1)$-th singular value of X), it is easy to see that the solution calculated by (2) is exactly the optimal solution to (1).

Note that the two parameters r and λ are not independent to each other. Instead, they are both determined by the intrinsic dimension of the underlying subspaces. So we treat r as a tunable parameter and automatically compute the parameter λ by

$$\lambda = \frac{1}{\sigma_{r+1}^2}. \tag{3}$$

The parameter r is the rank of the union of the underlying subspaces, and thus it should be chosen according to some prior knowledge about the data.

Suppose X contains n data points of d-dimensional, the complexity of computing its partial SVD is $O(drn)$. Provided that $d \ll n$ (note that $r \leq d$), this step can be made scalable to the case where n is very large.

2.2 Building a Sparse Affinity Matrix by LSH

Given the coefficient matrix $\tilde{Z}^*$ computed by (2), subspace segmentation can be done by firstly using Z^* to construct an undirected graph (the weight of the edge between the i-th and j-th nodes is $|(\tilde{Z}^*)_{i,j}|$), and then using NCut to segment the graph into a given number of m clusters. Nevertheless, such an approach is inefficient, because the coefficient matrix $\tilde{Z}^*$ is often dense and NCut will have a complexity of $O(n^2m)$.

Algorithm 1 Building a k-Sparse Affinity Matrix

input: $\{y_i\}_{i=1}^n$
parameters: l, b, ϵ, k.
construct l number of b-bit hash tables.
for $i = 1 \to n$ **do**
  compute the hash codes of y_i and $-y_i$.
  store y_i and $-y_i$ in each hash table according to their hash codes.
end for
for $i = 1 \to n$ **do**
  obtain a candidate set C_1 by using the algorithm in [27] to find $(1+\epsilon)k$ nearest neighbors of y_i.
  obtain a candidate set C_2 by using the algorithm in [27] to find $(1+\epsilon)k$ nearest neighbors of $-y_i$.
  merge together C_1 and C_2, denoted as $C = C_1 \cup C_2$.
  obtain k neighbors of the i-th data point by performing a linear search within C.
  compute the similarities between the i-th data point and its neighbors by using the absolute inner product measure.
end for
output: a k-sparse affinity matrix.

Moreover, the complexity of computing $\tilde{Z}^*$ by (2) is also $O(n^2 r)$.

Based on the fact that the graph built by $|\tilde{Z}^*|$ actually has many redundant edges – the segmentation results are dominated by the large values of $\tilde{Z}^*$, and thus it is unnecessary to involve all values of $\tilde{Z}^*$. Hence, we aim at building a k-sparse affinity matrix, each row of which has at most k nonzero entries, i.e., each node of the corresponding graph has at most k edges. A potential drawback of using a sparse affinity matrix is that the strongly-connected property might be destroyed. Usually, the graph connectivity can be usually preserved while the parameter k is slightly greater than the intrinsic dimension of the subspaces, as exemplified in Figure 2. For real datasets with complexity structures, one probably need to use relatively larger k. We shall investigate the influences of this parameter in the experiments.

To obtain a k-sparse affinity matrix from the given coefficient matrix $\tilde{Z}^*$, a straightforward approach is to select the k largest values from each row of $|\tilde{Z}^*|$. However, such an approach cannot be efficient when the number n is large. Actually, in order to obtain a k-sparse affinity matrix, it is unnecessary to compute the coefficient matrix $\tilde{Z}^*$, which essentially needs $O(n^2 r)$ complexity. This is because the value $(\tilde{Z}^*)_{i,j}$ (i.e., the (i,j)-th element of $\tilde{Z}^*$) is actually the inner product (denoted as $\langle\cdot\rangle$) between two vectors. More precisely, if we re-define each data point x_i as

$$y_i \doteq (V_r(\mathcal{S}_\lambda(\Sigma))^{\frac{1}{2}})_{i,:}, \tag{4}$$

where $(\cdot)_{i,:}$ denotes the i-th row of a matrix, then it is easy to see that $(\tilde{Z}^*)_{i,j} = \langle y_i, y_j\rangle$. In this way, the problem of building a k-sparse graph is equal to a "special" nearest neighbor search problem: given a set of r-dimensional data vectors $\{y_i\}_{i=1}^n$, the goal is to find the k nearest neighbors for each y_i. Unlike the common nearest neighbor search problems, in our problem the similarity for the pair (y_i, y_j) is defined according to the absolute value of $\langle y_i, y_j\rangle$, not $\langle y_i, y_j\rangle$ itself. To deal with this special condition, we assign two vectors, y_i and $-y_i$, for each data point x_i. In this way, we could use the existing similarity search techniques to build a k-sparse affinity matrix in an efficient way. In this paper we choose the LSH algorithm presented in [27].

Algorithm 2 Fast Subspace Segmentation

compute $X \approx U_r \Sigma_r V_r^T$ by partial SVD and calculate $\{y_i\}_{i=1}^n$ by (4)
construct a k-sparse graph by Algorithm 1.
use the NCut algorithm in [29] to segment the vertices of the graph into a given number of m clusters.

Algorithm 1 presents the whole procedure of building the sparse affinity matrix. The parameters l (number of hash tables) and ϵ are independent of the dataset size, and thus they can be fixed according to some experiences. In this work, we set them as $l = 10$ and $\epsilon = 9$. The parameter b, which is the length of hash codes, should be larger for larger datasets. According to [27], we automatically compute b by

$$b = \log_2(n). \tag{5}$$

It can be calculated that the complexity of constructing the hash tables is $O(nrbl)$, and the whole algorithm has a complexity of $O(nrbl) + O((1+\epsilon)kbn) + O((1+\epsilon)krn)$. Since the parameters r, l and k are independent of the dataset size n, it could be regarded that the complexity of Algorithm 1 is $O(n\log(n))$.

2.3 Producing the Segmentation Results by NCut

Given a k-sparse affinity matrix (there are in total nk edges in its corresponding graph), spectral clustering could be efficiently performed with a complexity of $O(nk)$. In this work, we choose the fast NCut algorithm presented in [29]. Algorithm 2 summarizes the whole process of our fast subspace segmentation algorithm. Combing the above analysis with the analysis in Section 2.1 and Section 2.2, it can be calculated that the overall complexity of our Algorithm 2 is $O(n\log(n))$, regarding to the dataset size n. When dataset size n is large, our algorithm will be significantly faster than the standard LRSS algorithm based on (1).

Besides of its high efficiency, it would be expectable that our algorithm is effective for subspace segmentation. When the subspaces are independent and the data is noiseless, it has been proven in several literatures (e.g., [24], [34]) that the coefficient matrix produced by (2) is block-diagonal. Also, the LSH algorithm can guarantee to find the true nearest neighbors with high probability. These imply that it has a high probability for our Algorithm 1 to output a block-diagonal affinity matrix, and thus our Algorithm 2 can be successful for subspace segmentation with high probability as well. For the complicate case where the subspaces are probably dependent and the data is of noisy, it is hard to establish theoretical guarantees for our algorithm, which is composed of three separated procedures.

TABLE 1: Some statistics of the three datasets used for experiments.

	data dimension	#of subspaces	# of data points
dataset 1	100	5	10000
dataset 2	1024	78	12480
dataset 3	100	4	16242

Empirically, our algorithms are effective in various conditions, as will be shown in our experiments.

3 EXPERIMENTS

In this section, three groups of comparison experiments are shown to verify the effectiveness of the proposed algorithm. The standard LRSS algorithm based on (1), Sparse Subspace Clustering (SSC) [7], Spectral Curvature Clustering (SCC) [35] and Local Subspace Analysis (LSA) [3] are chosen as the baselines.

3.1 Experimental Data

Three different datasets are used for experiments.

1. The first one is a synthetic databset (referred to as "dataset 1") generated in a similar was as in [16]. Concretely, we construct 5 subspaces whose bases $\{U_i\}_{i=1}^5$ are computed by $U_{i+1} = TU_i$, $1 \leq i \leq 4$, where T is a random rotation and U_1 is a random orthogonal matrix of size 100×5. So each subspace has a rank of 5. Then we sample 2000 data vectors from each subspace by $X_i = U_i Q_i$, $1 \leq i \leq 5$, where Q_i is a 5×2000 iid $\mathcal{N}(0, 1)$ matrix.
2. The second dataset ("dataset 2") consists of 12480 face images obtained as follows. First, we combine Extended Yale Database B [30] and the Olivetti face database [31], resulting in 78 face classes and 780 face images in total. Second, we use these 780 images as source samples to generate new samples. For each class, the new samples are generated by linear combination of its source samples. The combination coefficients are from iid $\mathcal{N}(0, 1)$.
3. The third dataset ("dataset 3") is a modified version of News20 [32]. In this dataset, there are 16242 100-dimensional data points belonging to 4 different subspaces. The dataset is downloaded form "http://www.cs.nyu.edu/∼roweis/data.html".

Table 1 summarizes some statistics about the three datasets used in our experiments.

3.2 Evaluation Metrics

The segmentation results can be evaluated in a similar way as classification results. Nevertheless, since segmentation methods cannot provide the class label for each cluster, a post-processing step is needed to assign each cluster a label. A commonly used strategy is to try every possible label vectors that satisfy the segmentation results. The final label vector is chosen as the one that best matches the ground truth classification results. Such a global search strategy is precise, but inefficient when the subspace number m is large. Namely, the computational complexity is $m!$, which is higher than 2^m for $m \geq 2$. Hence, we suggest a local search strategy as follows: given the ground truth classification results, the label of a cluster is the index of the ground truth class that contributes the maximum number of samples to the cluster. This local search strategy is quite efficient because its computational complexity is only $O(m)$, and can usually produce the same evaluation results as global search. Nevertheless, it is possible that two different clusters are assigned with the same label. So, we use the local search strategy only on the second dataset, which has 78 subpaces.

3.3 Results

On Choosing the Parameters. There are two parameters, r and k, need be tuned in our algorithm. The rank r should be determined via cross validation or using some prior knowledge. The neighborhood size k plays a crucial role in the trade-off between the efficiency and effectiveness of our algorithm. While k is very small, our algorithm can be very fast, but may not produce accurate segmentation. If k is set to be large, e.g., $k = n$, it is certain that our algorithm achieves the same segmentation accuracy as the original LRSS algorithm, but the advantages in efficiency will be lost. Figure 3 shows the influences of the parameter k on all the three datasets. On the first two datasets, in order to achieve the same accuracy as the original LRSS algorithm, the neighborhood size k can be small, saying $k \geq 5$ for "dataset 1" and $k \geq 10$ for "dataset 2". These results are consistent with the doctrine that it is enough to set k as the maximum of the dimensions of the subspaces (note that the subspace dimensions may not be uniform). However, to obtain comparable performance as LRSS on the third dataset, the parameter k needs to be greater than 380, which is much larger than the dimension of the subspaces (note that the ambient dimension is 100 and so the subspace dimension is 100 at most). This is because each cluster of the third dataset actually consists of several sub-clusters[2], and thus more edges are necessary for preserving the graph connectivity within the same cluster.

Comparison Results. Table 2 lists the results of our algorithm, SSC, LSA, SCC and LRSS. In terms of computational efficiency, our algorithm is much superior to SSC, LSA and LRSS. On the second dataset, the accuracy of SCC is only 7.1%, which is far behind the others. The reason might be the ambient dimension of this dataset is high. Unlike the first two datasets, SSC does not work well, only achieving an accuracy of 40%, which is much lower that the 73% produced by LRSS. This phenomenon is consistent with the results in Figure 3, which illustrates that the third dataset requires relatively larger k for our algorithm to work well. In summary, these results illustrate that our algorithm possesses distinct superiorities over the competitors for subspace segmentation, in terms of the comprehensive evaluation of both segmentation accuracy and computation time.

2. There are 20 news groups in News20. The modified version we used is created by merging 20 groups into 4 large topics.

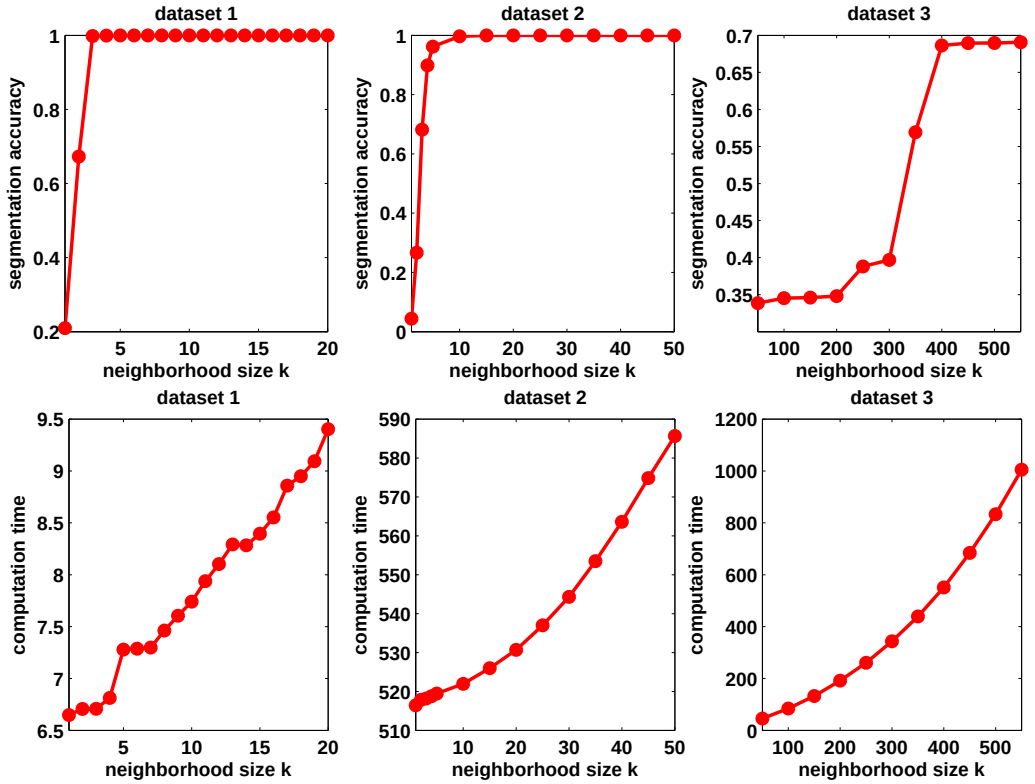

Fig. 3: **Influences of the parameter** k**.** Above: segmentation accuracy vs the neighborhood size k. Below: computational time vs the neighborhood size k. For the three datasets, the parameter r is chosen as 25, 300, 100, respectively. Note that the numbers shown above are averaged from 20 runs.

TABLE 2: **Segmentation accuracy (%) and computation time (seconds) on the three datasets.** The parameters of our algorithm are set as $r = 25, 300, 100$ and $k = 5, 10, 400$. The parameter λ for LRSS is set as in (3). The parameters of the other algorithms are manually turned to best in terms of segmentation accuracy. The numbers of our algorithm are averaged from 20 runs.

Method	DataSet 1		DataSet 2		DataSet 3	
	Accuracy(%)	Time(s)	Accuracy(%)	Time(s)	Accuracy(%)	Time(s)
SSC	100	25450.61	99.99	91140.21	39.98	36012.13
LSA	100	5534.36	81.39	21499.01	35.60	89089.04
SCC	100	29.15	7.1	19808.84	60.09	964.20
LRSS	100	76.05	100	2947.94	72.77	1405.90
Our	100	9.40	100	563.61	70.64	550.88

Robustness to Noise. Low-rankness based methods are known to be good at handling noisy data [24], [8]. As our algorithm is derived from LRSS, it would be natural to anticipate that it can be robust to noise. To verify this, we add to the first dataset some i.i.d Gaussian noise with zero mean and standard deviation σ (σ stands for the noise level). Figure 4 shows that our algorithm almost possesses the same ability as LRSS for dealing with noisy data. This is natural, because the process of noise is mainly performed by the first procedure (i.e., the partial SVD step). By setting λ as in (3), there is actually no difference between the coefficient matrices produced by our algorithm (i.e., (2)) and LRSS (i.e., (1)).

4 Conclusions and Future Work

In this work we addressed the problem of performing fast subspace segmentation in the case where the number of data points is large. By combining the techniques of LRSS, partial SVD and LSH, we devised a scalable subspace segmentation algorithm, which owns high efficiency and comparable effectiveness as LRSS.

Our basic observation is that the solution of the LRSS problem (1) has a closed form which can be approximated by partial SVD. However, such an approach is not applicable to the following LRR problem:

$$\min_{Z,E} \|Z\|_* + \lambda \|E\|_{2,1} \text{ s.t. } X = XZ + E,$$

which has more significant role than (1) in subspace segmentation. As shown in [33], it is possible to make the above problem scalable, and thus it would be feasible to devise an

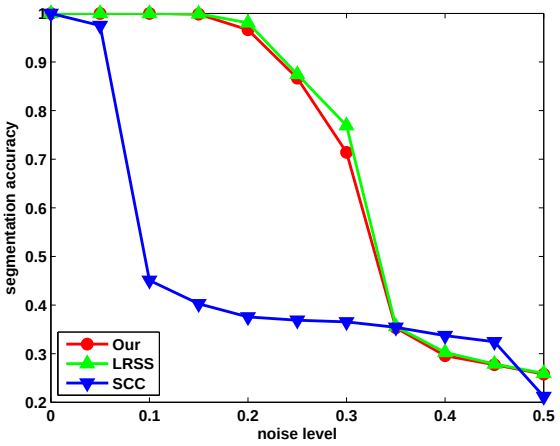

Fig. 4: **Robustness to dense Gaussian noise.** In these experiments, $r = 25$ and $k = 5$. The numbers of our algorithm are averaged from 20 runs.

efficient algorithm based on the above LRR formula. We leave this as a future work.

REFERENCES

[1] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.

[2] W. Gear, "Multibody grouping from motion images," *Int'l J. Computer Vision*, vol. 29, no. 2, pp. 133–150, 1998.

[3] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *Euro. Conf. Computer Vision*, vol. 4, 2006, pp. 94–106.

[4] S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1832–1845, 2010.

[5] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," in *IEEE Int'l Conf. Computer Vision*, 2011.

[6] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[7] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2009, pp. 2790–2797.

[8] G. Liu, H. Xu, and S. Yan, "Exact subspace segmentation and outlier detection by low-rank representation," in *Int'l Conf. Artificial Intelligence and Statistics*, 2012.

[9] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] J. Costeira and T. Kanade, "A multibody factorization method for independently moving objects," *Int'l J. Computer Vision*, vol. 29, no. 3, pp. 159–179, 1998.

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

[12] Z. Fan, J. Zhou, and Y. Wu, "Inference of multiple subspaces from high-dimensional data and application to multibody grouping," in *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 661–666.

[13] G. Liu, Z. Lin, X. Tang, and Y. Yu, "Unsupervised object segmentation with a hybrid graph model (HGM)," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 910–924, 2010.

[14] C. Zhang and R. Bitmead, "Subspace system identification for training-based MIMO channel estimation," *Automatica*, vol. 41, no. 9, pp. 1623–1632, 2005.

[15] T. Zhang, A. Szlam, and G. Lerman, "Median k-flats for hybrid linear modeling with many outliers," in *Workshop on Subspace Methods*, 2009.

[16] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Int'l Conf. Machine Learning*, 2010, pp. 663–670.

[17] E. Arias-Castro, G. Chen, and G. Lerman, "Spectral clustering based on local linear approximations," *Electronic Journal of Statistics*, vol. 5, pp. 1537–1587, 2011.

[18] G. Lerman and T. Zhang, "Robust recovery of multiple subspaces by geometric lp minimization," *arXiv:1104.3770*, 2011.

[19] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," *arXiv:1010.3460*, 2011.

[20] M. Soltanolkotabi and E. Candès, "A geometric analysis of subspace clustering with outliers," *arXiv:1112.4258v2*, 2011.

[21] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, pp. 888–905, 2000.

[22] D. Donoho, "For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution," *Comm. Pure and Applied Mathematics*, vol. 59, pp. 797–829, 2004.

[23] B. Nasihatkon and R. Hartley, "Graph connectivity in sparse subspace clustering," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2011, pp. 2137–2144.

[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 Analysis and Machine Intelligence*, vol. 35, pp. 171–184, 2013.

[25] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

[26] R. Larsen, "Lanczos bidiagonalization with partial reorthogonalization," *Department of Computer Science, Aarhus University, Technical report*, 1998.

[27] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: efficient indexing for high-dimensional similarity search," in *Proceedings of the 33rd international conference on Very large data bases*, ser. VLDB, 2007, pp. 950–961.

[28] M. Fazel, "Matrix rank minimization with applications," *PhD thesis*, 2002.

[29] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, 2007.

[30] K. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 684–698, 2005.

[31] A. H. Ferdinando Samaria, "Parameterisation of a stochastic model for human face identification," *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, 1994.

[32] K. Lang, "Newsweeder: Learning to filter netnews," in *International Machine Learning Conference*, 1995.

[33] G. Liu and S. Yan, "Active subspace: Towards scalable low-rank learning," *Neural Computation*, vol. 24, no. 12, pp. 3371–3394, 2012.

[34] S. Wei and Z. Lin, "Analysis and Improvement of Low Rank Representation for Subspace segmentation," *arXiv:1107.1561*, 2010.

[35] G. Chen and G. Lerman, Spectral curvature clustering (scc), *International Journal on Computer Vision*, vol. 81, pp. 317–330, 2009.