Two-Step Greedy Subspace Clustering

Lingxiao Song(✉), Man Zhang, Zhenan Sun, Jian Liang, and Ran He

Center for Research on Intelligent Perception and Computing,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
{lingxiao.song,zhangman,znsun,jian.liang,rhe}@nlpr.ia.ac.cn

Abstract. Greedy subspace clustering methods provide an efficient way to cluster large-scale multimedia datasets. However, these methods do not guarantee a global optimum and their clustering performance mainly depends on their initializations. To alleviate this initialization problem, this paper proposes a two-step greedy strategy by exploring proper neighbors that span an initial subspace. Firstly, for each data point, we seek a sparse representation with respect to its nearest neighbors. The data points corresponding to nonzero entries in the learning representation form an initial subspace, which potentially rejects bad or redundant data points. Secondly, the subspace is updated by adding an orthogonal basis involved with the newly added data points. Experimental results on real-world applications demonstrate that our method can significantly improve the clustering accuracy of greedy subspace clustering methods without scarifying much computational time.

Keywords: Greedy subspace clustering · Sparse representation · Subspace neighbor

1 Introduction

Clustering is a classic problem in multimedia and computer vision. As an important branch of clustering, subspace clustering seeks to cluster data into different subspaces and find a low-dimensional subspace fitting each group of points. It is based on the assumption that high-dimensional data often lies on low-dimensional subspaces, which usually holds true for the data acquired in real world. Subspace clustering can be widely applied in image segmentation [1], motion segmentation [2], face clustering [3], image representation, compression [4], and multimedia analysis [5,6]. In these applications, data points of the same class (e.g., pixels belong to the same object, feature points of the same rigid object in a moving video sequence, face images of the same person) lie on same underlying subspace, and the mixture dataset can be modeled by unions of subspaces.

1.1 Related Work on Subspace Clustering

Numerous subspace clustering approaches have been proposed in the past two decades. Existing work on subspace clustering in machine learning and computer vision communities can be divided into four main categories: algebraic, iterative, statistical and spectral clustering-based methods [7].

Algebraic methods such as matrix factorization-based algorithms [8,9] segment data points according to a low-rank factorization of the data matrix. But these methods are not effective when the subspaces are dependent. Generalized Principal Component Analysis (GPCA) [10] uses a polynomial function to fit a given point. It can handle both independent and dependent subspaces, but the computational complexity increases exponentially when the dimension of data grows. Such methods are sensitive to noise and outliers, due to their assumption of noise-free data. Iterative methods [3,11] iteratively refine subspaces of each cluster and assign points to the closest subspace. And these methods can be applied to linear as well as affine subspaces, but it is easy to run into a local optimum, thus several restarts are often needed. Statistical approaches such as [12,13], model both data and noise under explicit assumptions of the probabilistic distribution of data in each subspaces and noise. However, these statistical approaches are not suitable for real-world applications due to their sensitivity to outliers.

The standard procedure of spectral clustering-based methods consists of constructing an affinity matrix firstly whose elements measure the similarity between samples, and then applying spectral clustering given the affinity matrix. A number of spectral clustering-based methods spring out in recent years such as Sparse Subspace Clustering (SSC) [14,15], Low-Rank Representation (LRR) [16], Low-Rank Representation via Correntropy (CLRR) [17,18], Low-Rank Sparse Subspace Clustering (LRSSC) [19] and Spectral Curvature Clustering (SCC) [20]. The basic idea of SSC is that a data point can be written as a linear or affine combination of other data points in the same subspace under an l_1-minimization constraint. A similar optimization based method called LRR minimizes nuclear norm instead of the l_1-norm in SSC to guarantee a low-rank affinity matrix. CLRR proposed by Zhang et al. attempt to maximize the correntropy between data points and their reconstruction, and an efficient solution of the optimization problem based on half-quadratic minimization is given in their paper. Wang et al. propose a hybrid algorithm termed LRSSC by combining l_1-norm and nuclear norm, based on the fact that the representation matrix is often not only sparse but also low-rank.

More recently, greedy-like spectral clustering-based approaches gain increasing attention due to their low complexity. Dyer et al. induced a greedy method for sparse signal recovery called orthogonal matching pursuit (OMP) [21], which is used to replace the l_1-minimization in SSC. Heckel and H. Bölcskei proposed a simple algorithm based on thresholding the correlation between data points (TSC) [22]. Park and Caramanis presented Nearest Subspace Neighbor (NSN) in [23], which constructs neighbors via incrementally selecting point that is closest to the subspace. All the greedy approaches share a common advantage of computationally less demanding. However, a major disadvantage of these greedy-like methods is that they can not cope with complex situation where the subspaces intersect or lots of noise exist.

1.2 Paper Contributions

In this paper we propose a two-step greedy subspace clustering (T-GSC) algorithm, which is able to achieve superior clustering performance, comparing to several state-of-the-art methods. A initial subspace construction step is induced aiming to improve the robustness of greedy-like algorithms, especially when the data are not well distributed around their subspaces, e.g. the data is contaminated by outliers, the subspaces are intersected. After the first step, different-class neighbors can be directly ruled out in most cases, resulting in a better spanning of the subspace. Then, in the second step, the nearest subspace neighbor is added to the neighbors set in a greedy way, which requires much less run time than other kinds of algorithms. Numerous experimental results demonstrate that our algorithm can reach state-of-the-art clustering performance, with better robustness and lower computational cost.

The reminder of this paper is organized as follows. Section 2 reviews the Nearest Subspace Neighbor algorithm and then describes the technical details of our method. Then a number of experiments are presented in the Sect. 3. Finally we draw some conclusions in the Sect. 4.

2 Two-Step Greedy Subspace Clustering

As mentioned before, spectral clustering-based methods follow a basic procedure of computing an affinity matrix first and then deriving clusters using spectral clustering. Instead of computing each entry of the affinity matrix, we construct a neighborhood matrix whose entries are either 0 or 1, as in [22,23]. Hence, constructing an affinity matrix is transformed into finding a neighborhood of each data point. The proposed method is mainly inspired by the Nearest Subspace Neighbor (NSN), which iteratively selects neighbor points most likely to be on the same subspace. That is to say, NSN choose the nearest neighbor as the first member of the neighbor set to construct subspace. However, a data point and its nearest neighbor may belong to different true subspaces in many cases. Thus, we propose an initial subspace construction step in our algorithm to solve this problem.

Through this paper, we use uppercase boldface letters to denote matrices, lowercase letters to denote vectors and scalars, and letters that are uppercase but not bold stand for parameters and sets. The given data matrix is denoted as $X \in R^{D \times N}$, with each column representing a data point $x_i \in R^D$. The task of subspace clustering is to recover the union of K subspaces $S = S_1 \cup S_2 \cup ... \cup S_K$ where the data points are belong to, and the dimension of each subspace is supposed to be not higher than P. For each data point x_i, there is a neighbor set Ω_i contains M elements. In addition, U represents subspace spanned by a set of data points: $U = span(\Omega)$; $\text{proj}_U(x)$ is defined as the projection of data point x to the subspace U; $\eta_m, m = 1, 2, ..., P$ are orthogonal bases of U; $\mathbb{I}\{.\}$ is the indicator function; $\|.\|$ denote Euclidean norm for vectors; and $\langle . \rangle$ is the inner product operator.

2.1 First Step: Initial Subspace Construction

T-GSC constructs a good initial subspace based on the theory of sparse representation. Considering a data point $x_i \in R^D$ drawn from a union of subspaces, it can be sparsely represented by solving

$$\min \|c\|_0 \qquad s.t. \quad x_i = Xc, \tag{1}$$

where X is the data matrix. However, this optimization problem of l_0-norm is NP-hard and non-convex in general, while it is proved in [24] that the l_1-norm solution is equivalent to l_0-norm solution in certain conditions. So, the l_1-norm is adopted, and Eq. (1) can be reformulated as follows.

$$\min \|c\|_1 \qquad s.t. \quad x_i = Xc. \tag{2}$$

There are a huge number of approaches for extracting the sparse solution of Eq. (2) such as the Basis Pursuit Algorithm [24] and Lasso [25]. The work of SSC [9] proves that the optimal solution $\hat{c}$ has zero entries correspond to data points not lying in the same subspace with x_i. Based on this, neighbors from distinct subspaces can be rejected.

Fortunately, we do not need to find all data points that lie in the same subspace with x_i constructing the initial subspace. We just need to find one neighbor that most likely belongs to the same class with x_i to ensure the reliability of the subspace spanned by them. Therefore, optimizing Eq. (2) among all the data points is undesirable. In our approach, L nearest neighbors around x_i in the ambient space are chosen to be the bases. Then the l_1-optimization problem is simplified to

$$\min \|c\|_1 \qquad s.t. \quad x_i = X_i^{(L)}c. \tag{3}$$

where $X_i^{(L)}$ is comprised of the L nearest neighbors of x_i, and the problem can be efficiently solved when L is not large. We use the glmnet toolbox [25] of Lasso to solve this problem. The data point corresponding to the max entry of c will be the first neighbor added to x_i's neighbor set Ω_i, meanwhile the initial subspace U is spanned by Ω_i.

2.2 Second Step: Greedy Subspace Clustering

Once the initial subspace is built, T-GSC greedily adds the closest point to the subspace into the neighbor set in the following iterations as NSN, until enough neighbors are selected. This step mainly involves two stages: finding new neighbors and updating subspaces.

We use a set of orthogonal bases to represent the subspace. In every iteration, the subspace is updated by adding an orthogonal basis involved with the newly added data points according to the Gram-Schmidt process.

$$\eta_{m+1} = x_{j^*} - \sum_{k=1}^{m} \langle x_{j^*}, \eta_k \rangle \eta_k. \tag{4}$$

Then, $\text{proj}_U(x)$ can be easily obtained through the following equation. Note that we only need to compute the inner product with the newly added orthogonal basis in the k-th iteration.

$$\text{proj}_U(x)^2 = \sum_k \langle x, \eta_k \rangle^2. \tag{5}$$

After all the neighbors are selected, spectral clustering is applied to find the clusters as in other spectral-based algorithms. The following Algorithm 1 summarizes the whole procedure in T-GSC to cluster data points into different subspaces.

Algorithm 1. Two-step Greedy Subspace Clustering(T-GSC)

```
Input: Data matrix $X \in R^{D \times N}$, maximum subspace dimension $P$, number of neighbors $M$,
  number of subspaces $K$.
Output: Neighbor matrix $Z \in \{0,1\}^{N \times N}$. Estimated class labels $\hat{c}_1, \hat{c}_2, ..., \hat{c}_N$.
  1. Normalize all data points: $x_i \leftarrow x_i / \|x_i\|$;
  for each $x_i$ do
     2. (First step) Construct the initial subspace $U$ and initial neighbor set $\Omega_i$ by solving Eq.(3);
     3. (Second step) Find $M$ neighbors for $x_i$:
        (1). Select the closest point to current subspace: $j^* = \arg\max_{j \in [N] \setminus \Omega_i} \text{proj}_U(x_j)$;
        (2). Update the neighbor set: $\Omega_i \leftarrow \Omega_i \cup \{j^*\}$;
        (3). Update the subspace: if $M < P$, then $U \leftarrow span\{x_j : j \in \Omega_i\}$;
     4. Update the neighborhood matrix: $Z_{ij} = 1, \forall j \in \Omega_i$;
  end for
  5. Construct the affinity matrix: $W_{ij} = Z_{ij} + Z_{ji}$;
  6. Apply spectral clustering to $(W, K)$.
```

3 Experiments

This section presents experimental results of our study. We compare our method with several state-of-the-art methods on two real-world applications: motion segmentation and face clustering. For the baseline methods, we use the MATLAB codes provided by their authors.

Three evaluation metrics are used in our experiments: clustering error (CE), neighbor selection error (NSE) and run time (CT). CE is defined as

$$CE = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(c_i \neq \hat{c}_i), \tag{6}$$

where c_i is the class label, and $\hat{c}_i$ is the estimated class label. Since the definite label index of estimated class may be not consistent with its real index, every permutation of the estimated class label should be calculated in CE, and the minimum among all the permutation is adopted.

$$NSE = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(j|W_{ij} \neq 0, c_i \neq \hat{c}_i). \tag{7}$$

NSE is the proportion of points that do not have all correct neighbors, it measures the extent that algorithms misconnect data points from different subspaces in the adjacency matrix. Besides, we compare the average run time (RT) to evaluate our algorithm's efficiency.

3.1 Motion Segmentation

To verify the performance of T-GSC in motion segmentation problem, we evaluate our method on the Hopkins155 motion segmentation database [26],[1] which comprises 155 video sequences of 2 or 3 motions, and the goal of this test is to segment the tracked points in a frame into different motion clusters. All the experiments in this paper are directly done on the raw data downloaded from the database website without any preprocessing.

Choosing the Subspace Dimension *P*. In theory, the value of subspace dimension P should be set equal to the underlying subspace where the data lies in. However, considering the existence of corruption and noise, choosing the subspace dimension directly equal to its theoretical value is suboptimal. According to the affine projection model, all the trajectories associated with a single rigid motion live in an affine subspace of dimension 3, so we choose the subspace dimension P around 3. The average CE and NSE of T-GSC on Hopkins155 under different P are listed in Table 1: the minimum CE is obtained when $P = 5$. Thus, we set the maximum subspace dimension $P = 5$ in the following motion segmentation experiment.

Table 1. Results on Hopkins 155 under varied P

P	3	4	5	6	7
Mean CE(%)	8.57	5.61	**3.94**	4.73	3.81
Mean NSE(%)	4.71	4.46	3.74	3.54	**3.42**

Motion Segmentation Performance. Table 2 shows the results over all sequences in Hopkins155. Since there is no much corruption or missing data in the Hopkins155 dataset, most baseline algorithms can achieve a really good performance. We can see that T-GSC performs comparable to the state-of-the-art methods, while keeps a relatively low computational time. Moreover, the CE and NSE of T-GSC are much smaller than NSN, which demonstrates the effectiveness of the first step in our method.

[1] The Hopkins155 database is available online at http://www.vision.jhu.edu/data/hopkins155/.

Table 2. Results on Hopkins 155 dataset

	Algotithms	SSC	LRR	SCC	OMP	TSC	NSN	T-GSC(Ours)
2 Motions	Mean CE(%)	1.53	**2.13**	2.24	17.25	18.44	3.62	3.07
	Median CE(%)	**0**	**0**	**0**	13.33	16.92	**0**	**0**
	Mean NSE(%)	**1.09**	6.03	–	37.61	2.86	2.91	2.64
	Mean RT(s)	0.50	0.96	0.37	0.17	0.16	**0.06**	0.24
3 Motions	Mean CE(%)	4.40	**4.03**	4.32	27.61	28.58	8.28	6.87
	Median CE(%)	0.56	1.43	**0.21**	23.79	29.67	2.76	1.49
	Mean NSE(%)	**2.44**	10.56	–	78.03	7.42	8.30	7.51
	Mean RT(s)	1.03	1.33	0.68	0.28	0.38	**0.12**	0.38
All	Mean CE(%)	**2.18**	2.56	2.71	19.59	20.73	4.67	3.94
	Median CE(%)	0.13	0.32	**0.05**	15.69	19.80	0.62	0.34
	Mean NSE(%)	**1.39**	7.05	–	46.74	3.89	4.13	3.74
	Mean RT(s)	0.62	1.04	0.44	0.19	0.21	**0.07**	0.27

3.2 Face Clustering

In this section, we evaluate the face clustering performance of T-GSC as well as many state-of-the-art methods on the Extended Yale-B database [27].[2] The Extended Yale-B dataset contains frontal face images of 38 subjects under 64 different illumination conditions.

To reduce the computational time and memory cost, we use the cropped images of the Extended Yale-B dataset and resize them to 4842 pixels, then concatenate the raw pixels value into a 2016-dimensional vector for each data point. We take a series experiments under different number of subjects. All of the experimental results of face clustering are adopted average value under 100 random trials.

Choosing the Subspace Dimension *P*. Similar to the motion segmentation experiment, we test different P of 9 to 20. The solid line in Fig. 1 shows the influence of P to T-GSC on Extended Yale-B: when the subspace dimension ranges from 9 to 14, the clustering error drops monotonically; while the clustering error keeps almost unchanged when P is increased to 15. This phenomenon implies that T-GSC is relatively robust to the choice of subspace dimension. Results of different P to NSN can be also seen in Fig. 1. Comparing T-GSC with NSN,we can see that T-GSC performs much better in low subspace dimension, and this demonstrates that T-GSC has more powerful ability to recover the subspace. Besides, with almost the same clustering error, T-GSC needs fewer neighbors than NSN, which helps to reduce the computational time.

[2] The Extended Yale-B database is available online at http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html.

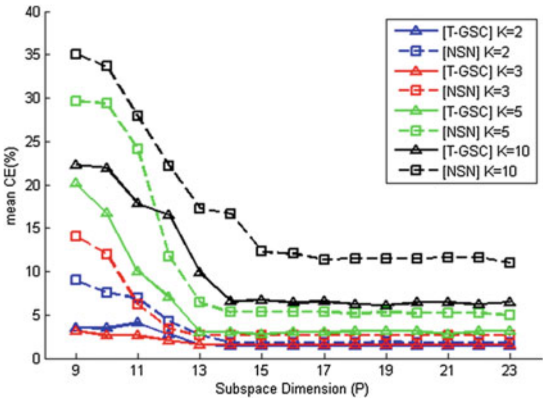

Fig. 1. Influence of P to T-GSC and NSN

Table 3. Results on Yale-B dataset

	Algotithms	SSC	LRR	OMP	TSC	NSN	T-GSC(Ours)
2 Subjects	Mean CE(%)	2.67	4.29	7.41	12.53	1.84	**1.45**
	Median CE(%)	**0**	0.78	1.57	2.36	0.78	0.78
	Mean NSE(%)	8.14	5.59	13.86	10.04	2.63	**1.90**
	Mean RT(s)	17.13	2.59	0.21	**0.15**	0.28	0.50
3 Subjects	Mean CE(%)	4.16	5.60	5.12	20.02	3.32	**2.24**
	Median CE(%)	**1.04**	**1.04**	2.08	13.54	2.6	1.56
	Mean NSE(%)	20.99	9.95	39.07	19.04	4.72	**3.07**
	Mean RT(s)	21.99	4.89	**0.35**	0.36	0.95	1.12
5 Subjects	Mean CE(%)	4.72	5.72	9.26	29.58	6.18	**3.40**
	Median CE(%)	**2.81**	2.90	5.00	31.25	5.31	2.97
	Mean NSE(%)	53.76	17.53	87.00	27.56	7.65	**4.66**
	Mean RT(s)	32.14	10.48	0.76	**0.49**	1.86	2.41
10 Subjects	Mean CE(%)	11.75	11.65	16.15	41.68	13.63	**8.20**
	Median CE(%)	11.02	12.73	17.19	42.66	12.03	**6.41**
	Mean NSE(%)	76.46	33.47	94.54	36.56	12.44	**7.89**
	Mean RT(s)	61.52	41.16	3.24	**1.56**	7.91	9.02

Taking the clustering error and computation time into account, we set $P = 16$ in the following face clustering experiment.

Face Clustering Performance. Table 3 demonstrates the face clustering results of different numbers of subspaces. T-GSC obtains the smallest mean CE as well as mean NSE in most cases, while SSC and LRR perform best on the median CE. One possible reason is that SSC and LRR, which are belong to optimization-based methods, are more suitable to handle general condition but not complex situation, while T-GSC is more robust. Thus the optimization-based methods fail in the small number of difficult cases, resulting in higher mean clustering errors than T-GSC. Besides, the proposed method is about three times faster than SSC, while holds comparable clustering error.

4 Conclusion

This paper studied the initialization problem of greedy subspace clustering methods, and proposed a two-step greedy subspace clustering method to alleviate this problem. First, for a data point, T-GSC constructs an initial subspace by seeking a sparse representation with respect to its nearest neighbors. Second, the subspace is updated by adding an orthogonal basis involved with the newly added data points. A series of experiments of motion segmentation on the Hopkins155 dataset, and face clustering on the Extended Yale-B dataset have been conducted. Experimental results show that T-GSC achieves better performance than other greedy subspace clustering methods, and meanwhile maintains comparable low computational cost.

References

1. Yang, A., Wright, J., Ma, Y., Sastry, S.: Unsupervised segmentation of natural images via lossy data compression. Computer Vis. Image Underst. (CVIU) **110**(2), 212–225 (2008)
2. Vidal, R., Tron, R., Hartley, R.: Multiframe motion segmentation with missing data using powerfactorization and GPCA. Int. J. Comput. Vis. (IJCV) **79**(1), 85–105 (2008)
3. Ho, J., Yang, M.H., Lim, J., Lee, K.C., Kriegman, D.: Clustering appearances of objects under varying illumination conditions. In: Computer Vision and Pattern Recognition (CVPR) (2003)
4. Hong, W., Wright, J., Huang, K., Ma, Y.: Multi-scale hybrid linear models for lossy image representation. IEEE Trans. Image Process. (TIP) **15**(12), 3655–3671 (2006)
5. Chaudhuri, K., Kakade, S.M., Livescu, K., Sridharan, K.: Multi-view clustering via canonical correlation analysis. In: International Conference on Machine Learning (ICML) (2009)
6. Zhao, X., Evans, N., Dugelay, J.L.: A subspace co-training framework for multi-view clustering. Pattern Recogn. Lett. (PRL) **41**, 73–82 (2014)
7. Vidal, R.: Subspace clustering. Sign. Process. Mag. **28**(2), 52–68 (2011)
8. Costeira, J., Kanade, T.: A multibody factorization method for independently moving objects. Int. J. Comput. Vis. (IJCV) **29**(3), 159–179 (1998)
9. Kanatani, K.: Motion segmentation by subspace separation and model selection. In: International Conference on Computer Vision (ICCV) (2001)
10. Vidal, R., Ma, Y., Sastry, S.: Generalized principal component analysis (GPCA). IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **27**(12), 1945–1959 (2005)
11. Zhang, T., Szlam, A., Lerman, G.: Median K-flats for hybrid linear modeling with many outliers. In: Proceedings of 2nd IEEE International Workshop on Subspace Methods, pp. 234–241 (2009)

12. Tipping, M., Bishop, C.: Mixtures of probabilistic principal component analyzers. Neural Comput. **11**(2), 443–482 (1999)
13. Yang, A.Y., Rao, S., Ma, Y.: Robust statistical estimation and segmentation of multiple subspaces. In: Computer Vision and Pattern Recognition Workshop (CVPRW) (2006)
14. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: Computer Vision and Pattern Recognition (CVPR) (2009)
15. Elhamifar, E., Vidal, R.: Sparse subspace clustering: algorithm, theory, and applications. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **35**(11), 2765–2781 (2013)
16. Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y.: Robust recovery of subspace structures by low-rank representation. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **35**(1), 171–184 (2013)
17. Zhang, Y., Sun, Z., He, R., Tan, T.: Robust subspace clustering via half-quadratic minimization. In: International Conference on Computer Vision (ICCV) (2013)
18. Zhang, Y., Sun, Z., He, R., Tan, T.: Robust low-rank representation via correntropy. In: Asian Conference on Pattern Recognition (ACPR) (2013)
19. Wang, Y., Xu, H., Leng, C.: Provable subspace clustering: when LRR meets SSC. In: Advances in Neural Information Processing Systems (NIPS) (2013)
20. Chen, G., Lerman, G.: Spectral curvature clustering. Int. J. Comput. Vis. (IJCV) **81**(3), 317–330 (2009)
21. Dyer, E.L., Sankaranarayanan, A.C., Baraniuk, R.G.: Greedy feature selection for subspace clustering. J. Mach. Learn. Res. (JMLR) **14**(1), 2487–2517 (2013)
22. Heckel, R., Bölcskei, H.: Subspace clustering via thresholding and spectral clustering. In: Acoustics, Speech, and Signal Processing (ICASSP) (2013)
23. Park, D., Caramanis, C.: Greedy subspace clustering. In: Advances in Neural Information Processing Systems (NIPS) (2014)
24. Donoho, D.L.: For most large underdetermined systems of linear equations the minimal $l1$-norm solution is also the sparsest solution. Commun. Pure Appl. Aathematics (CPAA) **59**(6), 797–829 (2006)
25. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Series B (Methodological) **58**, 267–288 (1996)
26. Tron, R., Vidal, R.: A benchmark for the comparison of 3-D motion segmentation algorithms. In: Computer Vision and Pattern Recognition (CVPR) (2007)
27. Lee, K.C., Ho, J., Kriegman, D.: Acquiring linear subspaces for face recognition under variable lighting. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **27**(5), 684–698 (2005)