Int J Comput Vis (2012) 100:217–240
DOI 10.1007/s11263-012-0535-6

Hybrid Linear Modeling via Local Best-Fit Flats

Teng Zhang · Arthur Szlam · Yi Wang · Gilad Lerman

Received: 14 October 2010 / Accepted: 11 May 2012 / Published online: 22 June 2012

Abstract We present a simple and fast geometric method for modeling data by a union of affine subspaces. The method begins by forming a collection of local best-fit affine subspaces, i.e., subspaces approximating the data in local neighborhoods. The correct sizes of the local neighborhoods are determined automatically by the Jones' β_2 numbers (we prove under certain geometric conditions that our method finds the optimal local neighborhoods). The collection of subspaces is further processed by a greedy selection procedure or a spectral method to generate the final model. We discuss applications to tracking-based motion segmentation and clustering of faces under different illuminating conditions. We give extensive experimental evidence demonstrating the state of the art accuracy and speed of the suggested algorithms on these problems and also on synthetic hybrid linear data as well as the MNIST handwritten digits data; and we demonstrate how to use our algorithms for fast determination of the number of affine subspaces.

Supp. webpage: http://www.math.umn.edu/~lerman/lbf/.

T. Zhang · Y. Wang · G. Lerman (✉)
School of Mathematics, University of Minnesota, Minneapolis, MN, USA
e-mail: lerman@umn.edu

T. Zhang
e-mail: zhang620@umn.edu

Y. Wang
e-mail: wangx857@umn.edu

A. Szlam
Courant Institute of Mathematical Sciences, New York University, New York City, NY, USA
e-mail: aszlam@courant.nyu.edu

Keywords Hybrid linear modeling · Subspace clustering · Spectral clustering · Local PCA · High-dimensional data · Motion segmentation · Face clustering

1 Introduction

Several problems from computer vision, such as motion segmentation and face clustering, give rise to modeling data by multiple subspaces. This is referred to as Hybrid Linear Modeling (HLM) or alternatively as "subspace clustering". In tracking-based motion segmentation, extracted feature points (tracked in all frames) are clustered according to the different moving objects. Under the affine camera model, the vectors of coordinates of feature points corresponding to a moving rigid object lie on an affine subspace of dimension at most 3 (see Costeira and Kanade 1998). Thus clustering different moving objects is equivalent to clustering different affine subspaces. Similarly, in face clustering, it has been proved that the set of all images of a Lambertian object under a variety of lighting conditions form a convex polyhedral cone in the image space, and this cone can be accurately approximated by a low-dimensional linear subspace (of dimension at most 9) (Epstein et al. 1995; Ho et al. 2003; Basri and Jacobs 2003). One may thus cluster certain images of faces by HLM algorithms.

The mathematical formulation of HLM assumes a data set $\mathrm{X} = \{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^D$ where each $\mathbf{x}_i$ lies on (or around) one of K flats (i.e., affine subspaces) and requires to find the partition of X corresponding to the flats. We would like to be able to do this when the data has been corrupted by additive noise and outliers;[1] in this case we may also want to determine the flats themselves. We first assume here that all flats

[1]Throughout the paper outliers are corrupted data points, i.e., points generated by a distribution, which assigns sufficiently small probabil-

have the same known dimension d (i.e., they are d-flats) and that their number K is known. In Sect. 3 we address to some extent the cases of unknown K and mixed dimensions.

Several algorithms have been suggested for solving the HLM problem (or even the more general problem of clustering manifolds), for example the K-flats (KF) algorithm or any of its variants (Tipping and Bishop 1999; Bradley and Mangasarian 2000; Tseng 2000; Ho et al. 2003; Zhang et al. 2009), methods based on direct matrix factorization (Boult and Brown 1991; Costeira and Kanade 1998; Kanatani 2001; Kanatani 2002), Generalized Principal Component Analysis (GPCA) (Vidal et al. 2005), Local Subspace Affinity (LSA) (Yan and Pollefeys 2006), RANSAC (for HLM) (Yang et al. 2006), Locally Linear Manifold Clustering (LLMC) (Goh and Vidal 2007), Agglomerative Lossy Compression (ALC) (Ma et al. 2007), Spectral Curvature Clustering (SCC) (Chen and Lerman 2009) and Sparse Subspace Clustering (SSC) (Elhamifar and Vidal 2009). Some theoretical guarantees for particular HLM algorithms appear in Chen and Lerman (2009), Arias-Castro et al. (2011), Lerman and Zhang (2011) and Soltanolkotabi and Candès (2011). We recommend a recent review on HLM by Vidal (2011).

Many of the algorithms described above require an initial guess of the subspaces. For example, the K-flats algorithm is an iterative method that requires an initialization, and in SCC, one needs to carefully choose collections of $d+1$ data points that lie close to each of the underlying d-flats. Other algorithms require some information about the suspected deviations from the hybrid linear model; for example both RANSAC (for HLM) and ALC ask for a model parameter corresponding to the level of noise.

Here we propose a straightforward geometric method for the estimation of local subspaces, which is inspired by Jones (1990), David and Semmes (1991) and Lerman (2003) as well as Fukunaga and Olsen (1971) and Little et al. (2009a, 2009b). These local subspace estimates can be used to set the model parameters for or initialize an HLM algorithm. The basic idea is that for a data set X sampled from a hybrid linear model (perhaps with some noise), there are many points $\mathbf{x}$ such that the principal components of an appropriately sized neighborhood of $\mathbf{x}$ give a good approximation to the subspace $\mathbf{x}$ belongs to. Using local subspaces to infer the global hybrid linear model was suggested in Yan and Pollefeys (2006) for linear subspaces; however, there they use very small neighborhoods that are not adaptive to the structure of the data (e.g., amount of noise etc.). An "appropriately sized neighborhood" needs to be larger than the noise, so that the subspace is recognized. However, the neighborhood cannot be so large that it contains points from multiple subspaces. The correct choice of this size is carefully quantified in Sect. 2.1. We refer to such "appropriately sized neighborhoods" as "optimal neighborhoods".

In addition to studying how to estimate local subspaces, we describe two complete HLM algorithms which are natural extensions of the local estimation: LBF (Local Best-fit Flat) and SLBF (spectral LBF). On many data sets, the first obtains state of the art speed with nearly state of the art accuracy (it can also deal with very large data), and the second obtains state of the art accuracy (SLBF) with reasonable run times (it seems to be able to deal to some extent with some nonlinear structures as the ones arising in motion segmentation data). We remark that we test accuracy in various scenarios, but in particular, with intersecting subspaces and with outliers. While in this work we only theoretically justify our choice of "optimal" neighborhoods, we are hopeful about developing a more complete theory justifying our algorithms.

In particular, we believe that such a theory can be valid in the setting suggested by Soltanolkotabi and Candès (2011) for analyzing the SSC algorithm, while having additional noise and restricting the fraction of outliers (or modifying our algorithms so they are even more robust to outliers). We are also interested in rigorously quantifying the limitations of our algorithms (as conjectured in Sect. 4).

We summarize the main contributions of this work, which is the full length version of Zhang et al. (2010), as follows.

- We make precise the local best-fit heuristic, using the β_2 numbers (Jones 1990; David and Semmes 1991; Lerman 2003). We give an algorithm to approximately find optimal neighborhoods in the above sense, in fact, we prove this under certain geometric conditions.
- Using the local best-fit heuristic, we introduce the LBF and SLBF algorithms for HLM. At each point of a randomly chosen subset of the data, they use the best-fit flats of the "optimal" neighborhoods to build a global model with different methods (LBF is based on energy minimization and SLBF is a spectral method).
- We perform extensive experiments on motion segmentation data (the Hopkins 155 benchmark of Tron and Vidal (2007)), face clustering (the extended Yale face database B), handwritten digits (the MNIST database), and artificial data, showing that both algorithms, in particular SLBF, are accurate on real and synthetic HLM problems, while LBF runs extremely fast (often on the order of ten times faster than most of the previously mentioned methods). For the cropped face data we actually indicate a fundamental problem of local methods like LBF and SLBF, though suggest a workaround that works for this particular data.
- We demonstrate how the local best-fit heuristic can be used with other algorithms. In particular, we give experimental evidence to show that the K-flats algorithm (Ho

ity for small neighborhoods around the underlying subspaces. This is different than corrupting selected entries of data points.

et al. 2003) is improved by initialization that is based on the local best-fit heuristic. We also use this heuristic to estimate the main parameters of both RANSAC (for HLM) (Yang et al. 2006) and ALC (Ma et al. 2007).
- We show how the combination of LBF and the elbow method can quickly determine the number of subspaces.

The rest of this paper is organized as follows. In Sect. 2 we describe the LBF and SLBF algorithms and state a theorem giving conditions that guarantee that good neighborhoods can be found. Section 3 carefully tests the LBF and SLBF algorithms (while comparing them to other common HLM algorithms) on both artificial data of synthetic hybrid linear models and real data of motion segmentation in video sequences, face clustering and handwritten digits recognition. It also demonstrates how to determine the number of clusters by applying the fast algorithm of this paper together with the straightforward elbow method. Section 4 concludes with a brief discussion and mentions possibilities for future work.

2 The Local Best-Fit Flats Heuristic and the LBF and SLBF Algorithms

We describe two methods, LBF and SLBF, which have at their heart an estimation of local flats capturing the global structures of the data (or part of it). Both methods first find a set of candidate flats (the number is an input parameter for LBF). These are best-fit flats for local "optimal" neighborhoods (we describe an algorithm for approximately finding such neighborhoods and justify it in Sect. 2.1). The two algorithms process the candidates in different ways: LBF uses energy minimization and SLBF uses a spectral approach.

2.1 Choosing the Optimal Neighborhood

We choose the candidate flats that capture the global structure of the data by fitting them to 'optimal' local neighborhoods of data points. For a point $\mathbf{x} \in \mathbb{R}^D$, we define an optimal neighborhood as the largest ball $B(\mathbf{x}, r)$ (centered at $\mathbf{x}$ and with radius r) that only contains points sampled from the same cluster as $\mathbf{x}$. Indeed, neighborhoods smaller than the optimal one (around $\mathbf{x}$) can mainly contain the noise around an underlying subspace (of the hybrid linear model); consequently their local best-fit flats may not match the underlying flat. On the other hand larger neighborhood than the optimal one (around $\mathbf{x}$) will contain points from more than one underlying flat, and the resulting best-fit flat will again not match any of the underlying flats. We note that the choice of neighborhood $B(\mathbf{x}, r)$ is equivalent to the choice of radius r, which we refer to as scale (even though it is also common to refer to $\log(r)$ or $-\log(r)$ as scale). While it is possible to take a guess at the optimal scale as a parameter (e.g., as commonly done by fixing the number of nearest neighbors to $\mathbf{x}$), we have found that it is possible to choose the optimal scale reasonably well automatically, while adapting it to the given point $\mathbf{x}$.

We will start at the smallest scale (i.e., smallest radius containing only $d+1$ points) and look at larger and larger neighborhoods of a given point $\mathbf{x}_0$. At the smallest scale, any noise may cause the local neighborhood to have higher dimension than d. As we add points to the neighborhood, it becomes better and better approximated in a scale-invariant sense (e.g., by scaling the neighborhood to have radius 1 and computing the error of approximation by best-fit flat then) until points belonging to other flats enter the neighborhood. To be more precise, we define the scale-invariant error for a neighborhood $\mathcal{N} \equiv B(\mathbf{x}_0, r)$ of $\mathbf{x}_0$ by the formula:

$$\beta_2(\mathcal{N}) = \frac{\min_{d\text{-flats } L} \sqrt{\sum_{\mathbf{y}\in\mathcal{N}} \|\mathbf{y} - P_L \mathbf{y}\|^2 / |\mathcal{N}|}}{\max_{\mathbf{x}\in\mathcal{N}} \|\mathbf{x} - \mathbf{x}_0\|}, \qquad (1)$$

where $|\mathcal{N}|$ denotes the number of points in $\mathcal{N}$, P_L denotes the projection onto the flat L and the minimization is over all d-flats in $\mathbb{R}^D$. We note that the numerator is the approximation error by best-fit ℓ_2 flat at scale r and the denominator is the scale r. The notion of scale-invariant error was introduced and utilized in Jones (1990), David and Semmes (1991) and Lerman (2003).

Using this scale-invariant error we can reformulate our criterion for choosing the optimal neighborhood more precisely. That is, we start with the smallest neighborhood containing S nearest neighbors of $\mathbf{x}$, increase the number of nearest neighbors by T in each iteration and check the β_2 number of each neighborhood using (1). We estimate the optimal neighborhood as the last one for which β_2 is smaller than β_2 of the previous neighborhood (that is, we search for the first local minimizer of $\beta_2(\mathcal{N})$). This procedure is summarized in Algorithm 1. It is experimentally robust to outliers if $\mathbf{x}$ is an inlier, since the nearest neighbors of inliers also tend to be inliers.

Algorithm 1 Neighborhood size selection for HLM by randomized local best-fit flats

```
Input: $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$: data, $\mathbf{x}$: a point in X, $S$:
  start size, $T$: step size
Output: $\mathcal{N}(\mathbf{x})$: a neighborhood of $\mathbf{x}$.
  Steps:
  • $k = -1$
  repeat
    • $k := k + 1$
    • Let $\mathcal{N}_k$ be the set of the $S + kT$ nearest points in X
      to $\mathbf{x}$
    • Compute $\beta_2(k) := \beta_2(\mathcal{N}_k)$ according to (1)
  until $k > 1$ and $\beta_2(k-1) < \min\{\beta_2(k-2), \beta_2(k)\}$
  • Output $\mathcal{N}(\mathbf{x}) := \mathcal{N}_{k-1}$
```

2.1.1 Theoretical Justification

The following theorem tries to justify our strategy of estimating the optimal scale around each point by showing that in the continuous setting the first local minimizer of $\beta_2(\mathbf{x}, r) := \beta_2(B(\mathbf{x}, r))$ is approximately the distance from $\mathbf{x}$ to the nearest cluster that does not contain $\mathbf{x}$ (here the underlying model is a mixture of Lebesgue measures in strips around several subspaces and $\mathbf{x}$ is an arbitrary point on one of these subspaces). Therefore, if we choose the size of neighborhood following Algorithm 1 (adapted to the continuous setting), then we will approximately obtain the optimal neighborhood. It is rather standard to extend such estimates for measures to a probabilistic setting, where i.i.d. data is sampled from the continuous distribution. The theorem will then hold with high probability for sufficiently large sample size (due to technicalities, which also require truncating the support of our continuous measure we avoid these details). The proof of this theorem is in Appendix A.

Our theorem uses the following analog of the discrete β_2 of (1) for a measure μ and a ball $B(\mathbf{x}, r)$ (see also Lerman 2003):

$$\beta_2(\mathbf{x}, r) = \frac{1}{r} \min_{d\text{-flats} L \subseteq \mathbb{R}^D} \sqrt{\int_{B(\mathbf{x},r)} \operatorname{dist}(\mathbf{x}, L)^2 \, d\mu / \mu(B(\mathbf{x}, r))}, \qquad (2)$$

where throughout the paper $\operatorname{dist}(\cdot,\cdot)$ denotes the Euclidean distance, for example in this case $\operatorname{dist}(\mathbf{x}, L) := \|\mathbf{x} - P_L\mathbf{x}\|$. The theorem also assumes that the underlying measure is supported on a union of K tubes $T(L_i, w_i) := \{\mathbf{x} \in \mathbb{R}^D : \|\mathbf{x} - P_{L_i}\mathbf{x}\| < w_i\}, i = 1, \ldots, K$ (centered around the d-flats $L_1, \ldots, L_K$ respectively).

Theorem 1 *Let $K \geq 2$, $d < D$, $\{L_i\}_{i=1}^K$ be K d-flats in $\mathbb{R}^D$, $\{\mu_i\}_{i=1}^K$ be K Lebesgue measures on tubes $\{T(L_i, w)\}_{i=1}^K \subset \mathbb{R}^D$ respectively and let $\mu = \sum_{i=1}^K \mu_i$. For fixed $1 \leq i^* \leq K$ and fixed $\mathbf{x}^* \in L_{i^*}$, let*

$$r_0 = r_0(\mathbf{x}^*) = \operatorname{dist}\Bigl(\mathbf{x}^*, \bigcup_{i \neq i^*} T(L_i, w)\Bigr). \qquad (3)$$

If $w < r_0$, then $\beta_2(\mathbf{x}^*, r)$ (as a function of r) is constant on $[0, w]$ and decreases on $[w, r_0]$. If also

$$\frac{w}{r_0} < \begin{cases} \min\Bigl(0.02, \sqrt{\frac{(D+1)}{150\sqrt{2}(D-1)K}}\Bigr), & \text{when } d = 1; \\ \min\Bigl(0.02, \sqrt{\frac{(D-d+2)}{6(50)^{\frac{d}{2}}(D-d)K}}\Bigr), & \text{when } d > 1, \end{cases} \qquad (4)$$

then there exists $r_0 < r^* < 1.09\, r_0$ such that

$$\beta_2\bigl(\mathbf{x}^*, r^*\bigr) > \beta_2\bigl(\mathbf{x}^*, r_0\bigr). \qquad (5)$$

That is, the first local minimum of $\beta_2(\mathbf{x}^*, r)$ (as a function of r) occurs in $(r_0, 1.09\, r_0)$.

The proof of this theorem indicates a weaker condition than (4), which is less intuitive. It also shows that $r^* \to r_0$ as $w/r_0 \to 0$ and clarifies by example why the first local minimum of $\beta_2(\mathbf{x}^*, r_0)$ is often bigger than r_0 (see Remark 1).

2.1.2 The Complexity of Algorithm 1

Algorithm 1 requires sorting the neighbors of $\mathbf{x}$ according to their distance to $\mathbf{x}$; the computational cost of this preprocessing step is $O(D \cdot N + N \cdot \log N)$. In order to obtain $\beta_2(\mathcal{N}_k)$, we need to obtain the top d singular values of the $|\mathcal{N}_k| \times D$ data matrix representing the $|\mathcal{N}_k|$ points, which requires a complexity of $O(d \cdot D \cdot |\mathcal{N}_k|)$. To find $\mathcal{N}(x)$, we need to generate $\beta_2(\mathcal{N}_k)$ for any $|\mathcal{N}_k| = S + kT$, where $k = 1, 2, \ldots, (N - S)/T$, hence the complexity for obtaining $\mathcal{N}(x)$ is of order:

$$O\Biggl(d \cdot D \cdot \sum_{k=1}^{(N-S)/T} (S + kT)\Biggr) \leq O\bigl(d \cdot D \cdot N^2/2T\bigr).$$

We thus note that if T is in the order of N, e.g., $T = \max(N/300, 2)$, the total complexity of Algorithm 1 is $O((d \cdot D + \log N) \cdot N)$. Note that if we limit the number of scales that we search, then the two log terms above can be replaced by a constant.

2.2 The LBF Algorithm

The LBF algorithm searches for a good set of flats from the candidates (described above) in a greedy fashion. A measure G of goodness of K flats is chosen; here, it will be the average distance of points to their nearest flats, i.e.,

$$G = G_{\mathrm{X}}(\{L_1, \ldots, L_K\}) = \sum_{\mathbf{x} \in \mathrm{X}} \operatorname{dist}\Bigl(\mathbf{x}, \bigcup_{i=1}^K L_i\Bigr). \qquad (6)$$

After randomly initializing K flats from the list of candidates, p passes are made through the data points. In each of the passes, we replace a random current flat with the candidate that minimizers the value of G. We then move to the next pass, picking a random flat, etc. Algorithm 2 sketches this procedure (where the greedy minimization of G is described in step 2).

The simplest choice of G is the sum of the squared distances of each point in X to its nearest flat, i.e., having the power 2 in (6). However, in some scenarios the l_1 energy of (6) is more robust to outliers than the mean squared error (see Lerman and Zhang (2011, 2012) for theoretical support and Zhang et al. (2009) for experimental support). This method also allows using energy functions, which are hard

Algorithm 2 LBF: energy minimization over randomized local best-fit flats

```
Input: $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$: data, $d$: dimension of subspaces, $C$: number of candidate planes, $K$: number of output flats/clusters, $p$: number of passes, $S$ and $T$: parameters for local scale calculation.
Output: A partition of X into $K$ disjoint clusters $\{\mathrm{X}_i\}_{i=1}^K$, each approximated by a $d$-dimensional flat.
  Steps:
  1. Pick $C$ random points in X
  2. For each of the $C$ points find appropriate local scale using Algorithm 1
  3. Generate a set $\mathcal{L}$ containing $C$ candidate flats $L_1, \ldots, L_C$ from the best fit flats to the neighborhoods from the previous step
  4. Pick a random subset of $K$ flats $\hat{\mathcal{L}} \subset \mathcal{L}$
  5. for $j = 1$ to $p$ do
       • Pick a random flat $L^* \in \hat{\mathcal{L}}$, and find $\hat{L} = \operatorname{argmin}_{L \in \mathcal{L}} G_{\mathrm{X}}(\{\hat{\mathcal{L}} \setminus \{L^*\}\} \cup \{L\})$
       • Update $\hat{\mathcal{L}} = \{\hat{\mathcal{L}} \setminus \{L^*\}\} \cup \{\hat{L}\}$
  6. end for
  7. Partition X by sending points to nearest flats in $\hat{\mathcal{L}}$
```

to minimize (even heuristically). Indeed, it only requires evaluating the energy on the candidate configurations. For example, when the data set requires stronger robustness to outliers, one may use the following energy:

$$G' = G'_{\mathrm{X}}(\{L_1, \ldots, L_K\}) = \operatorname*{Median}_{\mathbf{x} \in \mathrm{X}} \operatorname{dist}\left(\mathbf{x}, \bigcup_{i=1}^{K} L_i\right).$$

The LBF algorithm is closely related to RANSAC, since both of them use candidate subspaces to fit the data set. However Algorithm 1 gives LBF an advantage in choosing good candidates, while RANSAC fits a d-flat by arbitrarily chosen $d+1$ points.

2.2.1 *The Complexity and Storage of Algorithm 2*

For step 2 of this algorithm we need to run Algorithm 1 C times and thus its complexity is of order $O((d \cdot D + \log N) \cdot C \cdot N)$. Note that the $\log N$ comes from a full sort of N distances, and if we restrict to a fixed number of scales, this can be replaced by a constant. Step 3 of Algorithm 2, requires C SVD decompositions for C matrices of size at most $N \times D$, in order to obtain the first d vectors in $\mathbb{R}^D$. It thus also has a complexity at most $O(C \cdot d \cdot D \cdot N)$.

Step 2 of Algorithm 2 requires the evaluation of the $N \times C$ matrix representing the distances $\|x_i - P_{L_j} x_i\|$ between $\mathrm{X} = \{x_1, x_2, \ldots, x_N\}$ and $L_1, L_1, \ldots, L_C$. This costs $O(C \cdot d \cdot D \cdot N)$ operations, since each distance from a point to a subspace costs $O(d \cdot D)$. Moreover, the p passes have complexity of order $O(p \cdot (C - K) \cdot N)$. Therefore, step 2 of Algorithm 2 has a complexity of order $O(C \cdot N \cdot (d \cdot D + p))$. At last, Step 7 of Algorithm 2 has a complexity of order $O(K \cdot d \cdot D \cdot N)$, which comes from the construction of the $N \times K$ matrix of distances from N points to K subspaces. Combining these complexities together, we have an overall complexity of $O(C \cdot N \cdot (d \cdot D + p + \log N))$ for the LBF Algorithm; as before, if we fix the number of scales independently from N, the log terms can be replaced by a constant.

To compute the storage requirements of LBF, we note that the data set is saved in an $N \times D$ matrix, the candidate subspaces are organized in C projection matrices of size $D \times d$ and in addition the algorithm stores an $N \times C$ matrix of distances between the data points and the C candidate subspaces. Therefore, the storage of LBF is in the order of $O(D \cdot N + C \cdot D \cdot d + N \cdot C)$.

2.3 The SLBF Algorithm

The SLBF algorithm (which is sketched in Algorithm 3) processes the candidate subspaces via a spectral clustering method. It first finds the neighborhoods $\{\mathcal{N}_i\}_{i=1}^N$ for all points $\{\mathbf{x}_i\}_{i=1}^N$ via Algorithm 1 and fits d-flats $\{L_i\}_{i=1}^N$ (via PCA) in these neighborhoods. It then forms the $N \times N$ matrices $\mathbf{S}$ and $\hat{\mathbf{S}}$ as follows:

$$\mathbf{S}_{i,j} = \sqrt{\operatorname{dist}(\mathbf{x}_i, L_j) \operatorname{dist}(\mathbf{x}_j, L_i)}, \tag{7}$$

and

$$\hat{\mathbf{S}}_{i,j} = \exp\left(-\mathbf{S}_{i,j}/2\sigma_j^2\right) + \exp\left(-\mathbf{S}_{i,j}/2\sigma_i^2\right), \tag{8}$$

where

$$\sigma_j = \lambda \sqrt{\min_{d\text{-flats L}} \sum_{\mathbf{x} \in \mathcal{N}_j} \|\mathbf{x} - P_L \mathbf{x}\|^2 / |\mathcal{N}_j|} \tag{9}$$

(we explain the choice of λ below, when we clarify (9)). Finally, it applies spectral clustering with the matrix $\hat{\mathbf{S}}$. More precisely, SLBF follows the main algorithm of Ng et al. (2001, Sect. 2), replacing the matrix A there by $\hat{\mathbf{S}}$, multiplying the unit eigenvectors of Step 3 (of Ng et al. (2001, Sect. 2)) by the corresponding square roots of eigenvalues and skipping step 4. We remark that the two last changes to Ng et al. (2001, Sect. 2) are commonly used so that the normalized similarity matrix (defined in step 4 of Algorithm 3) can be considered as a Gram matrix, see e.g., Euclidean MDS (Cox and Cox 2001) and ISOMAP (Tenenbaum 2000).

As discussed in Vidal (2011), SLBF is a "spectral clustering-based method", similar to SCC, LSA and SSC. These algorithms construct an $N \times N$ affinity matrix, whose ij-th entry represents the similarity between points i and j,

Algorithm 3 SLBF: spectral clustering based on best-fit flats

```
Input: X = {$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$} ⊆ $\mathbb{R}^D$: data, $\lambda$: a parameter
    (or several parameters if we use step 7, with default values $[2, 2e, 2e^2, \ldots, 2e^6]$), other parameters used by Algorithm 1.
Output: A partition of X into $K$ disjoint clusters $\{\mathrm{X}_i\}_{i=1}^K$,
    each approximated by a single flat.
    Steps:
    1. For each point $\mathbf{x}_i$, fit a subspace $L_i$ by Algorithm 1
    2. Construct the $N \times N$ matrix $\mathbf{S}$ and $\hat{\mathbf{S}}$ by (7), (8) and (9)
    3. Let $\mathbf{D}$ be the $N \times N$ diagonal matrix, such that $\mathbf{D}_{i,i} = \sum_{j=1}^N \hat{\mathbf{S}}_{i,j}$
    4. Normalize $\hat{\mathbf{S}}$ as follows: $\hat{\mathbf{S}} = \mathbf{D}^{-\frac{1}{2}} \hat{\mathbf{S}} \mathbf{D}^{-\frac{1}{2}}$
    5. Let $\mathbf{U}$ be the $N \times K$ matrix whose columns are the top $K$ eigenvectors of $\hat{\mathbf{S}}$, and $\boldsymbol{\Sigma}$ be the $K \times K$ matrix representing the top $K$ eigenvalues of $\hat{\mathbf{S}}$
    6. Apply $K$-means to the rows of $N \times K$ matrix $\mathbf{U}\boldsymbol{\Sigma}^{1/2}$ and partition X accordingly
    7. Repeat steps 2–6 with the default values of $\lambda$ (see input) to obtain several segmentations and choose the segmentation minimizing the error:
```

$$\sum_{i=1}^{K} \min_{d\text{-flat } L} \left(\sum_{\mathbf{x} \in \mathrm{X}_i} \mathrm{dist}^2(\mathbf{x}, L_i) \right) \tag{10}$$

and then apply spectral clustering using this affinity matrix. Ideally, the affinities of points from the same cluster are of order 1 and the affinities of points from different clusters are of order 0. Indeed, for the affinity $\hat{\mathbf{S}}$ of SLBF, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same cluster, then we expect that $\mathbf{x}_i$ is close to L_j and $\mathbf{x}_j$ is close to L_i, which means $\mathbf{S}_{i,j}$ is close to 0 and thus $\hat{\mathbf{S}}_{i,j}$ is close to 1 (we assume here that L_i and L_j are good estimators for the underlying subspace of the cluster shared by $\mathbf{x}_i$ and $\mathbf{x}_j$ as suggested by Theorem 1). Otherwise, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are not in the same cluster, then we expect that $\mathbf{x}_i$ is sufficiently far from L_j and $\mathbf{x}_j$ is sufficiently far from L_i, which implies that $\hat{\mathbf{S}}_{i,j}$ is close to 0. The choice of σ_j clearly affects this heuristic argument on the size of $\hat{\mathbf{S}}_{i,j}$. Theoretically σ_j should be larger than the noise, such that $\hat{\mathbf{S}}_{i,j}$ is close to 1 when $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same cluster, but σ_j cannot be too large so that $\hat{\mathbf{S}}_{i,j}$ is close to 1 when $\mathbf{x}_i$ and $\mathbf{x}_j$ are not in the same cluster. Therefore we use (9), where $\sqrt{\min_{d\text{-flats } L} \sum_{\mathbf{x} \in \mathcal{N}_j} \|\mathbf{x} - P_L \mathbf{x}\|^2 / |\mathcal{N}_j|}$ is the estimated noise of the data set around the point $\mathbf{x}_j$ and λ is a parameter. Following the strategy in Chen and Lerman (2009), we choose different values of λ (our fixed default values are $[2, 2e, 2e^2, \ldots, 2e^6]$) and consequently obtain several segmentation results (7 results when using our default values). We then choose the segmentation with the smallest error in (10).

We remark that the robustness of SLBF to outliers can be partly explained by the robustness of spectral-type methods to outliers. Furthermore, it is possible to initially remove some outliers according to very small values of the corresponding diagonal elements of $\mathbf{D}$ (see e.g., Chen and Lerman (2009), Arias-Castro et al. (2011)).

Similar to SLBF, LSA (Yan and Pollefeys 2006) is also based on fitting local subspaces. However, LSA fits subspace by local neighborhoods of fixed number of points and is not adaptive. Moreover, the local subspaces of LSA are forced to be linear (since the affinity of LSA is based on principal angles between such subspaces) and this further restricts the applicability of LSA. There is also some similarity between the idea of SLBF and that of SCC (Chen and Lerman 2009). Indeed, we may view SCC as fitting candidate subspaces based on $d+1$ data points (the iterative procedure tries to enforce the points to be from the same cluster). However, in practice they operate very differently, in particular, SCC is not based just on local information (though a local version of SCC follows from Arias-Castro et al. (2011)). The SSC algorithm is also a spectral method, but similar to SCC its affinities are global (they are based on sparse representation of data points).

2.3.1 *Complexity and Storage of the SLBF Algorithm*

Step 1 of Algorithm 3 has a complexity of order $O((d \cdot D + \log N) \cdot N^2)$, since it applies Algorithm 1 to every point in the set X. The most expensive calculation of steps 2–4 in Algorithm 3 is the construction of $\mathbf{S}$, which requires a complexity of order $O(d \cdot D \cdot N^2)$. The eigenvalue decomposition in step 5 has a complexity of order $O(K \cdot N^2)$ and the K-means algorithm in step 6 has a complexity of order $O(n_s \cdot N \cdot K^2)$, where n_s is the iterations in K-means.

Combining these complexities together, we have an overall complexity of order $O((d \cdot D + \log N) \cdot N^2 + n_s \cdot N \cdot K^2)$ for SLBF. As before, limiting to a constant number of scales replaces the log term with a constant.

We note that SLBF stores the data set in a $D \times N$ matrix, the candidate subspaces in N $D \times d$ matrices (recall that in SLBF every data point is assigned a subspace and thus $C = N$) and it also uses the $N \times N$ matrix $\mathbf{S}$. Therefore, the storage of SLBF is in the order of $O(N \cdot D \cdot d + N^2)$.

2.4 Adaptation of the Proposed Algorithms to Motion Segmentation Data

Note that the first minimum in the Theorem 1 excludes the left endpoint, and thus $k = 0$ is excluded in Algorithm 1. In our experiments, we noticed that on data without too much noise, it is useful to allow the first scale to count as a local minimum and allow $k = 0$ in Algorithm 1. We refer to the implementation of LBF and SLBF with those two techniques tailored for motion segmentation data as LBF-MS and SLBF-MS.

3 Experimental Results

In this section, we conduct experiments on artificial and real data sets to verify the effectiveness of the proposed algorithm in comparison to other HLM algorithms. We will see that in many situations, the methods we propose are fast and accurate; however, in Sect. 3.3 we will show a failure mode of our method, and discuss how this can be corrected.

We measure the accuracy of those algorithms by the rate of misclassified points with outliers excluded, that is

$$\text{error\%} = \frac{\text{\# of misclassified inliers}}{\text{\# of total inliers}} \times 100\,\%. \tag{11}$$

In all the experiments below, the number C in Algorithm 2 is $70 \cdot K$, where K is the number of subspaces, the number p in Algorithm 2 is $5 \cdot K$, and the numbers S and T in Algorithm 1 are $2 \cdot d$ and 2 respectively, where d is the dimension of the subspace. According to our experience, LBF and SLBF are very robust to changes in parameters, but unsurprisingly, there is a general trade off between accuracy (higher C, higher p, smaller T), and run time (lower C, lower p, larger T). We have chosen these parameters for a balance between run time and accuracy. Nevertheless, we have insisted to use the same parameters for all data sets and experiments, even though particular parameters could obtain even better or near perfect results for some of the data sets. The experiments in Sects. 3.1 and 3.2 run on a computer with Intel Core 2 CPU at 2.66 GHz and 2 GB memory, and experiments in Sects. 3.3 and 3.4 run on a machine with Intel Core 2 Quad Q6600 at 2.4 GHz and 8 GB memory.

3.1 Clustering Results on Artificial Data

We compare our algorithms with the following algorithms: Mixtures of PPCA (MoPPCA) (Tipping and Bishop 1999), K-flats (KF) (Ho et al. 2003), Local Subspace Analysis (LSA) (Yan and Pollefeys 2006), Spectral Curvature Clustering (SCC) (Chen and Lerman 2009), Random Sample Consensus (RANSAC) for HLM (Yang et al. 2006), Agglomerative Lossy Compression (ALC) (Ma et al. 2007) and GPCA with voting/robust GPCA (GPCA) (Ma et al. 2008; Yang et al. 2006). Throughout the rest of the paper, we use the Matlab codes of the GPCA, MoPPCA and KF algorithms from http://perception.csl.uiuc.edu/gpca, the LSA algorithm from http://www.vision.jhu.edu/db, the SCC algorithm from http://www.math.umn.edu/~lerman/scc, the ALC algorithm from http://perception.csl.uiuc.edu/coding/motion/, the RANSAC algorithm from http://www.vision.jhu.edu/code/ and the SSC algorithm from http://www.cis.jhu.edu/~ehsan/ssc.htm.

For the SCC algorithm, we also try a slightly modified version tailored for motion segmentation as in step 6 of Algorithm 3, which we refer to as SCC-MS (SCC for motion segmentation): Following the notation of Chen and Lerman (2009, Algorithm 2) we let the matrix $\mathbf{U}$ be the $N \times K$ matrix whose columns are the top K left singular vectors of $\mathbf{A}_C^*$ and also denote by $\mathbf{\Sigma}$ the diagonal $K \times K$ matrix whose elements are the top K left singular values of $\mathbf{A}_C^*$. Then the K-means step of SCC-MS is applied directly to the rows of the $N \times K$ matrix $\mathbf{U}\mathbf{\Sigma}^{1/2}$ (as opposed to applying it to U (or its row-wise normalization by 1) in SCC).

The MoPPCA algorithm is always initialized with a random guess of the membership of the data points. The LSCC algorithm is initialized by randomly picking $100 \times K$ $(d+1)$-tuples (following Chen and Lerman 2009) and KF is initialized with a random guess. Since algorithms like KF tend to converge to local minimum, we use 10 restarts for MoPPCA, 30 restarts for KF, and recorded the misclassification rate of the one with the smallest ℓ_2 error for both of these algorithms. The number of restarts was restricted by the running time and accuracy. In SSC algorithm, we set the value λ to be 0.01, as suggested in the code.

The RANSAC for HLM and ALC algorithms (Yang et al. 2006; Ma et al. 2007) depend on a user supplied inlier threshold. RANSAC (oracle) and ALC (oracle) use the oracle inlier bound given by the true noise variance of the model and thus clearly have an advantage over the other algorithms listed. RANSAC (ϵ from LBF) and ALC (ϵ from LBF) estimate the inlier threshold by the local best-fit flats heuristic of this paper. That is, they fit best-fit neighborhoods for all N points using the latter heuristic and estimate the least error of approximation by d-flats in these N neighborhoods. The inlier bound ϵ is then the average of these errors. If the number of clusters resulting from ALC (ϵ from LBF or oracle) is larger than K, then we choose the K largest clusters and identify the points in the rest of clusters as outliers. For some cases the RANSAC algorithm breaks down and we then report it as N/A. The reason for this is that RANSAC (for HLM) (Yang et al. 2006) is very sensitive to the estimate of ε and an overestimate can result in removal of points belonging to more than one subspace, so that the algorithm may exhaust all points before detecting K subspaces.

We remark that GPCA cannot naturally deal with outliers, therefore we use robust GPCA with Multivariate Trimming (Yang et al. 2006) and the parameters ‘angleTolerance’

Table 1 Mean percentage of misclassified points in artificial data for linear-subspace cases or affine-subspace case

Linear		$2^2 \in \mathbb{R}^4$		$4^2 \in \mathbb{R}^6$		$2^4 \in \mathbb{R}^4$		$10^2 \in \mathbb{R}^{15}$		$(4,5,6) \in \mathbb{R}^{10}$		$(1,5) \in \mathbb{R}^6$	
Outl. %		0	30	0	30	0	30	0	30	0	30	0	30
LSCC	e %	2.6	6.9	**0.0**	2.6	**0.1**	22.4	0.5	3.8	1.8	28.2	N/A	34.6
	$t(s)$	1.1	0.8	1.0	1.8	1.5	2.0	13.3	5.7	5.1	8.4	N/A	1.9
LSCC-MS	e %	2.7	**10.0**	**0.0**	4.1	**0.1**	36.7	0.7	31.9	1.4	19.8	N/A	32.9
	$t(s)$	1.1	0.5	1.1	1.4	1.7	1.5	5.1	5.6	4.0	4.6	N/A	2.0
LSA	e %	18.4	19.6	0.1	12.7	0.4	21.0	0.1	9.9	5.9	6.6	27.4	35.4
	$t(s)$	6.8	16.0	7.1	20.8	23.8	54.4	11.7	31.5	20.1	54.4	6.6	13.8
KF	e %	**2.5**	15.8	2.5	18.4	**0.1**	34.3	**0.0**	33.8	**1.0**	30.6	20.2	23.5
	$t(s)$	0.5	0.6	0.2	0.8	0.7	1.8	0.4	1.0	**0.7**	2.8	0.3	0.5
MoPPCA	e %	**2.5**	14.2	**0.0**	17.7	**0.1**	34.2	**0.0**	38.8	1.6	34.7	23.4	24.0
	$t(s)$	0.3	0.5	0.2	0.7	0.7	2.0	**0.2**	1.1	1.1	3.3	0.5	0.5
GPCA	e %	6.0	**2.5**	**0.0**	**2.0**	**0.1**	**6.3**	**0.0**	14.6	14.6	N/A	5.9	N/A
	$t(s)$	2.1	38.0	1.9	85.2	10.8	136.2	11.2	546.0	73.8	N/A	0.7	N/A
LBF	e %	2.8	3.7	**0.0**	2.3	**0.1**	11.5	**0.0**	**1.9**	1.5	**1.5**	18.8	14.1
	$t(s)$	0.6	0.5	0.5	0.5	1.8	2.7	0.6	0.8	1.1	1.4	0.5	0.5
LBF-MS	e %	2.7	3.0	**0.0**	2.6	**0.1**	11.7	**0.0**	2.2	1.3	**1.5**	19.5	13.7
	$t(s)$	0.6	0.5	0.4	0.5	1.7	2.6	0.4	**0.6**	0.9	**1.3**	**0.4**	0.4
SLBF	e %	5.2	6.3	0.1	7.0	**0.1**	23.9	**0.0**	6.2	2.0	2.4	11.1	13.5
	$t(s)$	11.2	20.7	9.4	21.7	65.0	174.9	9.5	23.3	23.2	64.2	9.3	15.3
SLBF-MS	e %	7.8	11.7	0.1	6.6	0.2	46.6	**0.0**	4.8	1.9	2.6	19.7	22.1
	$t(s)$	12.0	24.0	8.8	24.4	68.1	202.0	8.4	23.5	22.0	72.4	9.8	16.3
RANSAC (oracle)	e %	2.7	2.6	2.9	2.1	8.0	9.4	0.5	5.8	1.7	1.5	N/A	31.6
	$t(s)$	**0.1**	**0.1**	**0.1**	**0.2**	**0.1**	**0.2**	5.9	6.7	1.5	7.1	N/A	**0.2**
RANSAC (ϵ from LBF)	e %	3.2	2.6	2.1	2.4	7.7	9.8	0.4	6.7	1.8	**1.5**	N/A	30.6
	$t(s)$	**0.1**	**0.1**	**0.1**	**0.2**	**0.1**	**0.2**	5.9	6.7	1.5	7.0	N/A	0.3
ALC (oracle)	e %	4.1	3.4	0.1	16.3	**0.1**	30.1	50.0	50.0	5.4	36.1	**0.3**	0.4
	$t(s)$	7.3	23.2	7.7	33.6	28.4	136.3	13.9	172.6	23.0	180.1	7.8	17.3
ALC (ϵ from LBF)	e %	4.5	5.7	0.1	10.0	**0.1**	14.0	50.0	50.0	2.5	1.8	0.4	**0.3**
	$t(s)$	8.0	28.0	8.1	37.9	29.6	121.9	16.6	152.4	24.0	151.6	8.3	18.1
SSC	e %	19.5	34.3	0.2	43.5	0.4	52.8	47.0	44.9	11.5	54.0	9.4	15.9
	$t(s)$	114.8	236.2	97.6	247.9	227.7	591.3	106.0	276.6	185.5	437.9	94.1	142.1
SCC	e %	**0.0**	0.6	**0.0**	**0.0**	**0.0**	0.5	**0.0**	0.7	**0.0**	5.8	N/A	N/A
	$t(s)$	1.2	1.0	1.1	2.0	1.4	2.5	6.1	13.7	5.6	6.0	N/A	N/A
SCC-MS	e %	**0.0**	2.2	**0.0**	0.5	**0.0**	5.8	**0.0**	**0.0**	**0.0**	3.1	N/A	N/A
	$t(s)$	1.2	0.7	1.2	1.6	1.7	2.2	5.4	6.0	4.6	4.8	N/A	N/A
LSA	e %	0.1	11.0	**0.0**	4.7	0.4	41.7	**0.0**	**0.0**	**0.0**	1.1	37.5	37.9
	$t(s)$	6.7	16.1	7.1	20.8	22.2	54.0	11.7	32.2	38.3	54.0	6.6	13.9
KF	e %	0.2	15.1	0.1	26.0	0.3	37.1	**0.0**	24.9	**0.0**	23.5	24.8	27.1
	$t(s)$	0.8	0.6	0.4	0.7	1.0	1.4	0.6	1.7	**1.0**	1.4	0.5	**0.5**
MoPPCA	e %	0.2	23.7	0.1	38.3	0.5	39.8	**0.0**	45.2	**0.0**	46.8	30.8	30.4
	$t(s)$	0.9	0.5	0.5	0.6	1.1	1.4	0.9	1.9	1.9	2.0	0.5	**0.5**
GPCA	e %	0.2	18.4	0.2	22.2	0.4	38.1	**0.0**	27.9	0.3	N/A	N/A	N/A
	$t(s)$	1.8	43.7	3.3	104.0	8.3	209.3	11.8	501.1	69.1	N/A	N/A	N/A
LBF	e %	**0.0**	2.0	**0.0**	0.7	**0.0**	4.5	**0.0**	0.3	**0.0**	**0.0**	4.7	11.2
	$t(s)$	0.7	0.6	0.5	0.6	1.9	2.8	0.6	0.8	1.2	1.5	**0.4**	0.5
LBF-MS	e %	**0.0**	2.7	**0.0**	1.5	**0.0**	5.2	**0.0**	0.5	**0.0**	**0.0**	3.9	10.5
	$t(s)$	0.6	0.5	0.4	**0.5**	1.7	2.7	**0.4**	**0.6**	**1.0**	**1.3**	**0.4**	**0.4**

Table 1 (*Mean percentage of misclassified points in artificial data for linear-subspace cases or affine-subspace case*)

Linear		$2^2 \in \mathbb{R}^4$		$4^2 \in \mathbb{R}^6$		$2^4 \in \mathbb{R}^4$		$10^2 \in \mathbb{R}^{15}$		$(4,5,6) \in \mathbb{R}^{10}$		$(1,5) \in \mathbb{R}^6$	
Outl. %		0	30	0	30	0	30	0	30	0	30	0	30
SLBF	e %	**0.0**	1.0	**0.0**	**0.0**	**0.0**	**0.1**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	$t(s)$	9.3	19.1	5.8	19.0	37.7	143.1	6.3	19.4	35.1	61.4	5.9	14.8
SLBF-MS	e %	**0.0**	0.1	**0.0**	**0.0**	**0.0**	**0.1**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	$t(s)$	8.8	21.7	5.6	21.9	38.0	175.5	5.9	21.1	40.1	66.7	5.9	14.3
RANSAC (oracle)	e %	13.8	11.6	9.8	9.6	30.9	27.0	1.9	8.3	1.2	3.4	N/A	23.6
	$t(s)$	**0.1**	**0.2**	**0.4**	1.8	**0.4**	**0.8**	6.4	6.8	3.7	7.4	N/A	**0.5**
RANSAC (ϵ from LBF)	e %	13.6	11.6	11.6	10.4	29.9	28.5	1.4	9.6	1.2	2.4	N/A	23.1
	$t(s)$	**0.1**	**0.2**	**0.4**	1.9	**0.4**	**0.8**	6.4	6.7	3.7	7.4	N/A	**0.5**
ALC (oracle)	e %	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	25.1	**0.0**	40.0	**0.0**	65.0	**0.0**	**0.0**
	$t(s)$	17.6	25.2	16.6	39.1	64.2	119.3	20.0	43.0	39.7	92.7	18.3	36.8
ALC (ϵ from LBF)	e %	**0.0**	0.4	**0.0**	**0.0**	**0.0**	0.3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	$t(s)$	18.7	26.8	17.2	29.8	65.2	113.6	24.4	55.5	47.9	85.2	18.8	38.9
SSC	e %	**0.0**	1.9	**0.0**	0.1	0.1	6.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	$t(s)$	135.9	226.8	176.0	134.7	283.8	592.4	187.0	311.9	338.6	504.1	127.1	183.9

and ‘boundarythreshold’ are set to be 0.3 and 0.4 respectively.

The artificial data represents various instances of K linear subspaces in $\mathbb{R}^D$. If their dimensions are fixed and equal d, we follow Chen and Lerman (2009) and refer to the setting as $d^K \in \mathbb{R}^D$. If they are mixed, then we follow Ma et al. (2008) and refer to the setting as $(d_1, \dots, d_K) \in \mathbb{R}^D$. Fixing K and d (or $d_1, \dots, d_K$), we randomly generate 100 different instances of corresponding hybrid linear models according to the code in http://perception.csl.uiuc.edu/gpca. More precisely, for each of the 100 experiments, K linear subspaces of the corresponding dimensions in $\mathbb{R}^D$ are randomly generated. The random variables sampled within each subspace are sums of two other variables. One of them is sampled from a uniform distribution in a d-dimensional ball of radius 1 of that subspace (centered at the origin for the case of linear subspaces). The other is sampled from a D-dimensional multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix $0.05^2 \cdot \mathbf{I}_{D\times D}$. Then, for each subspace 250 samples are generated according to the distribution just described. Next, the data is further corrupted with 5 % or 30 % uniformly distributed outliers in a cube of sidelength determined by the maximal distance of the former 250 samples to the origin (using the same code).

Since most algorithms (SCC, LSA, MoPPCA, LBF, SLBF, RANSAC, SSC) do not support mixed dimensions natively, we assume each subspace has the maximum dimension in the experiment. GPCA and ALC support mixed dimensions natively, and GPCA is the only algorithm for which we specify the dimensions for each subspace in mixed-dimension case (note that the knowledge of dimensions are unnecessary in ALC algorithm).

The mean (over 100 instances) misclassification rates and the mean running time of the various algorithms are recorded in Table 1. From Table 1 we can see that our algorithms, i.e., LBF, LBF-MS, SLBF, SLBF-MS, perform well in various artificial instances of hybrid linear modeling (with both linear subspaces and affine subspaces), and their advantage is especially obvious with many outliers and affine subspaces. The robustness to outliers is a result of our use of both ℓ_1 loss function (see e.g., Lerman and Zhang 2011, 2012) and random sampling. The SLBF and SLBF-MS are better at the affine cases because of their use of spectral clustering. Also unlike many other methods, the proposed methods natively support affine subspace models (the particular data has non-intersecting subspaces, which makes advantageous to some other algorithms, e.g., SSC). The results of RANSAC (ϵ from LBF) and ALC (ϵ from LBF) show that the local best-fit heuristic can be effectively used to estimate the main parameter of RANSAC and ALC, i.e., to estimate the local noise. Table 1 also shows that the running time of LBF/LBF-MS is less than the running time of most other algorithms, especially GPCA, LSA, RANSAC, ALC and SSC. The difference is large enough that we can also use the proposed algorithm as an initialization for the others. LBF and LBF-MS algorithms are slower than a single run of K-flats, but it usually takes many restarts of K-flats to get a decent result. Notice that the choice of C and p in our algorithm function in a similar manner to the number of restarts in KF. SLBF and SLBF-MS cost more time when N is large, because of the construction of the $N \times N$ matrix in spectral clustering, but it still has a comparable speed to LSA and is faster than SSC, which are two spectral-clustering based algorithms.

Table 2 The mean and median percentage of misclassified points for two-motions and three-motions in Hopkins 155 database

	Checker		Traffic		Articulated		All	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median
2-motion								
GPCA	6.09	1.03	1.41	**0.00**	2.88	**0.00**	4.59	0.38
LLMC 5	4.37	**0.00**	0.84	**0.00**	6.16	1.37	3.62	**0.00**
LSA $4K$	2.57	0.27	5.43	1.48	4.10	1.22	3.45	0.59
LBF$(4K, 3)$	3.65	**0.00**	3.89	**0.00**	4.43	0.15	3.78	**0.00**
LBF-MS$(4K, 3)$	2.90	**0.00**	1.64	**0.00**	2.51	0.06	2.54	**0.00**
SLBF$(2F, 3)$	1.59	**0.00**	**0.20**	**0.00**	**0.80**	**0.00**	1.16	**0.00**
SLBF-MS$(2F, 3)$	**1.28**	**0.00**	0.21	**0.00**	0.94	**0.00**	**0.98**	**0.00**
SCC$(4K, 3)$	2.42	**0.00**	4.44	**0.00**	9.51	2.04	3.60	**0.00**
SCC-MS$(4K, 3)$	2.00	**0.00**	0.35	**0.00**	4.11	1.12	1.77	**0.00**
SSC-N$(4K, 3)$	1.29	**0.00**	0.29	**0.00**	0.97	**0.00**	1.00	**0.00**
MSL	4.46	**0.00**	2.23	**0.00**	7.23	**0.00**	4.14	**0.00**
RANSAC	6.52	1.75	2.55	0.21	7.25	2.64	5.56	1.18
3-motion								
GPCA	31.95	32.93	19.83	19.55	16.85	28.66	28.66	28.26
LLMC $4K$	12.01	9.22	7.79	5.47	9.38	9.38	11.02	6.81
LLMC 5	10.70	9.21	2.91	**0.00**	5.60	5.60	8.85	3.19
LSA $4K$	5.80	1.77	25.07	23.79	7.25	7.25	9.73	2.33
LSA 5	30.37	31.98	27.02	34.01	23.11	23.11	29.28	31.63
LBF$(4K, 3)$	8.50	1.26	16.31	13.52	20.75	20.75	10.77	1.70
LBF-MS$(4K, 3)$	6.97	1.15	7.06	0.62	21.38	21.38	7.81	0.98
SLBF$(2F, 3)$	4.57	0.94	0.38	**0.00**	2.66	2.66	3.63	0.64
SLBF-MS$(2F, 3)$	3.33	0.39	**0.24**	**0.00**	**2.13**	**2.13**	2.64	**0.22**
SCC$(4K, 3)$	7.80	1.04	8.05	2.37	7.07	7.07	7.81	0.67
SCC-MS$(4K, 3)$	6.28	0.80	4.09	0.58	7.22	7.22	5.89	0.68
SSC-N$(4K, 3)$	**3.22**	**0.29**	0.53	**0.00**	**2.13**	**2.13**	**2.62**	**0.22**
MSL	10.38	4.61	1.80	**0.00**	2.71	2.71	8.23	1.76
RANSAC	25.78	26.01	12.83	11.45	21.38	21.38	22.94	22.03

3.2 Clustering Results on Motion Segmentation Data

We test the proposed algorithms on the Hopkins 155 database of motion segmentation, which is available at http://www.vision.jhu.edu/data/hopkins155. This data set contains 155 video sequences along with the coordinates of certain features extracted and tracked for each sequence in all its frames. The main task is to cluster the feature vectors (across all frames) according to the different moving objects and background in each video. It consists of three types of videos: checker, traffic and articulated (see Fig. 2 for demonstration of frames of such videos).

More formally, for a given video sequence, we denote the number of frames by F. In each sequence, we have either one or two independently moving objects, and the background can also move due to the motion of the camera. We let K be the number of moving objects plus the background, so that K is 2 or 3 (and distinguish accordingly between two-motions and three-motions). For each sequence, there are also N feature points $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N \in \mathbb{R}^3$ that are detected on the objects and the background. Let $\mathbf{z}_{ij} \in \mathbb{R}^2$ be the coordinates of the feature point $\mathbf{y}_j$ in the ith image frame for every $1 \le i \le F$ and $1 \le j \le N$. Then $\mathbf{z}_j = [\mathbf{z}_{1j}, \mathbf{z}_{2j}, \dots, \mathbf{z}_{Fj}] \in \mathbb{R}^{2F}$ is the trajectory of the jth feature point across the F frames. The actual task of motion segmentation is to separate these trajectory vectors $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N$ into K clusters representing the K underlying motions.

It has been shown (Costeira and Kanade 1998) that under the affine camera model, the trajectory vectors corresponding to different moving objects and the background across the F image frames live in distinct affine subspaces of dimension at most three in $\mathbb{R}^{2F}$. Following this theory, we implement our algorithm with $d = 3$.

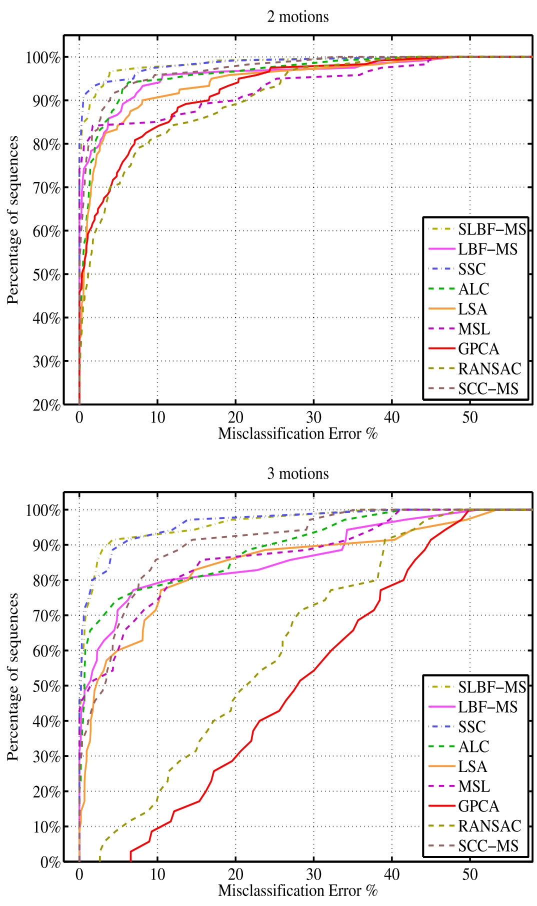

Fig. 1 The misclassification rate of some algorithms for the Hopkins 155 database. The y-axis represent the percentage of data sets that have misclassification rates (under corresponding algorithms) lower than that of x-axis

Fig. 2 Frames of the traffic, articulated and checker (from left to right) videos in Hopkins 155 database

We compare our algorithm with the following ones: improved GPCA for motion segmentation (GPCA) (Vidal et al. 2008), K-flats (KF) (Ho et al. 2003) (implemented for linear subspaces), Local Linear Manifold Clustering (LLMC) (Goh and Vidal 2007), Local Subspace Analysis (LSA) (Yan and Pollefeys 2006), Multi Stage Learning (MSL) (Sugaya and Kanatani 2004), Spectral Curvature Clustering (SCC) (Chen and Lerman 2009) and SCC-MS (see description earlier), Sparse Subspace Clustering (SSC) (Elhamifar and Vidal 2009), and RANSAC for HLM (Yang et al. 2006).

For GPCA (improved for motion segmentation), LLMC, LSA, MSL and RANSAC (for HLM), we copy the results from http://www.vision.jhu.edu/data/hopkins155 (they are based on experiments reported in Tron and Vidal (2007) and Goh and Vidal (2007)). We perform our own experiments for SCC, SCC-MS, SSC-N (SSC-B is not reported since it did not perform as well as SSC-N), LBF, LBF-MS, SLBF, SLBF-MS, we perform the experiments on our own and record the mean misclassification rate and the median misclassification rate for each algorithm for any fixed K (two or three-motions) and for the different type of motions ("checker", "traffic" and "articulated"). Each experiment (testing the latter set of algorithms) was repeated 500 times. The average misclassification rates, standard deviation and running time are recorded in Tables 5 and 6 and demonstrated in Fig. 1.

Our misclassification rates for SCC are different than Chen and Lerman (2009) and Lauer and Schnorr (2009) and our misclassification rates for SSC are different than Elhamifar and Vidal (2009) (the difference between our and their results differ more than twice the standard deviations of misclassification rates obtained here). This can be explained by possible evolutions of the codes since then (at least for SSC). We remark though that the misclassification rates of SCC-MS here are even slightly better than the misclassification rates of SCC in Chen and Lerman (2009).

From Table 2 and Fig. 1 we can see that our algorithms work well for the Hopkins database. Of all the methods tested, SLBF-MS and SSC-N are the most accurate algorithms. Besides SLBF/SLBF-MS and SSC-N, only SCC-MS is better than LBF-MS. However, from Table 4, LBF-MS ran more than 100 times faster than SSC-N and SLBF-MS is also more than 10 times faster than SSC. In many of the cases, the ℓ_1 energy (as well as the ℓ_2 energy) was lower for the labels obtained by LBF than the true labels. We thus suspect that the reason SLBF/SLBF-MS works better than LBF/LBF-MS is that good clustering of the Hopkins data requires additional type of information (e.g., spectral information) to be combined with subspace clustering (i.e., hybrid linear modeling).

By adapting the parameters of SLBF-MS (or alternatively, SLBF, LBF, LBF-MS), we can further improve its misclassification rates on Hopkins 155 (e.g., total 0.81 % for two-motions and 2.12 % for three-motions by SLBF-MS). However, we have fixed in advance all parameters and insisted using the same parameters on all four kinds of data (see the third paragraph of Sect. 3).

From Table 3 we can see that SLBF-MS, SLBF and SSC-N have negligible randomness. Indeed, their only randomness come from the K-means step, but this randomness is

Table 3 The standard deviation to the mean and median percentage of misclassified points for two-motions and three-motions in Hopkins 155 database

	Checker		Traffic		Articulated		All	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median
2-motion								
LBF$(4K, 3)$	0.71	**0.00**	1.22	**0.00**	1.04	0.66	0.50	**0.00**
LBF-MS$(4K, 3)$	0.53	**0.00**	1.06	**0.00**	1.14	0.28	0.47	**0.00**
WLBF$(4K, 3)$	0.53	**0.00**	0.98	**0.00**	1.35	**0.00**	0.47	**0.00**
SLBF-MS$(4K, 3)$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
SCC$(4K, 3)$	0.27	**0.00**	1.51	**0.00**	2.34	1.52	0.38	**0.00**
SCC-MS$(4K, 3)$	0.33	**0.00**	0.25	**0.00**	1.03	0.46	0.25	**0.00**
SSC-N$(4K, 3)$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3-motion								
LBF$(4K, 3)$	1.52	0.58	3.71	9.69	7.37	7.37	1.43	0.65
LBF-MS$(4K, 3)$	1.48	0.45	3.81	2.35	6.59	6.59	1.42	0.40
SLBF$(4K, 3)$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
SLBF-MS$(4K, 3)$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
SCC$(4K, 3)$	1.20	0.53	5.70	7.00	1.77	1.77	1.43	0.49
SCC-MS$(4K, 3)$	0.94	0.50	3.25	0.54	2.54	2.54	0.92	0.33
SSC-N$(4K, 3)$	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

Table 4 Average total computation times for all 155 sequences

RANSAC	LBF-MS$(4K, 3)$	LBF$(4K, 3)$	SCC-MS$(4K, 3)$	SLBF-MS$(2F, 3)$	SLBF$(2F, 3)$	SSC-N$(4K, 3)$
60 s	73 s	91 s	196 s	28 min	31 min	427 min

effectively reduced because of the restarting strategy. LBF and LBF-MS are more random, but still have comparable standard deviations with other good algorithms on Hopkins 155 database such as SCC/SCC-MS.

3.3 Clustering Results on the Extended Yale Face Database B

We test LBF, LBF-MS, SLBF and SLBF-MS and compare them with ALC, K-flats, and SSC on the extended Yale face database B (Lee et al. 2005), which is available on http://vision.ucsd.edu/leekc/ExtYaleDatabase/ExtYaleB.html. We will see that this data set shows a failure mode of our algorithms; and we will show how we can engineer a work-around.

We use subsets of the extended Yale face database B consisting of face images of $K = 2, 3, \ldots, 10$ persons under 64 varying lighting conditions. Our objective is to cluster these images according to the persons. In implementation, for any fixed K we repeat each algorithm on 100 randomly chosen subsets of K persons. The HLM model is applicable to this database, because the images of a face under variable lighting lies in a three-dimensional linear subspace if shadow is not considered (Lee et al. 2005), and a nine-dimensional subspace with shadow considered (Basri and Jacobs 2003). In our experiments, we found that the images of a person in this database lie roughly in a 5-dimensional subspace, and therefore we first reduce the dimension of the data to $5K$ (recall that K is the number of persons). We do not include the GPCA algorithm since it is slow and does not work well on this database. We also do not include SCC and RANSAC since the code provided returns errors for some examples. The setting of ALC (voting with K) follows Rao et al. (2010, Sect. 2.2) exactly: it chooses ε from 101 values in the range 10^{-5}–10^{3} (see the code in http://perception.csl.uiuc.edu/coding/motion/#Software).

We can see from the first row of Table 5 that LBF does a poor job discriminating the linear clusters in this data set. The failure occurs because of a combination of two factors: the first is the relatively sparse sampling of the data, with only 64 points per 5-dimensional cluster, and the second, the relative nearness of the underlying subspaces to each other. In particular, almost any neighborhood of any given point (even very small neighborhood) has points from the other affine clusters and consequently there is no “optimal” scale. For example, in the 128 face images from persons 1

Table 5 Mean percentage of misclassified points and mean running time on clustering the extended Yale face database

K		2	3	4	5	6	7	8	9	10
LBF (without whitening)	e %	32.49	54.42	57.45	56.00	56.24	56.94	59.53	59.66	60.74
	$t(s)$	0.24	0.48	0.82	1.26	1.93	2.97	4.18	5.81	8.05
LBF-MS (without whitening)	e %	18.27	36.22	48.24	50.18	49.99	50.68	53.08	54.06	54.73
	$t(s)$	0.12	0.21	0.36	0.57	0.89	1.41	2.06	2.98	4.13
LBF	e %	7.94	8.33	12.89	17.83	27.40	31.89	35.04	38.53	38.95
	$t(s)$	0.24	0.50	0.87	1.38	2.09	3.28	4.58	6.38	8.57
LBF-MS	e %	8.40	9.51	12.18	15.57	19.18	22.88	27.20	30.39	33.17
	$t(s)$	**0.12**	**0.22**	**0.37**	**0.58**	**0.89**	**1.41**	**2.07**	**2.94**	**4.02**
SLBF	e %	11.12	14.78	20.42	26.52	32.96	36.91	40.49	42.99	46.63
	$t(s)$	4.17	12.72	25.70	44.89	72.99	111.58	165.47	226.56	310.30
SLBF-MS	e %	9.12	12.48	18.61	25.27	30.50	33.97	36.22	38.66	41.44
	$t(s)$	3.84	12.20	23.88	41.24	64.10	95.73	142.09	192.34	262.40
ALC (voting with K)	e %	**3.46**	**6.08**	14.59	29.59	67.06	69.04	76.00	73.94	77.16
	$t(s)$	42.99	122.29	258.20	451.07	699.52	1090.96	1625.10	2384.69	3343.93
ALC (ϵ from LBF)	e %	10.43	15.23	32.20	42.15	58.10	62.54	70.84	81.14	84.25
	$t(s)$	0.95	2.49	5.54	11.54	24.38	45.27	78.05	132.35	211.15
SCC	e %	5.39	11.82	29.39	41.96	49.56	54.51	55.49	57.24	58.94
	$t(s)$	1.62	3.85	9.52	15.37	22.71	32.45	54.54	56.91	75.92
SCC-MS	e %	4.51	15.05	36.00	51.68	59.66	64.15	68.71	71.18	74.01
	$t(s)$	1.62	4.20	9.28	14.49	22.08	31.71	54.21	56.99	73.10
SSC	e %	6.45	8.10	**10.04**	**10.32**	**11.02**	**11.85**	**12.47**	**13.41**	**15.44**
	$t(s)$	28.36	46.45	67.11	92.75	128.46	182.65	259.66	340.12	612.21
K-flats	e %	7.20	12.12	19.06	26.77	32.59	35.18	38.58	42.00	44.40
	$t(s)$	0.16	0.37	0.76	1.29	2.14	3.25	5.18	6.91	9.60

and 2, more than a fifth of the points are closer to the subspace spanned by the first 5 principal components of the points in the other cluster than to their second nearest neighbors, and more than two thirds of the points are closer to the other subspace than to their 4th nearest neighbors. In some sense, this *is* a single 5-dimensional set, rather than two 5-dimensional sets. For example, the average distance of a point to the 5-dimensional best fit subspace by the points in the same cluster is 2.7×10^3, and the average distance to the 5-dimensional best fit subspace of the whole data set is 3.3×10^3, whereas the average norm of a point in the data set is 1.1×10^4. Thus one loses little in terms of relative fitting error by considering the set as spanned by a single subspace.

However, most points are actually closer to the subspace spanned by the same face than to the subspace spanned by the other face, if only by a little, and a global method such as SSC is still able to find and discriminate between the two affine clusters. The problem of data having large variance in directions irrelevant to a classification task is not unusual. A standard method of dealing with this situation is to "whiten" the data; i.e. reduce the value of the large singular values. A very crude whitening is obtained by simply removing the first few principal components. If we exclude the first two principal components after reducing the dimension to $5K$ for LBF/SLBF algorithms, we see in Table 5 that the results are greatly improved and become competitive.[2] With more sophisticated whitening, the results can be further improved.

3.4 Clustering Results on MNIST Data Set

Finally, we work on the MNIST data set (available at http://yann.lecun.com/exdb/mnist/). This data set consists of several thousand 28×28 images of the digits 0 through 9. We work on some subsets of the data which contain 2 or 3 digits and choose 200 images for each digit at random. We apply PCA to reduce the dimension to $D = 5$ for GPCA and to both $D = 10$ and $D = 50$ for the rest of algorithms. The choice of both $D = 10$ and $D = 50$ provide richer testing opportunities, this is however unavailable for GPCA, which cannot handle $D = 50$ and often get stuck with $D = 10$. We process the data the same way as in Sect. 3.3. We run

[2]Removing principal components harms the performance of the other algorithms.

Table 6 The standard deviation(%) to the mean percentage of misclassified points on the extended Yale face database

Real K	2	3	4	5	6	7	8	9	10
LBF (without whitening)	20.46	14.73	4.87	3.89	5.54	4.99	4.63	3.95	3.22
LBF-MS (without whitening)	18.23	18.77	13.40	6.74	4.52	5.51	5.31	4.90	4.14
LBF	5.27	3.73	7.97	9.86	11.21	10.38	8.27	6.52	6.20
LBF-MS	4.25	**3.08**	5.33	6.24	7.73	8.02	8.29	8.05	7.25
SLBF	4.76	5.37	5.08	5.25	5.48	5.42	4.57	4.74	3.79
SLBF-MS	4.77	5.37	5.84	4.91	3.75	3.76	**2.87**	**3.01**	**3.22**
ALC (voting with K)	**2.21**	6.93	13.87	14.89	16.84	24.71	18.05	21.62	16.98
ALC (ϵ from LBF)	13.14	12.96	14.91	16.40	15.22	12.22	10.89	6.76	6.10
SCC	5.21	11.71	14.65	10.60	6.68	5.14	4.67	4.32	5.03
SCC-MS	2.84	13.66	14.66	10.41	8.29	6.72	5.61	5.93	5.46
SSC	4.57	3.76	**4.52**	**3.82**	**3.59**	**2.87**	3.18	3.45	5.21
K-flats	4.67	6.86	8.53	8.89	7.29	6.41	6.67	4.84	5.43

Table 7 Mean percentage of misclassified points and mean running time on clustering MNIST data set ($D = 5$ for GPCA, $D = 10$ for other algorithms)

Subsets		[1 2]	[1 3]	[1 7]	[4 7]	[2 4 8]	[3 6 8]	[1 2 3]
K		2	2	2	2	3	3	3
LBF	e %	8.0	8.5	12.9	25.5	28.8	28.1	20.2
	$t(s)$	0.4	0.4	0.3	0.4	0.7	0.7	0.7
LBF-MS	e %	9.7	7.8	8.8	24.0	40.2	33.5	21.5
	$t(s)$	**0.2**	**0.2**	**0.2**	**0.2**	**0.5**	**0.4**	**0.4**
SLBF	e %	0.5	**1.0**	**2.0**	**3.0**	**3.8**	**19.7**	**17.3**
	$t(s)$	13.9	13.7	13.5	14.5	41.9	41.0	42.7
SLBF-MS	e %	0.5	**1.0**	**2.0**	**3.0**	**3.8**	19.7	17.3
	$t(s)$	12.8	13.7	13.0	14.6	38.6	46.3	39.0
ALC (voting with K)	e %	**0.2**	2.2	3.5	48.5	4.2	42.7	45.3
	$t(s)$	830.5	823.3	813.3	753.2	1789.5	1871.8	1987.7
ALC (ϵ from LBF)	e %	20.3	32.0	51.8	27.5	4.0	30.3	14.5
	$t(s)$	23.2	22.5	21.6	23.0	55.6	54.7	54.0
SCC	e %	7.0	6.4	11.4	23.4	22.8	26.7	39.2
	$t(s)$	1.2	1.5	1.4	1.3	2.5	2.7	2.3
SCC-MS	e %	6.3	7.9	10.5	23.2	23.3	26.9	32.8
	$t(s)$	0.9	0.8	1.1	1.0	1.9	1.9	1.5
GPCA	e %	22.3	30.8	32.5	47.0	48.2	33.8	31.0
	$t(s)$	8.7	9.2	9.4	10.8	24.9	24.5	22.5
K-flats	e %	11.1	6.8	6.3	29.1	43.9	40.7	29.2
	$t(s)$	0.4	0.4	0.4	0.4	0.9	0.8	0.6
SSC	e %	4.5	3.5	9.0	21.0	19.5	24.5	49.3
	$t(s)$	220.6	196.6	200.8	203.2	322.6	333.0	338.2

each experiment 500 times, using $d = 3$ and the correct number of clusters, and record the misclassification rates, the standard deviation and running time in Tables 7, 9, 8 and 10.

From Tables 7 and 8, SLBF and SLBF-MS are the best algorithms among all the methods in terms of misclassification rates, although these misclassification rates are larger when $K = 3$. SCC, SCC-MS, SSC, LBF and LBF-MS are also good algorithms for this data set. ALC is almost as good as SLBF and SLBF-MS when $K = 2$, but it fails when $K = 3$. LBF, LBF-MS and K-flats are the fastest algorithms in MNIST data set.

Table 8 Mean percentage of misclassified points and mean running time on clustering MNIST data set ($D = 50$)

Subsets		[1 2]	[1 3]	[1 7]	[4 7]	[2 4 8]	[3 6 8]	[1 2 3]
K		2	2	2	2	3	3	3
LBF	e %	20.5	13.1	18.2	30.2	26.3	24.1	**19.2**
	$t(s)$	2.8	2.8	2.6	3.1	5.2	5.1	4.7
LBF-MS	e %	12.5	16.9	10.7	19.1	23.5	27.3	24.3
	$t(s)$	1.3	1.3	1.3	1.3	2.3	2.3	2.3
SLBF	e %	8.3	4.3	**2.3**	13.8	4.3	**17.5**	21.7
	$t(s)$	15.1	15.0	14.6	16.8	37.5	38.5	39.5
SLBF-MS	e %	5.5	**3.3**	5.0	**5.5**	**3.2**	18.5	21.7
	$t(s)$	11.8	12.3	12.3	12.5	34.3	36.9	34.4
ALC (voting with K)	e %	47.0	46.0	48.8	100.0	100.0	100.0	65.3
	$t(s)$	1469.2	1445.6	1489.2	679.0	1530.1	1528.5	3032.4
ALC (ϵ from LBF)	e %	50.5	50.8	50.5	99.8	99.8	99.8	67.0
	$t(s)$	93.0	93.6	91.0	9.4	18.2	17.9	163.5
SCC	e %	5.8	4.9	5.3	17.1	23.0	29.7	33.6
	$t(s)$	**0.9**	**1.0**	**1.1**	**0.9**	**1.6**	2.0	**1.7**
SCC-MS	e %	**5.1**	5.4	5.1	26.2	28.6	41.7	33.0
	$t(s)$	**0.9**	**1.0**	1.2	1.0	1.8	**1.9**	2.0
GPCA	e %	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	$t(s)$	N/A	N/A	N/A	N/A	N/A	N/A	N/A
K-flats	e %	10.9	14.9	13.5	30.4	45.3	41.6	26.9
	$t(s)$	2.8	2.9	2.9	3.1	6.2	5.6	5.1
SSC	e %	16.8	2.0	3.2	20.0	11.3	17.8	45.5
	$t(s)$	411.8	402.7	395.1	396.0	760.9	763.1	777.0

3.5 Automatic Determination of the Number of Flats

We explain how to use the elbow method to determine the number of affine clusters in any HLM algorithm, in particular LBF and SLBF. Fixing an arbitrary HLM algorithm with the correct input of number of clusters K, let F_j, $j = 1, \dots, K$ be the K flats returned by that algorithm and W_K be the sum of squared distances of all data points to the flat, among these K flats, corresponding to their clusters. That is,

$$W_K = \sum_{j=1}^{K} \sum_{\mathbf{x} \in C_j} \operatorname{dist}^2(\mathbf{x}, F_j). \tag{12}$$

We note that W_K decreases as K increases.

A classical method for determining the number of clusters is to find the "elbow", or the K past which adding more clusters does not significantly decrease the error. We search for the elbow by finding the maximum of the Second Order Difference (SOD) of the logarithm of W_K (Yue et al. 2008):

$$\operatorname{SOD}(\ln W_K) = \ln W_{K-1} + \ln W_{K+1} - 2 \ln W_K. \tag{13}$$

The optimal K is thus found by

$$K_{\text{SOD}} = \arg\max_K \operatorname{SOD}(\ln W_K), \tag{14}$$

where $K = 2, \dots, K_{\max}$.

In the following sections, we compare SOD (LBF), i.e., SOD applying LBF, SOD (LBF-MS), SOD (SLBF), SOD (SLBF-MS), SOD (SCC), SOD (SCC-MS) and SOD(SSC) with ALC (Ma et al. 2007) and part of GPCA (Vidal et al. 2005) on a number of artificial data sets and real data sets. These experiments run on a machine with Intel Core 2 Quad Q6600 at 2.4 GHz and 8 GB memory.

3.5.1 Finding the Number of Clusters on Artificial Data

We test SOD with LBF and SLBF on artificial data (where the number of clusters is not provided to the user) and compare them with some other methods (three variations of ALC, number of clusters by GPCA and SOD with SSC and SCC). The artificial data sets are generated by the Matlab code borrowed from the GPCA (Vidal et al. 2005) package on http://perception.csl.uiuc.edu/gpca. For each subspace $100d$ initial data points are uniformly sampled in a unit cube in this subspace centered around the origin and

Table 9 The standard deviation to the mean percentage of misclassified points on clustering MNIST data set ($D = 5$ for GPCA, $D = 10$ for other algorithms)

Subsets	[1 2]	[1 3]	[1 7]	[4 7]	[2 4 8]	[3 6 8]	[1 2 3]
K	2	2	2	2	3	3	3
LBF	3.5	4.1	10.0	11.4	11.6	8.3	9.5
LBF-MS	5.9	3.8	10.0	10.0	10.3	7.2	7.8
SLBF	**0.0**	**0.0**	**0.0**	**0.0**	0.0	**0.0**	**0.0**
SLBF-MS	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ALC (voting with K)	**0.0**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ALC (ϵ from LBF)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
SCC	2.3	2.7	4.6	9.9	9.4	7.5	11.9
SCC-MS	2.0	3.7	5.2	10.2	8.3	8.5	9.2
GPCA	**0.0**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
K-flats	7.6	8.5	7.8	5.7	7.4	7.5	5.9
SSC	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

Table 10 The standard deviation of the mean percentage of misclassified points on clustering MNIST data set ($D = 50$)

Subsets	[1 2]	[1 3]	[1 7]	[4 7]	[2 4 8]	[3 6 8]	[1 2 3]
K	2	2	2	2	3	3	3
LBF	5.6	8.0	8.3	10.6	11.0	6.0	6.0
LBF-MS	8.7	10.5	11.4	11.2	12.3	8.9	9.1
SLBF	**0.0**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
SLBF-MS	**0.0**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ALC (voting with K)	**0.0**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
ALC (ϵ from LBF)	**0.0**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
SCC	0.6	1.0	0.9	10.3	3.7	4.3	13.9
SCC-MS	0.4	0.7	0.9	15.5	5.4	4.5	5.8
GPCA	N/A	N/A	N/A	N/A	N/A	N/A	N/A
K-flats	7.2	11.3	11.1	7.5	7.3	8.1	7.7
SSC	**0.0**	0.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

then corrupted with Gaussian noise in $\mathbb{R}^D$ of standard deviation 0.05. For the last four experiments, we restrict the angle between subspaces to be at least $\pi/8$ for separation. The dimension d is given and we let $K_{\max} = 10$ in SOD.

In ALC (voting), we try 101 different values from 10^{-5} to 10^3 for ϵ (as in Rao et al. 2010) and choose the estimated K by majority. In ALC (ϵ from LBF), we choose the average noise in the neighborhood using the local best-fit heuristic as the distortion rate ϵ. In ALC (oracle), we input the true noise level ($\epsilon = 0.05$) as the distortion rate. For GPCA, we use the original idea of Vidal et al. (2005, Eqs. (26), (28)) to find the number of clusters (see our implementation in the supplemental webpage). We project the data onto a $(d+1)$-dimensional subspace by PCA and let the tolerance of rank detection be 0.05 (chosen by trying different values and picking the one obtaining the lowest error). This algorithm is independent of other parts of the GPCA algorithm and is thus extremely fast and can perform in high ambient dimensions. We even tried other ideas of Ma et al. (2008, Eqs. (3.28), (3.29)) (for the same given dimension d), while applying them to several HLM algorithms (even though they were originally presented for GPCA). Nevertheless, they did not work well and we thus did not report them. Each experiment is repeated 100 times (except for SOD(SSC), which is repeated 10 times due to its low speed) and the error rates of finding the number of clusters K and the computation time (in seconds) are recorded in Table 11.

As in Table 11, ALC (oracle) and ALC (ϵ from LBF) work the best for low dimensions ($d = 1, 2, 3$), but in real problems this choice (the noise level) for ϵ is usually unknown. The local best-fit flat heuristic provides a good estimation for the distortion rate and helps ALC reduce its running time. ALC (voting) is not as good as SOD (LBF) for artificial data. All options of ALC suffer from the computation complexity, especially ALC (voting). SOD (LBF) and SOD (LBF-MS) get reasonable outputs and have obvious advantage of computing time. GPCA is very fast, but does not work well.

Table 11 The mean percentage of incorrectness (e %) for finding the number of clusters K and the computation time in seconds $t(s)$ on artificial data

		No minimum angle							Minimum angle $= \pi/8$			
		$1^6 \in \mathbb{R}^5$	$2^4 \in \mathbb{R}^5$	$3^3 \in \mathbb{R}^5$	$1^6 \in \mathbb{R}^3$	$2^4 \in \mathbb{R}^3$	$3^3 \in \mathbb{R}^4$	$10^2 \in \mathbb{R}^{15}$	$1^6 \in \mathbb{R}^3$	$2^4 \in \mathbb{R}^3$	$3^3 \in \mathbb{R}^4$	$10^2 \in \mathbb{R}^{15}$
SOD (LBF)	e %	22	2	**0**	58	32	12	**0**	2	6	5	**0**
	$t(s)$	10.43	13.76	14.83	9.84	13.08	14.49	34.16	9.95	13.22	14.47	34.04
SOD (LBF-MS)	e %	13	1	3	67	33	9	**0**	3	8	6	**0**
	$t(s)$	8.70	11.90	12.92	8.37	11.54	12.84	27.56	8.42	11.60	12.84	27.69
SOD (SLBF)	e %	75	10	5	**0**	90	95	55	**0**	85	90	55
	$t(s)$	1097.19	2148.06	2895.85	1076.24	1774.74	2629.26	16124.50	1224.96	2387.70	2830.83	16510.13
SOD (SLBF-MS)	e %	90	95	70	**0**	90	85	85	0	75	80	80
	$t(s)$	908.76	2094.68	3141.77	927.25	1740.03	2695.59	15754.05	990.88	2302.66	3010.64	16493.95
ALC (voting)	e %(K)	24	12	11	32	30	17	100	5	9	9	100
	$t(s)$	2094.75	2700.07	3530.26	1207.54	2346.69	3628.24	119584.04	1184.08	2354.19	3956.05	117353.17
ALC (ϵ from LBF)	e %(K)	**1**	**0**	1	20	**20**	3	58	**0**	**3**	**1**	63
	$t(s)$	23.72	43.50	57.50	19.76	36.67	53.25	1516.02	19.81	36.60	53.01	1770.77
ALC (oracle)	e %(K)	**1**	**0**	**0**	34	31	**1**	16	**0**	10	**1**	13
	$t(s)$	23.74	43.44	59.14	20.49	37.49	53.59	1370.92	20.22	37.41	54.11	1354.11
GPCA	e %(K)	88	100	100	27	100	100	100	13	100	100	100
	$t(s)$	**0.03**	**0.09**	**0.12**	**0.06**	**0.09**	**0.12**	**1.30**	**0.04**	**0.09**	**0.12**	**1.30**
SOD (SCC)	e %(K)	35	21	1	63	39	17	**0**	9	32	11	1
	$t(s)$	32.09	61.26	95.79	25.83	59.41	76.13	475.45	26.74	41.95	61.53	466.79
SOD (SCC-MS)	e%(K)	71	32	2	80	50	12	**0**	46	33	3	**0**
	$t(s)$	31.78	67.77	111.15	22.29	55.25	74.07	475.50	24.53	51.98	75.03	471.31
SOD (SSC)	e %(K)	10	80	70	100	70	70	100	50	80	80	100
	$t(s)$	39.88	2634.80	3039.55	1708.37	2447.01	2925.27	14918.10	1452.43	2101.84	2641.68	14227.32

3.5.2 Finding the Number of Clusters on the Extended Yale Face Database B

We use the extended Yale face database B as in Sect. 3.3 for testing the above algorithms for detecting the number of clusters. The ambient dimension is reduced to $D = 5K$ by PCA for all of the methods and the intrinsic dimension of subspaces is set as $d = 5$ (see Sect. 3.3). For SOD with different clustering algorithms, we let $K_{\max} = 6$, 8, 8, 10 and 10 respectively for 2 to 6 clusters. For GPCA, we let tolerance be 0.05 which does not affect the performance in this experiment. Each experiment is repeated 500 times (except for SOD(SSC), which is repeated 10 times due to its low speed). Following Sect. 3.3, we apply LBF, LBF-MS, SLBF and SLBF-MS with whitening. The error rates of finding the correct number of clusters and the computation time are recorded in Table 12.

We see from Table 12 that SOD only performs well with SSC with K smaller than 4. We note that this is due to the difficulty of the database. Indeed for a simpler database such as Yale Face database B (Georghiades et al. 2001) of uncropped faces, SOD (SLBF), SOD (SLBF-MS), ALC (ϵ from LBF) and ALC (voting) have perfect detection for $K \leq 10$ (whitening is not applied then).

3.5.3 Finding the Number of Clusters on MNIST Data Set

We preprocess MNIST data set exactly the same way as we did in Sect. 3.4. The ambient dimension is reduced to both $D = 10$ and $D = 50$ by PCA for all of the methods including GPCA and 3 is given as the intrinsic subspace dimension. For SOD with different clustering algorithms, we let $K_{\max} =$ 6, and 8 respectively for 2 and 3 clusters. For GPCA, we let the tolerance be 0.05 which does not affect the performance in this experiment. Each experiment is repeated 500 times (except for SOD(SSC), which is repeated 10 times due to its low speed). The error rates of finding the correct number of clusters and the computation time are recorded in Tables 13 and 14.

For all the methods, determining the number K of clusters becomes very difficult when the real K is larger than 3. For real $K \leq 3$, we see from Table 13 that when we project data to 10-dimensional space, ALC and GPCA fail in most cases, except for ALC (ϵ from LBF) on digits [3 6 8]. SOD (SLBF), SOD (SLBF-MS) and SOD (SSC) outperform all others although they are not very efficient.

Table 12 The mean percentage of incorrectness (e %) for finding the correct number of clusters K and the computation time in seconds $t(s)$ on the extended Yale face database

Real K		2	3	4	5	6
SOD (LBF)	$e\,\%(K)$	62	61	69	78	84
	$t(s)$	1.30	3.60	5.69	11.30	15.84
SOD (LBF-MS)	$e\,\%(K)$	65	75	78	81	80
	$t(s)$	0.67	1.65	2.49	4.90	6.83
SOD (SLBF)	$e\,\%(K)$	24	60	70	86	98
	$t(s)$	115.97	303.02	338.35	729.74	811.40
SOD (SLBF-MS)	$e\,\%(K)$	**20**	60	76	92	96
	$t(s)$	106.87	272.88	306.22	649.50	721.42
ALC (voting)	$e\,\%(K)$	100	100	100	100	100
	$t(s)$	42.99	122.29	258.20	451.07	699.52
ALC (ϵ from LBF)	$e\,\%(K)$	42	36	76	86	100
	$t(s)$	0.95	2.49	5.54	11.54	24.38
GPCA	$e\,\%(K)$	100	100	100	100	100
	$t(s)$	**0.07**	**0.13**	**0.52**	**0.71**	**1.02**
SOD (SSC)	$e\,\%(K)$	100	**8**	**12**	**28**	**38**
	$t(s)$	172.50	389.66	567.39	1015.99	1336.57

Table 13 The mean percentage of incorrectness (e %) for finding the correct number of clusters K and the computation time in seconds $t(s)$ on MNIST data set ($D = 10$)

Subsets		[1 2]	[1 3]	[1 7]	[4 7]	[2 4 8]	[3 6 8]	[1 2 3]
K		2	2	2	2	3	3	3
SOD (LBF)	e %	16.8	3.8	50.8	50.4	75.6	70.0	54.8
	$t(s)$	3.5	3.2	3.0	3.3	7.7	7.5	7.3
SOD (LBF-MS)	e %	9.6	6.6	33.4	68.2	80.4	76.6	44.2
	$t(s)$	1.9	1.9	1.9	1.8	4.6	4.6	4.7
SOD (SLBF)	e %	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	100.0	**0.0**
	$t(s)$	173.9	164.6	160.3	248.6	710.1	610.9	548.5
SOD (SLBF-MS)	e %	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	100.0	**0.0**
	$t(s)$	164.6	159.9	150.1	228.5	676.6	586.4	556.2
ALC (voting)	e %	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	$t(s)$	830.4	823.2	813.2	753.2	1789.5	1871.8	1987.5
ALC (ϵ from LBF)	e %	100.0	100.0	100.0	100.0	100.0	**0.0**	100.0
	$t(s)$	23.2	22.5	21.5	22.9	55.6	54.7	54.0
GPCA	e %	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	$t(s)$	**1.0**	**1.0**	**1.0**	**1.1**	**2.8**	**2.8**	**2.7**
SOD (SCC)	$e\,\%(K)$	3.8	7.8	66.4	81.8	64.4	47.6	82.6
	$t(s)$	14.5	13.3	14.7	16.9	37.5	34.1	35.0
SOD (SCC-MS)	$e\,\%(K)$	2.4	16.4	53.0	77.4	70.4	49.6	77.8
	$t(s)$	13.7	13.8	13.5	16.4	38.0	35.6	29.4
SOD (SSC)	$e\,\%(K)$	**0.0**	**0.0**	**0.0**	100.0	**0.0**	100.0	100.0
	$t(s)$	233.6	210.3	213.3	218.4	380.0	386.4	390.5

3.6 Initializing K-Flats with the Local Best-Fit Heuristic

Here we demonstrate that our choice of neighborhoods in Algorithm 1 can be used to get a more robust initialization of K-flats. We work with geometric farthest insertion. For fixed neighborhood sizes, say of m neighbors, this goes as follows: we pick a random point $\mathbf{x}_0$ and then find the best-fit flat F_0 for the m point neighborhood of $\mathbf{x}_0$. Then we find the point $\mathbf{x}_1$ in our data farthest from F_0, find the best-fit flat F_1 of the m neighborhood of $\mathbf{x}_1$, and then choose the point $\mathbf{x}_2$ farthest from F_0 and F_1 to continue. We stop when we have K flats; we use these as an initialization for K-flats.

Table 14 The mean percentage of incorrectness (e %) for finding the correct number of clusters K and the computation time in seconds $t(s)$ on MNIST data set ($D = 50$)

Subsets		[1 2]	[1 3]	[1 7]	[4 7]	[2 4 8]	[3 6 8]	[1 2 3]
K		2	2	2	2	3	3	3
SOD (LBF)	e %	45.0	35.0	54.0	79.0	72.0	67.0	60.0
	$t(s)$	22.9	23.5	22.2	24.9	56.2	54.6	51.1
SOD (LBF-MS)	e %	32.0	22.0	38.0	66.0	44.0	82.0	58.0
	$t(s)$	**12.2**	12.2	12.2	12.2	29.3	29.4	29.4
SOD (SLBF)	e %	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	100.0	100.0
	$t(s)$	204.2	198.1	207.8	295.8	864.5	766.5	706.1
SOD (SLBF-MS)	e %	**0.0**	**0.0**	100.0	**0.0**	**0.0**	100.0	100.0
	$t(s)$	213.7	201.7	176.6	259.9	748.1	640.0	681.1
ALC (voting)	e %	100.0	100.0	100.0	100.0	100.00	100.0	100.0
	$t(s)$	1469.2	1445.6	1489.2	679.0	1530.1	1528.5	3032.4
ALC (ϵ from LBF)	e %	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	$t(s)$	93.0	93.6	91.0	**9.4**	**18.2**	**17.9**	163.5
GPCA	e %	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	$t(s)$	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SOD (SCC)	$e\,\%(K)$	**0.0**	4.0	1.0	50.5	78.8	30.3	83.8
	$t(s)$	14.9	**10.6**	**11.6**	11.6	24.7	26.2	**25.4**
SOD (SCC-MS)	$e\,\%(K)$	**0.0**	**0.0**	**0.0**	42.4	89.9	97.0	93.9
	$t(s)$	12.6	13.0	14.7	13.9	34.0	36.8	30.7
SOD (SSC)	$e\,\%(K)$	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	100.0	100.0
	$t(s)$	426.4	417.6	409.3	413.5	823.8	821.2	836.8

We work on three data sets. Data set #1 consists of 1500 points on three parallel 2-planes in $\mathbb{R}^3$. 500 points are drawn from the unit square in x, y plane, and then 500 more from the $x, y, z + 0.2$ plane, and then 500 more from the $x, y, z + 0.4$ plane. This data set is designed to favor the use of small neighborhoods. The next data set is three random flats with 15 % Gaussian noise and 5 % outliers, generated using the Matlab code from GPCA, as in Sect. 3.1. This data set is designed to favor large neighborhood choices. Finally, we work on a data set with 1500 points sampled from 3 planes in $\mathbb{R}^2$ as in Fig. 3. The error rates of K-flats with farthest insertion initialization with fixed neighborhoods of size 10, 20, . . . , 160 are plotted against the error rates for farthest insertion with adapted neighborhoods (searched over the same range), averaged over 400 runs in Fig. 4. Although our method did not always beat the best fixed neighborhood, it was quite close; and it always significantly better than the wrong fixed neighborhood size. Both methods did significantly better than a random initialization.

In Fig. 3 we plot the number of neighbors picked by our algorithm for each point of a realization of data set #3.

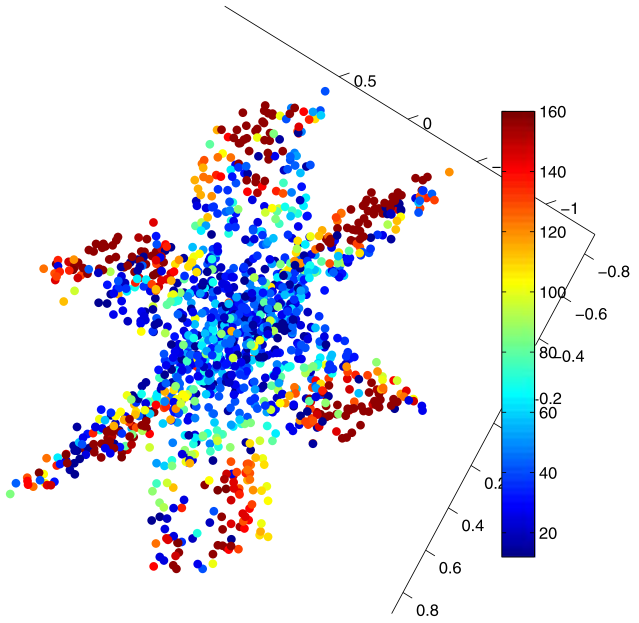

Fig. 3 Color map of neighborhood size obtained by the local best–fit flat heuristic. The *color value* represents the number of neighbors chosen at that point. Note that the algorithm chooses smaller neighborhoods for points closer to the intersection of the planes (Color figure online)

4 Conclusions and Future Work

We presented a very simple geometric method for HLM based on selecting a set of local best-fit flats. The size of the local neighborhoods is determined automatically using the ℓ_2 β numbers; it is proven under certain geometric conditions that our method approximately finds the optimal local

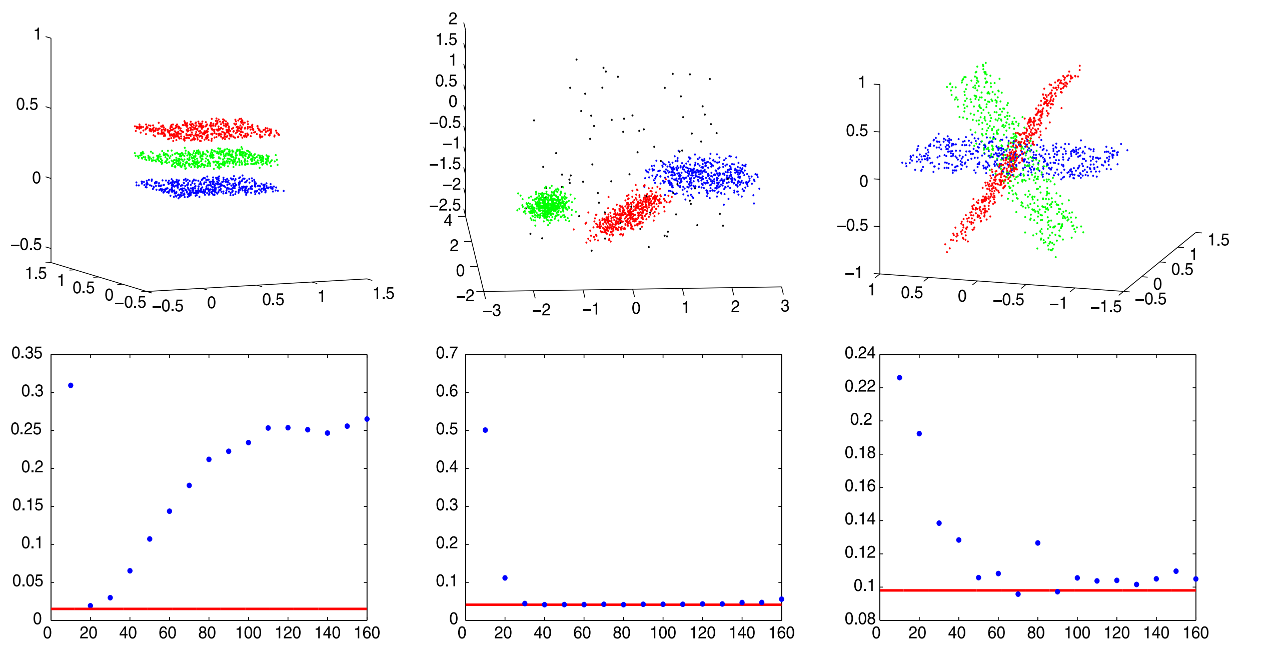

Fig. 4 Using our neighborhood choice to improve initialization of k-flats: the first row is the visualization of three data sets, and the seconds row shows the corresponding figures such that the vertical axis is accuracy, and the horizontal axis is fixed neighborhood size in geometric farthest insertion for initialization of K flats. The *red line* is the result of using adapted neighborhoods. The data sets are #1, #2, and #3 as described in Sect. 3.6. Random initialization leads to misclassification rates of 0.4 or greater for all three data sets (Color figure online)

neighborhoods. We give extensive experimental evidence demonstrating the state of the art accuracy and speed of the algorithm on synthetic and real hybrid linear data.

We believe that one promising next step is to adapt the method for multi-manifold clustering. As it is, our method, while quite good at unions of flats, cannot successfully handle unions of curved manifolds. We expect that by gluing together groups of local best-fit flats related by some smoothness conditions, we will be able to approach the problem of clustering data which lies on unions of smooth manifolds.

We also believe that it will be possible to provide a theoretical framework for performance guarantees with noise for LBF and SLBF. Specifically, we hope to prove a quantitative form of the following alternative: suppose the data lies on the union of d-dimensional affine sets, perhaps with additive noise and outliers. Then either

1. Most points are roughly as close to an affine set they don't belong to as they are to their nearest $O(d)$ neighbors;
2. A large fraction of the points have optimal neighborhoods contained in only one of the affine clusters, the principal components of these neighborhoods are good approximations to the clusters; and LBF and SLBF recover good approximations to the two affine clusters, or
3. The data looks locally lower than d-dimensional, even though each cluster is globally d-dimensional, and has high curvature; in this case, there are pure optimal neighborhoods, but the local estimation does not accurately represent the affine clusters.

Acknowledgements This work was supported by NSF grants DMS-06-12608, DMS-08-11203, DMS-09-15064 and DMS-09-56072. Thanks to the action editor and the reviewers for the careful reading and comments; Peter Jones, Mauro Maggioni and Amit Singer for discussions that motivated our exploration for a multiscale SVD-based HLM algorithm; Ehsan Elhamifar and René Vidal for answering various questions regarding the SSC code and providing us an initial version before the code was available to the public; Allen Yang for clarifying the estimation of the number of clusters in GPCA; and the IMA for a stimulating multi-manifold modeling workshop.

Appendix A: Proof of Theorem 1

Assume without loss of generality that $i^* = 1$. Note that when $r \le r_0$, $B(\mathbf{x}^*, r) \cap T(L_1, w) = B(\mathbf{x}^*, r) \cap \operatorname{supp}(\mu)$ and that L_1 is the minimizer of the RHS of (2). Combining these observations with (2) and the fact that $\beta_2(\mathbf{x}^*, r)$ is invariant to scaling of r and w, we immediately obtain that for $r < r_0$:

$$\beta_2^2(\mathbf{x}^*, r) = \frac{\int_{T(L_1, \frac{w}{\max(r,w)}) \cap B(\mathbf{x}^*, 1)} \operatorname{dist}(\mathbf{x}^*, L_1)^2 \, \mathrm{d}\mu_1}{\mu_1(T(L_1, \frac{w}{\max(r,w)}) \cap B(\mathbf{x}^*, 1))}. \tag{15}$$

In particular, $\beta_2(\mathbf{x}^*, r)$ is constant for all $0 \le r \le w$.

To show that $\beta_2(\mathbf{x}^*, r)$ is strictly decreasing whenever $w \le r \le r_0$, we first note that for any r_1 and r_2 satisfying $w \le r_1 \le r_2 \le r_0$:

$$T\left(L_1, \frac{w}{\max(r_2, w)}\right) \cap B\left(\mathbf{x}^*, 1\right) \subset T\left(L_1, \frac{w}{\max(r_1, w)}\right) \cap B\left(\mathbf{x}^*, 1\right). \tag{16}$$

Moreover, any point in $T(L_1, \frac{w}{\max(r_1,w)}) \setminus T(L_1, \frac{w}{\max(r_2,w)})$ has a larger distance to L_1 than any point in $T(L_1, \frac{w}{\max(r_2,w)})$. Combining these observations with (15), we conclude that $\beta_2(\mathbf{x}^*, r_1) > \beta_2(\mathbf{x}^*, r_2)$, i.e., $\beta_2(\mathbf{x}^*, r)$ is strictly decreasing on $[w, r_0]$.

Next, we will prove (5) with a weaker requirement on r^*. More precisely, we define $r^* = \max(r_1^*, r_2^*)$, where

$$r_1^* = \begin{cases} \dfrac{r_0+2w}{\sqrt{1-\frac{3\sqrt{2}(D-1)Kw^2}{(D+1)r_0(r_0+2w)}}}, & \text{when } d = 1; \\ \dfrac{r_0+2w}{\sqrt{1-(\frac{6(D-d)Kw^2}{(D-d+2)(r_0+2w)^2})^{\frac{2}{d}}}}, & \text{when } d > 1 \end{cases} \tag{17}$$

and

$$r_2^* = \frac{1}{\sqrt{\frac{2}{(r_0+2w)^2} - \frac{1}{r_0^2}}}. \tag{18}$$

We further assume that $w < r_0$ and

$$\frac{(r^{*2} - r_0^2)^{\frac{d}{2}} \cdot (r_0 + w)}{(r^{*2} - r_0^2)^{\frac{d}{2}} + (r^{*2} - w^2)^{\frac{d}{2}}} + \frac{r^*}{\sqrt{d+1}} \le r_0. \tag{19}$$

We will later show that (4) implies (19) and we will also verify that $r_0 \le r^* < 1.09 r_0$.

Without loss of generality we assume that

$$\operatorname{argmin}_{i>1} \operatorname{dist}(\mathbf{x}^*, L_i) = 2,$$

and let $\mathbf{x}_0$ be the center of mass of $\mu_1 + \mu_2$ in $B(0, r)$. Then for any $r > r_0$

$$\begin{aligned} &\min_L \int_{B(\mathbf{x}^*,r)} \left(\frac{\operatorname{dist}(\mathbf{x}, L)}{r}\right)^2 \mathrm{d}\mu \\ &\ge \min_L \int_{B(\mathbf{x}^*,r)} \left(\frac{\operatorname{dist}(\mathbf{x}, L)}{r}\right)^2 \mathrm{d}(\mu_1 + \mu_2) \\ &= \min_{L:\mathbf{x}_0 \in L} \int_{B(\mathbf{x}^*,r)} \left(\frac{\operatorname{dist}(\mathbf{x}, L)}{r}\right)^2 \mathrm{d}(\mu_1 + \mu_2) \\ &\ge \min_L \int_{B(\mathbf{x}^*,r)} \left(\frac{\operatorname{dist}(\mathbf{x}, L)}{r}\right)^2 \mathrm{d}\mu_1 \\ &\quad + \min_{L:\mathbf{x}_0 \in L} \int_{B(\mathbf{x}^*,r)} \left(\frac{\operatorname{dist}(\mathbf{x}, L)}{r}\right)^2 \mathrm{d}\mu_2 \\ &= \int_{B(\mathbf{x}^*,r)} \left(\frac{\operatorname{dist}(\mathbf{x}, L_1)}{r}\right)^2 \mathrm{d}\mu_1 \\ &\quad + \min_{L:\mathbf{x}_0 \in L} \int_{B(\mathbf{x}^*,r)} \left(\frac{\operatorname{dist}(\mathbf{x}, L)}{r}\right)^2 \mathrm{d}\mu_2. \end{aligned} \tag{20}$$

We claim that when $r = r^*$, the minimizer in the second expression in the RHS of (20) (denoted by L_0) satisfies that $\dim(L_0 \cap L_2) = d - 1$ and $(L_0 \cap L_2^\perp) \perp L_2$. We denote the orthonormal vector passes through $\mathbf{x}^*$ and $\mathbf{x}_0$ by $\mathbf{u}_1$, one of the d orthonormal vectors that span L_2 by $\mathbf{u}_2$, and one of the $D - d - 1$ orthonormal vectors that span $(\operatorname{span}(L_2))^\perp$ by $\mathbf{u}_2$. We will prove that $\mathbf{u}_1$ is the top eigenvector of $\int_{B(\mathbf{x}^*,r^*)} (\mathbf{x} - \mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)^T \mathrm{d}\mu(\mathbf{x})$, and $\mathbf{u}_2$ is the second top eigenvector, by proving

$$\begin{aligned} &\int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)} \left(\mathbf{u}_1^T(\mathbf{x} - \mathbf{x}_0)\right)^2 \mathrm{d}\mu_2(\mathbf{x}) \\ &> \int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)} \left(\mathbf{u}_2^T(\mathbf{x} - \mathbf{x}_0)\right)^2 \mathrm{d}\mu_2(\mathbf{x}) \\ &> \int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)} \left(\mathbf{u}_3^T(\mathbf{x} - \mathbf{x}_0)\right)^2 \mathrm{d}\mu_2(\mathbf{x}). \end{aligned} \tag{21}$$

We note that

$$\begin{aligned} &\left(B\left(\mathbf{x}^*, w\right) \cap L_1^\perp\right) \times \left(B\left(\mathbf{x}^*, \sqrt{r^{*2} - w^2}\right) \cap L_1\right) \\ &\subset T(L_1, w) \cap B\left(\mathbf{x}^*, r^*\right) \\ &\subset \left(B\left(\mathbf{x}^*, w\right) \cap L_1^\perp\right) \times \left(B\left(\mathbf{x}^*, r^*\right) \cap L_1\right). \end{aligned} \tag{22}$$

Defining $\mathbf{y}$ as nearest point to $\mathbf{x}^*$ on L_2, then for $r^* > r_0 + 2w$, we have that

$$\begin{aligned} &\left(B(\mathbf{y}, w) \cap L_2^\perp\right) \times \left(B\left(\mathbf{y}, \sqrt{r^{*2} - (r_0 + 2w)^2}\right) \cap L_2\right) \\ &\subset T(L_2, w) \cap B\left(\mathbf{x}^*, r^*\right) \\ &\subset \left(B(\mathbf{y}, w) \cap L_2^\perp\right) \times \left(B\left(\mathbf{y}, \sqrt{r^{*2} - r_0^2}\right) \cap L_2\right). \end{aligned} \tag{23}$$

Moreover

$$\begin{aligned} &\operatorname{vol}\left(B\left(\mathbf{x}^*, r_1\right) \cap L^\perp\right) \times \left(B\left(\mathbf{x}^*, r_2\right) \cap L\right) \\ &= C_0(d, D - d)\, r_1^{D-d} r_2^d. \end{aligned} \tag{24}$$

Denote the center of mass of $B(\mathbf{x}^*, r^*) \cap T(L_2, w)$ by $\mathbf{x}_1$, notice that $\|\mathbf{x}_0 - \mathbf{x}^*\| < r_0 + w$, the center of mass of $B(\mathbf{x}^*, r^*) \cap T(L_1, w)$ is $\mathbf{x}^*$, and x^*, $\mathbf{x}_0$ and $\mathbf{x}_1$ satisfies

$$\mathbf{x}_0 = \frac{\operatorname{vol}(B(\mathbf{x}^*,r^*)\cap T(L_1,w))\,\mathbf{x}^* + \operatorname{vol}(B(\mathbf{x}^*,r^*)\cap T(L_2,w))\,\mathbf{x}_1}{\operatorname{vol}(B(\mathbf{x}^*,r^*)\cap T(L_1,w)) + \operatorname{vol}(B(\mathbf{x}^*,r^*)\cap T(L_2,w))}. \tag{25}$$

Combining (22), (23), (24) and (25) we have the estimation

$$\begin{aligned}
&\|\mathbf{x}_0-\mathbf{x}^*\| \\
&\quad\le \frac{\operatorname{vol}(B(\mathbf{x}^*,r^*)\cap T(L_2,w))\,(r_0+w)}{\operatorname{vol}(B(\mathbf{x}^*,r^*)\cap T(L_1,w))+\operatorname{vol}(B(\mathbf{x}^*,r^*)\cap T(L_2,w))} \\
&\quad\le \frac{(r^{*2}-r_0^2)^{\frac{d}{2}}}{(r^{*2}-w^2)^{\frac{d}{2}}+(r^{*2}-r_0^2)^{\frac{d}{2}}}\cdot(r_0+w). \qquad (26)
\end{aligned}$$

Therefore for any point $\mathbf{x}_1$ in $B(\mathbf{x}^*,r^*)\cap T(L_2,w)$, using (19) and (26) we have

$$\left|\mathbf{u}_1^T(\mathbf{x}_1-\mathbf{x}_0)\right|\ge r_0-\|\mathbf{x}_0-\mathbf{x}^*\|\ge\frac{r^*}{\sqrt{d+1}}$$

and

$$\begin{aligned}
&\int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)}\left(\mathbf{u}_1^T(\mathbf{x}-\mathbf{x}_0)\right)^2\mathrm{d}\mu_2(\mathbf{x}) \\
&\quad\ge\frac{r^{*2}}{d+1}\mu_2\bigl(B\bigl(\mathbf{x}^*,r^*\bigr)\cap T(L_2,w)\bigr). \qquad (27)
\end{aligned}$$

Since any points in $B(\mathbf{x}^*,r^*)\cap T(L_2,w)$ has a distance to $\mathbf{x}_0$ smaller than r^*, we have

$$\begin{aligned}
&\int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)}\left(\mathbf{u}_1^T(\mathbf{x}-\mathbf{x}_0)\right)^2 \\
&\qquad+d\int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)}\left(\mathbf{u}_2^T(\mathbf{x}-\mathbf{x}_0)\right)^2 \\
&\qquad\times\int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)}\|\mathbf{x}-\mathbf{x}_0\|^2\,\mathrm{d}\mu_2(\mathbf{x}) \\
&\quad<r^{*2}\mu_2\bigl(B\bigl(\mathbf{x}^*,r^*\bigr)\cap T(L_2,w)\bigr). \qquad (28)
\end{aligned}$$

Combining (27) and (28), the first inequality in (21) is proved.

By direct integration one obtains that the average of $(\mathbf{u}_2^T\mathbf{x}_1)^2$ for $\mathbf{x}_1$ in

$$\bigl(B(\mathbf{y},w)\cap L_2^{\perp}\bigr)\times\bigl(B(\mathbf{y},\sqrt{r^{*2}-(r_0+2w)^2})\bigr),$$

is $\frac{d}{d+2}(r^{*2}-(r_0+2w)^2)$, and the average of $(\mathbf{u}_2^T\mathbf{x}_1)^2$ for $\mathbf{x}_1$ in

$$T(L_2,w)\setminus\Bigl(\bigl(B(\mathbf{y},w)\cap L_2^{\perp}\bigr)\times\Bigl(B\Bigl(\mathbf{y},\sqrt{r^{*2}-(r_0+2w)^2}\Bigr)\Bigr)\Bigr)$$

is larger than that of the set

$$\bigl(B(\mathbf{y},w)\cap L_2^{\perp}\bigr)\times\Bigl(B\Bigl(\mathbf{y},\sqrt{r^{*2}-(r_0+2w)^2}\Bigr)\Bigr).$$

Applying these two facts, we obtain the estimate

$$\begin{aligned}
&\int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)}\left(\mathbf{u}_2^T(\mathbf{x}^*-\mathbf{x}_0)\right)^2\mathrm{d}\mu_2 \\
&\quad\ge\frac{d}{d+2}\bigl(r^{*2}-(r_0+2w)^2\bigr) \\
&\quad\times\mu_2\bigl(B\bigl(\mathbf{x}^*,r^*\bigr)\cap T(L_2,w)\bigr). \qquad (29)
\end{aligned}$$

We also have

$$\begin{aligned}
&\int_{B(\mathbf{x}^*,r^*)\cap T(L_2,w)}\left(\mathbf{u}_3^T(\mathbf{x}^*-\mathbf{x}_0)\right)^2\mathrm{d}\mu_2 \\
&\quad\le w^2\mu_2\bigl(B\bigl(\mathbf{x}^*,r^*\bigr)\cap T(L_2,w)\bigr). \qquad (30)
\end{aligned}$$

Using the fact that $r^*\ge r_2^*$, we have

$$\begin{aligned}
&r^{*2}-(r_0+2w)^2\ge r_2^{*2}-(r_0+2w)^2 \\
&\quad=(r_0+2w)^2\left(\frac{r_0^2}{r_0-4r_0w-4w^2}-1\right) \\
&\quad=\frac{(r_0+2w)^2}{r_0-4r_0w-4w^2}\cdot\bigl(r_0w+4w^2\bigr) \\
&\quad>4w^2. \qquad (31)
\end{aligned}$$

Combining (29), (30) and (31), the second inequality in (21) is also proved.

To estimate $\beta_2(\mathbf{x}^*,r^*)$ and $\beta_2(\mathbf{x}^*,r_0)$, using integration the points in $(B(\mathbf{x}^*,r_1)\cap L^{\perp})\times(B(\mathbf{x}^*,r_2)\cap L)$ has an average squared distance $\frac{D-d}{D-d+2}r_2^2$ to L. Besides, the points in $(B(\mathbf{y},w)\cap L_2^{\perp})\times(B(\mathbf{y},\sqrt{r^{*2}-(r_0+2w)^2})\cap L_2)$ has an average squared distance at least $(r^{*2}-(r_0+2w)^2)/3$ to the minimizer L in (20). Combining these facts with (20), (22), (23), and (24), we have

$$\begin{aligned}
&\beta_2^2(x^*,r^*) \\
&\quad>\frac{\frac{D-d}{D-d+2}w^{D-d+2}r^{*d}+w^{D-d}(r^{*2}-(r_0+2w)^2)^{\frac{d+2}{2}}/3}{r^{*2}(w^{D-d}r^{*d}+(K-1)w^{D-d}(r^{*2}-r_0^2)^{\frac{d}{2}})} \qquad (32)
\end{aligned}$$

and

$$\beta_2^2(x^*,r_0)<\frac{D-d}{D-d+2}\frac{w^2}{r_0^2}. \qquad (33)$$

To prove (5), we only need to prove that the RHS of (32) is larger than the RHS of (33), which has a following simplified form:

$$\begin{aligned}
&\frac{D-d+2}{3(D-d)}\left(1-\frac{(r_0+2w)^2}{r^{*2}}\right)^{\frac{d+2}{2}} \\
&\quad\ge(K-1)\frac{w^2}{r_0^2}\left(1-\frac{r_0^2}{r^{*2}}\right)^{\frac{d}{2}}+\frac{w^2}{r_0^2}\left(1-\frac{r_0^2}{r^{*2}}\right). \qquad (34)
\end{aligned}$$

When $d = 1$, (34) follows from

$$\left(\frac{D+1}{3(D-1)}\right)^2\left(1-\frac{(r_0+2w)^2}{r^{*2}}\right)^3 \geq K^2\frac{w^4}{r_0^4}\left(1-\frac{r_0^2}{r^{*2}}\right). \tag{35}$$

From $r^* \geq r_1^*$, we have

$$\left(1-\frac{(r_0+2w)^2}{r^{*2}}\right)^2 \geq \frac{18(D-d)^2K^2}{(D-d+2)^2}\frac{w^4}{r_0^2(r_0+2w)^2}, \tag{36}$$

and from $r^* \geq r_2^*$ we have

$$2\left(\frac{1}{(r_0+2w)^2}-\frac{1}{r^{*2}}\right) \geq \frac{1}{r_0^2}-\frac{1}{r^{*2}}. \tag{37}$$

Then (35) follows from (36) and (37), and therefore (5) is proved for the case $d = 1$.

For the case $d \geq 2$, the proof of (34) follows a similar strategy. Combing (37) and

$$\left(1-\frac{(r_0+2w)^2}{r^{*2}}\right)^{\frac{d}{2}} \geq \frac{6(D-d)K}{(D-d+2)}\frac{w^2}{(r_0+2w)^2}, \tag{38}$$

we obtain

$$\frac{D-d+2}{3(D-d)}\left(1-\frac{(r_0+2w)^2}{r^{*2}}\right)^{\frac{d+2}{2}} \geq K\frac{w^2}{r_0^2}\left(1-\frac{r_0^2}{r^{*2}}\right), \tag{39}$$

and (34) follows from (39).

Now we will prove that (4) satisfies (19). Notice that $r^{*2} - w^2 > (r_0 + 2w)^2 - w^2 > r_0^2$, it is sufficient to prove

$$\frac{r_0+w}{1+(\frac{r_0^2}{r^{*2}-r_0^2})^{\frac{d}{2}}}+\frac{r^*}{\sqrt{d+1}} \leq r_0$$

and since $r^* > r_1^* > (r_0+2w)/\sqrt{1-c} > (1+2c)r_0/\sqrt{1-c}$ and $w < c\,r_0$, where $c = 0.02$, we only need to prove

$$\frac{1+c}{1+(\frac{1}{\frac{(1+2c)^2}{1-c}-1})^{\frac{d}{2}}}+\frac{1+2c}{\sqrt{(d+1)(1-c)}} \leq 1. \tag{40}$$

It holds for $c = 0.02$ and $d = 1$. Since that when c is fixed, $d = 1$ maximizes the LHS of (40), (40) holds for any d with $c = 0.02$. Therefore (4) satisfies (19) and Theorem 1 is proved.

At last we will show that $r^* < 1.09\,r_0$. Indeed, $r_1^* < r_0(1+2\cdot 0.02)/\sqrt{1-0.02} < 1.09\,r_0$, and $r_2^* < \frac{1}{\sqrt{\frac{2}{1.04^2}-1}}r_0 < 1.09r_0$, therefore $r^* = \max(r_1^*, r_2^*) < 1.09r_0$.

Remark 1 The function $\beta_2(x, r)$ often does not have a local minimum at exactly r_0. We demonstrate it for a particular case, but it is evident that this is rather typical. Assume that $K = 2$, $d = 1$, $D = 2$ and $L_1 \perp L_2$, then for sufficiently small η, $\{B(\mathbf{x}^*, r) \cap T(L_2, w_2)\} \subset T(L_1, \beta_2(\mathbf{x}^*, r_0))$. Following the same argument for the interval $[w_{i^*}, r_0]$, $\beta_2(\mathbf{x}^*, r)$ is decreasing in the interval $[r_0, r_0 + \eta]$.

References

Arias-Castro, E., Chen, G., & Lerman, G. (2011). Spectral clustering based on local linear approximations. *Electronic Journal of Statistics*, *5*, 1537–1587 (available at arXiv:1001.1323).

Basri, R., & Jacobs, D. (2003). Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(2), 218–233.

Boult, T. E., & Brown, L. G. (1991). Factorization-based segmentation of motions. In *Proceedings of the IEEE workshop on visual motion* (pp. 179–186).

Bradley, P., & Mangasarian, O. (2000). *k*-plane clustering. *Journal of Global Optimization*, *16*(1), 23–32.

Chen, G., & Lerman, G. (2009). Foundations of a multi-way spectral clustering framework for hybrid linear modeling. *Foundations of Computational Mathematics*, *9*(5), 517–558. doi:10.1007/s10208-009-9043-7.

Chen, G., & Lerman, G. (2009). Motion segmentation by SCC on the Hopkins 155 database. In *Computer vision workshops (ICCV workshops), 2009 IEEE 12th international conference on computer vision* (pp. 759–764), Kyoto, Japan, 2009. doi:10.1109/ICCVW.5457626.

Chen, G., & Lerman, G. (2009). Spectral curvature clustering (SCC). *International Journal of Computer Vision*, *81*(3), 317–330.

Costeira, J., & Kanade, T. (1998). A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, *29*(3), 159–179.

Cox, T. F., & Cox, M. A. A. (2001). *Multidimensional scaling* (2nd ed.). London: Chapman and Hall.

David, G., & Semmes, S. (1991). Singular integrals and rectifiable sets in $\mathbb{R}^n$: au-delà des graphes Lipschitziens. *Astérisque*, *193*, 1–145.

Elhamifar, E., & Vidal, R. (2009). Sparse subspace clustering. In *Proceedings of the 2009 IEEE computer society conference on computer vision and pattern recognition (CVPR 09)* (pp. 2790–2797).

Epstein, R., Hallinan, P., & Yuille, A. (1995). 5 ± 2 eigenimages suffice: an empirical investigation of low-dimensional lighting models. In *Proceedings of the workshop on physics-based modeling in computer vision, 1995* (p. 108). doi:10.1109/PBMCV.514675.

Fukunaga, K., & Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, *20*(2), 176–183. doi:10.1109/T-C.1971.223208.

Georghiades, A., Belhumeur, P., & Kriegman, D. (2001). From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 643–660.

Goh, A., & Vidal, R. (2007). Segmenting motions of different types by unsupervised manifold clustering. In *IEEE conference on computer vision and pattern recognition, 2007, CVPR'07* (pp. 1–6). doi:10.1109/CVPR.2007.383235.

Ho, J., Yang, M., Lim, J., Lee, K., & Kriegman, D. (2003). Clustering appearances of objects under varying illumination conditions. In *Proceedings of international conference on computer vision and pattern recognition* (pp. 11–18).

Jones, P. (1990). Rectifiable sets and the traveling salesman problem. *Inventiones Mathematicae*, *102*(1), 1–15.

Kanatani, K. (2001). Motion segmentation by subspace separation and model selection. In *Proc. of 8th ICCV* (Vol. 3, pp. 586–591). Vancouver, Canada 2001.

Kanatani, K. (2002). Evaluation and selection of models for motion segmentation. In *7th ECCV* (Vol. 3, pp. 335–349).

Lauer, F., & Schnorr, C. (2009). Spectral clustering of linear subspaces for motion segmentation. In *IEEE 12th international conference on computer vision, 2009* (pp. 678–685). doi:10.1109/ICCV.2009.5459173.

Lee, K., Ho, J., & Kriegman, D. (2005). Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(5), 684–698.

Lerman, G. (2003). Quantifying curvelike structures of measures by using L_2 Jones quantities. *Communications on Pure and Applied Mathematics*, *56*(9), 1294–1365.

Lerman, G., & Zhang, T. (2010) ℓ_p-Recovery of the most significant subspace among multiple subspaces with outliers. Available at arXiv:1012.4116.

Lerman, G., & Zhang, T. (2011). Robust recovery of multiple subspaces by geometric l_p minimization. *Annals of Statistics*, *39*(5), 2686–2715. doi:10.1214/11-AOS914.

Little, A. V., Jung, Y. M., & Maggioni, M. (2009a). Multiscale estimation of intrinsic dimensionality of data sets. In *Manifold learning and its applications: papers from the AAAI fall symposium* (pp. 26–33).

Little, A. V., Lee, J., Jung, Y. M., & Maggioni, M. (2009b). Estimation of intrinsic dimensionality of samples from noisy low-dimensional manifolds in high dimensions with multiscale SVD. In *IEEE/SP 15th workshop on statistical signal processing, 2009. SSP'09* (pp. 85–88).

Ma, Y., Derksen, H., Hong, W., & Wright, J. (2007). Segmentation of multivariate mixed data via lossy coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(9), 1546–1562.

Ma, Y., Yang, A. Y., Derksen, H., & Fossum, R. (2008). Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM Review*, *50*(3), 413–458.

Ng, A., Jordan, M., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, *14*, 849–856.

Rao, S., Tron, R., Vidal, R., & Ma, Y. (2010). Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(10), 1832–1845. doi:10.1109/TPAMI.2009.191.

Soltanolkotabi, M., & Candès, E. J. (2011) A geometric analysis of subspace clustering with outliers. arXiv:1112.4258.

Sugaya, Y., & Kanatani, K. (2004). Multi-stage unsupervised learning for multi-body motion segmentation. *IEICE Transactions on Information and Systems*, *E87-D*(7), 1935–1942.

Tenenbaum, J. B., Silva, V.D., & Langford, J.C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Tipping, M., & Bishop, C. (1999). Mixtures of probabilistic principal component analysers. *Neural Computation*, *11*(2), 443–482.

Tron, R., & Vidal, R. (2007). A benchmark for the comparison of 3-d motion segmentation algorithms. In *IEEE conference on computer vision and pattern recognition, 2007, CVPR'07* (pp. 1–8). doi:10.1109/CVPR.2007.382974.

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

Vidal, R. (2011). Subspace clustering. *IEEE Signal Processing Magazine*, *28*(2), 52–68. doi:10.1109/MSP.2010.939739.

Vidal, R., Ma, Y., & Sastry, S. (2005). Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12).

Vidal, R., Tron, R., & Hartley, R. (2008). Multiframe motion segmentation with missing data using powerfactorization and GPCA. *International Journal of Computer Vision*, *79*(1), 85–105. doi:10.1007/s11263-007-0099-z.

Yan, J., & Pollefeys, M. (2006). A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In *ECCV* (Vol. 4, pp. 94–106).

Yang, A., Rao, S., & Ma, Y. (2006). Robust statistical estimation and segmentation of multiple subspaces. In *Conference on computer vision and pattern recognition workshop, 2006. CVPRW'06* (p. 99). doi:10.1109/CVPRW.2006.178.

Yang, A. Y., Rao, S. R., & Ma, Y. (2006). Robust statistical estimation and segmentation of multiple subspaces. In *CVPRW '06: proceedings of the 2006 conference on computer vision and pattern recognition workshop* (p. 99). Washington: IEEE Computer Society. doi:10.1109/CVPRW.2006.178.

Yue, S., Wang, X., & Wei, M. (2008). Application of two-order difference to gap statistic. *Transactions of Tianjin University*, *14*, 217–221. doi:10.1007/s12209-008-0039-1.

Zhang, T., Szlam, A., & Lerman, G. (2009). Median K-flats for hybrid linear modeling with many outliers. In *Computer vision workshops (ICCV workshops), 2009 IEEE 12th international conference on computer vision* (pp. 234–241). Kyoto, Japan, 2009. doi:10.1109/ICCVW.5457695.

Zhang, T., Szlam, A., Wang, Y., & Lerman, G. (2010). Randomized hybrid linear modeling by local best-fit flats. In *IEEE conference on computer vision and pattern recognition (CVPR 2010)*, pp. 1927–1934. doi:10.1109/CVPR.2010.5539866.