# HYBRID LINEAR MODELING VIA LOCAL BEST-FIT FLATS

By

**Teng Zhang**

**Arthur Szlam**
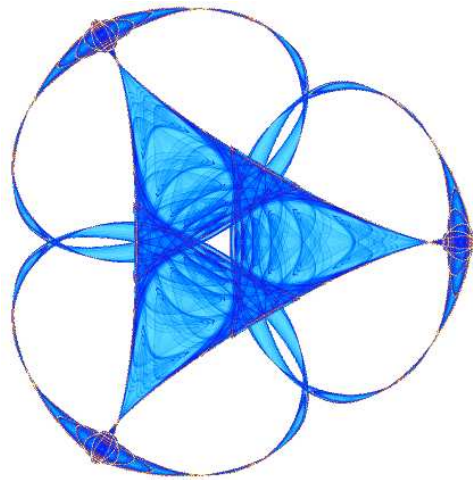
**Yi Wang**

and

**Gilad Lerman**

# INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

# Hybrid Linear Modeling via Local Best-fit Flats [*]

Teng Zhang[♮]    Arthur Szlam[♭]    Yi Wang[♮]    Gilad Lerman[♮]

[♮] School of Mathematics        [♭] Courant Institute of Mathematical Sciences
University of Minnesota                    New York University

{zhang620, wangx857, lerman}@umn.edu        aszlam@courant.nyu.edu

## Abstract

*In this paper we present a simple and fast geometric method for modeling data by a union of affine sets. The method begins by forming a collection of local best fit affine subspaces. The correct sizes of the local neighborhoods are determined automatically by the Jones' $\beta_2$ numbers; we prove under certain geometric conditions that good local neighborhoods exist and are found by our method. The collection 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/

## 1. Introduction

Several problems from computer vision, for example motion segmentation and face clustering, give rise to modeling data by multiple subspaces. This is commonly referred to as Hybrid Linear Modeling (HLM). 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 [1]). Thus clustering different mov-

ing 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) [2, 3, 4]. If we assume that images of different faces lie in different subspaces, then we can cluster these images by segmenting an arrangement of linear subspaces using HLM.

Several algorithms have been suggested for solving the HLM problem (or even solving the more general problem of clustering manifolds), for example the $K$-flats (KF) algorithm or any of its variants [5, 6, 7, 3, 8], methods based on direct matrix factorization [9, 1, 10, 11], Generalized Principal Component Analysis (GPCA) [12], Local Subspace Affinity (LSA) [13], RANSAC (for HLM) [14], Locally Linear Manifold Clustering (LLMC) [15], Agglomerative Lossy Compression (ALC) [16], Spectral Curvature Clustering (SCC) [17] and Sparse Spectral Clustering (SSC) [18].

In this paper, we describe two HLM algorithms: LBF (Local Best-fit Flat) and SLBF (spectral LBF). The first obtains state-of-the-art speed with nearly state of the art accuracy, and the second obtains state of the art accuracy (SLBF) with reasonable run times. Both LBF and SLBF are based on a straightforward geometric method for HLM, inspired by [19, 20, 21] and [22, 23, 24], that can be used in a stand alone manner or as an initialization of many of the above methods. 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. 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 contributions of this work, which extends [25], are as follows:

- We make precise the local best-fit heuristic, using the $\beta_2$

numbers [19, 20, 21]. We give an algorithm to find good neighborhoods in the above sense, and show that under certain geometric conditions, the good neighborhoods exist, and our algorithm finds them.

- Using the local best-fit heuristic we introduce the LBF and SLBF algorithms for affine clustering. At each point of a randomly chosen subset of the data, they use the best fit flats of the "correct" neighborhoods to build a global model with different methods (LBF is based on energy minimization and SLBF is a spectral method).

- We show how extensive experiments on motion segmentation data (the Hopkins 155 benchmark of [34]), face clustering (the Yale face database), handwritten digits (the MNIST dataset), and on 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).

- 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 [3] is improved by initialization that is based on the local best-fit heuristic. We also use this heuristic to estimate the main parameter of both RANSAC (for HLM) [14] and ALC [16].

- We show how the combination of LBF and the elbow method can quickly determine the number of affine subspaces.

The rest of this paper is organized as follows. In Section 2 we describe in greater depth 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

Our goal is to solve the HLM problem, that is, to partition a data set $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$ into $K$ clusters $X_1, X_2, \ldots, X_K$, with each cluster approximated by a $d$-dimensional affine subspace. In this section, we assume that all flats have the same known dimension $d$ and that the number of flats $K$ is known. The cases of unknown $K$ and mixed dimensions are addressed to some extent in

Section 3. In this section we describe two methods that have at their heart an estimation of the locally affine scale of the data.

Both methods, LBF and SLBF, break into two parts. Their first part finds a set of candidate flats, i.e., affine subspaces. It starts by randomly selecting points from the data, and then chooses a "best neighborhood" around each point (see Section 2.1). The best fit flats (in $L^2$ sense) for these neighborhoods, which we refer to as local best-fit flats, are collected as *candidates*. We remark that the number of candidates is an input parameter in LBF.

The two algorithms process the candidates in different ways: LBF uses energy minimization and SLBF uses a spectral approach. Their sketches appear in Algorithms 2 and 4 and a more detailed explanation is in Sections 2.2 and 2.3.

### 2.1. Choosing the optimal neighborhood

Choosing the correct neighborhood is crucial for the success of both proposed methods, and is in some sense the central problem of this paper. If the neighborhood is too small, even if the point is in a good affine cluster, then a small amount of noise in the data will result in a flat which does not match most of the points in the affine cluster. If the neighborhood is too large, it will contain points from more than one affine cluster, and the resulting best fit flat will again not match any of the actual data points. While it is possible to take a guess at the correct scale as a parameter, we have found that it is possible to choose the correct scale reasonably well automatically.

What we will do is start at the smallest scale (say $d + 1$) and look at larger and larger neighborhoods of a given point $\mathbf{x}_0$. At the smallest scale, any noise causes the local neighborhood to have higher dimension than $d$. As we add points to the neighborhood, it becomes better and better approximated in an average sense by its best fit flat, until points belonging to other flats enter the neighborhood. We thus take the neighborhood which is the first local minimum of the 'average errors' to the neighborhoods' best fit flats; for the 'average error', $\beta_2$, to the best fit flat within a neighborhood $\mathcal{N}$ of $\mathbf{x}_0$ we use the formula:

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

where $P_L$ denotes the projection onto the flat $L$. This notion of scaled error introduced and utilized in [19, 20, 21], and considered recently in [26] for dimension estimation. The procedure we have just described is summarized in Algorithm 1.

The following theorem tries to justify our strategy of fitting the correct scale around each point. We work with a "geometric" set of assumptions in the continuous setting,

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

**Input:** $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$: data, $\mathbf{x}$: a point in X, $S$: start size, $T$: step size, $\ell, m$ (optional): mean shifts parameters.

**Output:** $\mathcal{N}(\mathbf{x})$: a neighborhood of $\mathbf{x}$.

  **Steps**:
- (Optional) Update the point $\mathbf{x}$ as the center of its $\ell$-nearest neighborhood in X, while repeating $m$ times
- $k = -1$

**repeat**
- k:=k+1
- Let $\mathcal{N}_k$ be the set of the $S + kT$ nearest points in X to $\mathbf{x}$
- Set $\tilde{L}_k$ to be the best fit flat to $\mathcal{N}_k$
- 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 $\mathcal{N}(\mathbf{x}) := \mathcal{N}_{k-1}$

---

where our data set will be presumed to be a collection of tubes around flats. This corresponds roughly to a probabilistic setting of sampling according to mixtures of uniform distributions around subsets of $d$-flats. For convenience we assume infinite tubes but restrict to local scales.

The analog of the discrete $\beta_2$ of (1) when having an underlying continuous set $\Omega$ in a ball of center $\mathbf{x}$ and radius $r$ is defined as follows:

$$\beta_2(\mathbf{x}, r) =$$
$$\min_L \sqrt{\int_{\Omega \cap B(\mathbf{x}, r)} \left(\frac{\text{dist}(\mathbf{x}, L)}{r}\right)^2 \frac{d\mathbf{x}}{\text{vol}(\Omega \cap B(\mathbf{x}, r))}},$$

where the minimum is over all $d$-flats $L$ (see also [21]).

**Theorem 2.1.** *Let $K \geq 2$, $d < D$, $L_i$, $i = 1, \ldots, K$, be $K$ $d$-flats in $\mathbb{R}^D$, and $\Omega_i := T(L_i, w_i)$ be $K$ tubes in $\mathbb{R}^D$ around these flats of comparable widths $\{w_i\}_{i=1}^K$.*

*For fixed $1 \leq i^* \leq K$ and fixed $\mathbf{x} \in L_{i^*}$, let*

$$\mathbf{y} = \mathbf{y}(\mathbf{x}) = \text{argmin}_{\mathbf{y} \in \Omega \setminus \Omega_{i^*}} \text{dist}(\mathbf{y}, \mathbf{x}) \quad (2)$$

*and*

$$r_0 = r_0(\mathbf{x}) = \text{dist}(\mathbf{y}, \mathbf{x}). \quad (3)$$

*Assume that $r_0 > w_{i^*}$. Then the function $\beta_2(\mathbf{x}, r)$ is constant for $r$ in $[0, w_{i^*}]$, comparable to a function which is decreasing for a sufficiently large subinterval of $[w_{i^*}, r_0]$, and satisfies the inequality*

$$\beta_2(\mathbf{x}, (1 + \varepsilon) \cdot r_0) \gtrapprox \beta_2(\mathbf{x}, r_0) \quad (4)$$

*for sufficiently small $\varepsilon$, i.e., it has an "approximate" local minimum in the interval $[r_0, (1 + \varepsilon) \cdot r_0]$. If $d \leq 4$, then*

$\varepsilon \approx w_{i^*}/r_0$, *and if $d > 4$ then $\varepsilon \approx (w_{i^*}/r_0)^{4/d}$. As $w_{i^*}/r_0$ approaches zero, all comparability constants mentioned above approach one.*

## 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 of goodness of a $K$ tuple of flats $G$ is chosen; here, it will be the average $l_1$ distance of each point to its nearest flat, i.e.,

$$G = G_X(L_1, \cdots, L_K) = \sum_{\mathbf{x} \in X} \text{dist}\left(\mathbf{x}, \cup_{i=1}^K L_i\right). \quad (5)$$

After randomly initializing $K$ flats from the list of candidates, $p$ passes are made through the data points. One of the current choices of flats is removed, and all the other candidates are tried in its place. If $G$ decreases, we replace the current flat with the one which gives the lowest value for $G$. We then move to the next pass, picking a random flat, etc.

---

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

**Input:** $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$: 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 $\{X_i\}_{i=1}^K$, each approximated by flats $\{F_i\}_{i=1}^K$.

  **Steps**:
1. Pick $C$ random points in X
2. For each of the $C$ points find appropriate local scale using Algorithm 1
3. Generate $C$ candidate flats $L_1, ..., L_C$ from the best fit flats to the neighborhoods from the previous step
4. Choose $K$ flats from the candidates using Algorithm 3; collect these in $\mathcal{L}$
5. Partition X by sending points to nearest flats in $\mathcal{L}$

---

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 (5). However, in some special scenarios the $l_1$ energy of (5) is more robust to outliers than the mean squared error (see [27] for theoretical support and [8] for experimental support). One can also imagine using spectral distances that measure the smoothness of the clusters with respect to some kernel, or many other global energy functionals of a partition. The nice thing about this method is that it allows for energy functionals which may be hard to minimize; since we are only testing the energy of our candidate configurations, as long as we can compute the energy of a partition quickly, we can run the greedy descent.

**Algorithm 3** Greedy $l_1$ candidate selection for HLM by randomized local best fit flats

**Input:** $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$: data, $K$: number of flats, $L_1, ..., L_C$: candidate flats, and $p$: number of passes.

**Output:** A set of $K$ "active" flats $\mathcal{L} \subset \{L_1, ..., L_C\}$.

**Steps**:

Initialize $\mathcal{L}$ by randomly choosing $K$ "active" flats $L_{A_1}, ..., L_{A_K}$

**for** pass $= 1$ to $p$ **do**

Pick a random flat $L_{A_l} \subset \mathcal{L}$ $(1 \leq l \leq K)$

**for** $j = 1$ to $C - K$ **do**
- Pick one of the "inactive" flats $L_j$ and form the collection of flats $\tilde{\mathcal{L}} = L_j \bigcup \mathcal{L} \setminus L_{A_l}$
- Set $s_j = \sum_{i=1}^{N} \min_{L \in \tilde{\mathcal{L}}} ||x_i - P_L x_i||$

**end for**

If $\min s_j < \sum_{i=1}^{N} \min_{L \in \{L_{A_1}, ..., L_{A_K}\}} ||x_i - P_L x_i||$, set $L_{A_l} := L_{\mathrm{argmin} \, s_j}$

**end for**

## 2.3. The SLBF Algorithm

We can also process the candidate subspaces via a spectral clustering method. We first find the neighborhoods $\{\mathcal{N}_i\}_{i=1}^{N}$ for all points $\{\mathbf{x}_i\}_{i=1}^{N}$ via algorithm 1 and fit $d$-flats $\{L_i\}_{i=1}^{N}$ in these neighborhoods, then form the $N \times N$ matrices $\mathbf{S}$ and $\hat{\mathbf{S}}$ as follows:

$$\mathbf{S}_{i,j} = \sqrt{\mathrm{dist}(\mathbf{x}_i, L_j) \, \mathrm{dist}(\mathbf{x}_j, L_i)}, \qquad (6)$$

and

$$\hat{\mathbf{S}}_{i,j} = \exp(-\mathbf{S}_{i,j}/2\sigma_j^2) + \exp(-\mathbf{S}_{i,j}/2\sigma_i^2), \qquad (7)$$

where

$$\sigma_j = \sqrt{\min_{d\text{-flats L}} \sum_{\mathbf{x} \in \mathcal{N}_j} \mathrm{dist}(\mathbf{x}, L)^2}. \qquad (8)$$

Finally, we apply spectral clustering using the matrix $\hat{\mathbf{S}}$. More precisely, we follow the main algorithm of [28, Section 2], while replacing the matrix $A$ there by $\hat{\mathbf{S}}$, multiplying the unit eigenvectors of Step 3 (of [28, Section 2]) by the corresponding square roots of eigenvalues and skipping step 4. We remark that the two last changes to [28, Section 2] are common to spectral-based manifold learning algorithms (unlike spectral clustering), so that the similarity matrix $\hat{\mathbf{S}}$ is considered as a gram matrix, see e.g., Euclidean MDS [29] and ISOMAP [30]. We describe this algorithm in Algorithm 4, and refer to it as the SLBF (spectral LBF) algorithm.

**Algorithm 4** SLBF: spectral clustering based on best-fit flats

**Input:** $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$: data, $\sigma$: a parameter, other parameters used by Algorithm 1.

**Output:** A partition of X into $K$ disjoint clusters $\{X_i\}_{i=1}^{K}$, each approximated by a single flat.

**Steps**:

1. For each point $\mathbf{x}_i$, fit a subspace $L_i$ by Algorithm 1, and find $\sigma_i$ by (8)
2. Construct the $N \times N$ matrix $\mathbf{S}$ and $\hat{\mathbf{S}}$ by (6) and (7)
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 $\hat{\mathbf{S}}$ by: $\hat{\mathbf{S}} = \mathbf{D}^{-\frac{1}{2}} \hat{\mathbf{S}} \mathbf{D}^{-\frac{1}{2}}$
5. Let $U$ be the $N \times K$ matrix whose columns are the top $K$ eigenvectors of $\hat{\mathbf{S}}$, and $\Sigma$ be the $K \times K$ matrix representing the top $K$ eigenvalues of $\hat{\mathbf{S}}$
6. Implement $K$-means to the rows of $N \times K$ matrix $U \, \Sigma^{1/2}$ and partition X accordingly

## 2.4. Adaptation of the proposed algorithms to motion segmentation data

Note that the first minimum in the Theorem 2.1 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.

## 2.5. Complexity and storage of LBF and SLBF

We first discuss the complexity of Algorithm 1, which is used in both the LBF and the SLBF algorithms. 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 $k = 1, 2, \cdots, (N - S)/T$, hence the complexity for obtaining $\mathcal{N}(x)$ is of order:

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

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 $O(d \cdot D \cdot N)$.

Next, we clarify the complexity of Algorithm 2. For step 2 of this algorithm we need to run Algorithm 1 $C$ times and thus its complexity of order $O(C \cdot d \cdot D \cdot N)$. 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 4 of Algorithm 2 is an application of Algorithm 3. The later algorithm requires the evaluation of the $N \times C$ matrix representing the distances $||x_i - P_{L_j} x_i||$ between $X = \{x_1, x_2, \cdots, x_N\}$ and $L_1, L_1, \cdots, L_C$. 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 in Algorithm 3 have complexity of order $O(p \cdot (C - K) \cdot N)$. Therefore, step 4 of Algorithm 2 has a complexity of order $O(C \cdot N \cdot (d \cdot D + p))$. At last, Step 5 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 $O(C \cdot N \cdot (d \cdot D + p))$ for the LBF Algorithm.

As for the storage 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 that the algorithm also requires 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 $O(D \cdot N + C \cdot D \cdot d + N \cdot C)$.

We conclude with the complexity and storage of the SLBF algorithm. Step 1 of Algorithm 4 has a complexity of $O(d \cdot D \cdot N^2)$, since Algorithm 1 has a complexity of $O(d \cdot D \cdot N)$ and it is applied to every point in the set X. The most expensive calculation of steps 2-4 in Algorithm 4 is the construction of $\mathbf{S}$, which requires a complexity of $O(d \cdot D \cdot N^2)$. The SVD decomposition in step 5 has a complexity of $O(K \cdot N^2)$ and the $K$-means algorithm in step 6 has a complexity of $O(n_s \cdot K \cdot N \cdot D)$, where $n_s$ is the iterations in $K$-means.

Combining these complexities together, we have an overall complexity of $O((K + d \cdot D) \cdot N^2 + n_s \cdot K \cdot D \cdot N)$ for the SLBF Algorithm. Moreover, it 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 the SLBF Algorithm is in the order of $O(N \cdot D \cdot d + N^2)$.

# 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 hybrid linear modeling (HLM) algorithms.

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

$$\text{error}\% = \frac{\text{\# of misclassified inliers}}{\text{\# of total inliers}} \times 100\%. \quad (9)$$

In all the experiments below, the number $C$ in Algorithm 3 is $70 \cdot K$, where $K$ is the number of subspaces, the number $p$ in Algorithm 3 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 Sections 3.1 and 3.2 run on a computer with Intel Core 2 CPU at 2.66GHz and 2 GB memory, and experiments in Sections 3.3 and 3.4 run on a machine with Intel Core 2 Quad Q6600 at 2.4GHz and 8 GB memory.

## 3.1. Clustering Results on Simulated data

We compare our algorithm with the following algorithms: Mixtures of PPCA (MoPPCA) [5], $K$-flats (KF) [3], Local Subspace Analysis (LSA) [13], Spectral Curvature Clustering (SCC) [17], Random Sample Consensus (RANSAC) for HLM [14], Agglomerative Lossy Compression (ALC) [16] and GPCA with voting (GPCA) [31]. Throughout the rest of the paper, we use the Matlab codes of the GPCA, MoPPCA, RANSAC and KF algorithm from http://perception.csl.uiuc.edu/gpca, the SCC algorithm from http://www.math.umn.edu/~lerman/scc, the LSA algorithm from http://www.vision.jhu.edu/db, the ALC algorithm from http://perception.csl.uiuc.edu/coding/motion/ and the SSC algorithm from http://www.cis.jhu.edu/~ehsan/ssc.htm.

For the SCC algorithm, we also tried a slightly modified version tailored for motion segmentation as in step 6 of Algorithm 4, which we refer to as SCC-MS (SCC for motion segmentation): Following the notation of [Algorithm 2][17] 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 $\Sigma$ 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 $U \Sigma^{1/2}$ (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 [17]), and KF are initialized with 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 $\ell_2$ 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 $\lambda$ to be 0.01, as suggested in the code.

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

| Linear | $2^2 \in \mathbb{R}^4$ | | $4^2 \in \mathbb{R}^6$ | | $2^4 \in \mathbb{R}^4$ | | $10^2 \in \mathbb{R}^{15}$ | | $(4,5,6) \in \mathbb{R}^{10}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| Outl. % | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 |
| LSCC | 3.0 | 6.9 | 2.3 | 2.6 | 7.7 | 22.4 | 0.5 | 3.8 | 1.8 | 28.2 |
| LSCC-MS | 3.8 | 10.0 | 2.4 | 4.1 | 8.5 | 36.7 | 0.7 | 31.9 | 1.4 | 19.8 |
| LSA | 18.7 | 19.6 | 10.9 | 12.7 | 44.3 | 21.0 | 7.6 | 9.9 | 6.1 | 6.6 |
| KF | 3.0 | 15.8 | 2.5 | 18.4 | 9.4 | 34.3 | 0.8 | 33.8 | 0.8 | 30.6 |
| MoPPCA | 3.1 | 14.2 | 2.5 | 17.7 | 8.4 | 34.2 | 0.9 | 38.8 | 1.4 | 34.7 |
| GPCA | 19.7 | 30.9 | 11.7 | 35.9 | 29.2 | 43.9 | 10.2 | 42.6 | 10.1 | 45.4 |
| LBF | 2.6 | 3.7 | 2.5 | 2.3 | 6.4 | 11.5 | 1.3 | 1.9 | 1.5 | 1.5 |
| LBF-MS | 2.7 | 3.0 | 2.6 | 2.6 | 6.6 | 11.7 | 1.7 | 2.2 | 1.4 | 1.5 |
| SLBF | 5.2 | 6.3 | 5.6 | 7.0 | 18.5 | 23.9 | 5.4 | 6.2 | 2.7 | 2.4 |
| SLBF-MS | 8.7 | 11.7 | 5.9 | 6.6 | 33.5 | 46.6 | 3.9 | 4.8 | 2.4 | 2.6 |
| RANSAC (oracle) | 3.3 | 2.6 | 2.3 | 2.2 | 8.6 | 9.8 | 0.9 | 6.7 | 1.8 | 1.4 |
| RANSAC ($\epsilon$ from LBF) | 2.4 | 2.8 | 2.2 | 2.5 | 5.8 | 7.5 | 30.4 | 42.8 | 0.7 | 13.5 |
| ALC (oracle) | 4.0 | 3.4 | 13.1 | 16.3 | 27.5 | 30.1 | 50.0 | 50.0 | 5.3 | 36.1 |
| ALC ($\epsilon$ from LBF) | 4.1 | 5.7 | 8.7 | 10.0 | 9.9 | 14.0 | 50.0 | 50.0 | 2.3 | 1.8 |
| SSC | 24.8 | 34.3 | 32.2 | 43.5 | 49.2 | 52.8 | 18.9 | 44.9 | 32.4 | 54.0 |

| Affine | $2^2 \in \mathbb{R}^4$ | | $4^2 \in \mathbb{R}^6$ | | $2^4 \in \mathbb{R}^4$ | | $10^2 \in \mathbb{R}^{15}$ | | $(4,5,6) \in \mathbb{R}^{10}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| Outl. % | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 |
| SCC | 0.0 | 0.6 | 0.0 | 0.0 | 0.2 | 0.5 | 0.0 | 0.7 | 0.0 | 5.8 |
| SCC-MS | 0.0 | 2.2 | 0.0 | 0.5 | 1.4 | 5.8 | 0.0 | 0.0 | 0.0 | 3.1 |
| LSA | 11.8 | 11.0 | 5.3 | 4.7 | 45.0 | 41.7 | 0.0 | 0.0 | 1.0 | 1.1 |
| KF | 7.3 | 15.1 | 9.9 | 26.0 | 19.7 | 37.1 | 11.1 | 24.9 | 7.3 | 23.5 |
| MoPPCA | 25.6 | 23.7 | 27.8 | 38.3 | 45.5 | 39.8 | 37.1 | 45.2 | 42.9 | 46.8 |
| GPCA | 13.8 | 14.4 | 22.6 | 22.1 | 33.6 | 32.4 | 36.0 | 29.6 | 26.7 | 29.1 |
| LBF | 0.2 | 2.0 | 0.0 | 0.7 | 0.3 | 4.5 | 0.0 | 0.3 | 0.0 | 0.0 |
| LBF-MS | 0.2 | 2.7 | 0.1 | 1.5 | 0.8 | 5.2 | 0.0 | 0.5 | 0.0 | 0.0 |
| SLBF | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| SLBF-MS | 0.0 | 0.1 | 0.0 | 0.0 | 0.2 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| RANSAC (oracle) | 13.2 | 12.2 | 11.5 | 11.2 | 31.5 | 28.4 | 2.6 | 9.2 | 1.1 | 2.2 |
| RANSAC ($\epsilon$ from LBF) | 6.3 | 41.2 | 10.1 | 47.9 | 10.5 | 45.8 | 40.7 | N/A | N/A | 61.8 |
| ALC (oracle) | 0.0 | 0.0 | 0.0 | 0.0 | 35.9 | 25.1 | 0.0 | 40.0 | 0.0 | 65.0 |
| ALC ($\epsilon$ from LBF) | 0.7 | 0.4 | 0.0 | 0.0 | 0.7 | 0.3 | 0.0 | 0.0 | 0.0 | 0.0 |
| SSC | 1.1 | 1.9 | 0.1 | 0.1 | 6.6 | 6.4 | 0.0 | 0.0 | 0.0 | 0.0 |

Table 2. Mean running time for linear-subspaces cases and affine-subspaces cases.

| Linear | $2^2 \in \mathbb{R}^4$ | | $4^2 \in \mathbb{R}^6$ | | $2^4 \in \mathbb{R}^4$ | | $10^2 \in \mathbb{R}^{15}$ | | $(4,5,6) \in \mathbb{R}^{10}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| Outl. % | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 |
| LSCC | 0.7 | 0.8 | 16.0 | 1.8 | 2.1 | 2.0 | 13.3 | 5.7 | 5.1 | 8.4 |
| LSCC-MS | 0.5 | 0.5 | 1.2 | 1.4 | 1.7 | 1.5 | 5.1 | 5.6 | 4.0 | 4.6 |
| LSA | 8.8 | 16.0 | 11.1 | 20.8 | 28.3 | 54.4 | 31.3 | 31.5 | 38.2 | 54.4 |
| KF | 0.5 | 0.6 | 0.5 | 0.8 | 1.4 | 1.8 | 1.9 | 1.0 | 1.1 | 2.8 |
| MoPPCA | 0.2 | 0.5 | 0.3 | 0.7 | 1.2 | 2.0 | 1.7 | 1.1 | 1.0 | 3.3 |
| GPCA | 3.5 | 7.6 | 9.8 | 19.0 | 20.9 | 29.7 | 30.3 | 31.6 | 39.1 | 57.8 |
| LBF | 0.5 | 0.5 | 0.5 | 0.5 | 2.2 | 2.7 | 0.7 | 0.8 | 1.2 | 1.4 |
| LBF-MS | 0.4 | 0.5 | 0.4 | 0.5 | 2.0 | 2.6 | 0.5 | 0.6 | 1.0 | 1.3 |
| SLBF | 10.5 | 20.7 | 11.8 | 21.7 | 90.1 | 174.9 | 12.0 | 23.3 | 31.3 | 64.2 |
| SLBF-MS | 13.2 | 24.0 | 13.1 | 24.4 | 152.0 | 202.0 | 13.2 | 23.5 | 39.5 | 72.4 |
| RANSAC (oracle) | 8.0 | 9.5 | 15.7 | 16.1 | 14.8 | 18.3 | 44.4 | 46.4 | 104.0 | 103.7 |
| RANSAC ($\epsilon$ from LBF) | 9.2 | 11.4 | 17.9 | 19.0 | 19.3 | 23.8 | 48.7 | 44.9 | 114.3 | 141.6 |
| ALC (oracle) | 12.0 | 23.2 | 16.2 | 33.6 | 68.7 | 136.3 | 37.7 | 172.6 | 53.1 | 180.1 |
| ALC ($\epsilon$ from LBF) | 18.9 | 28.0 | 19.6 | 37.9 | 59.2 | 121.9 | 73.1 | 152.4 | 79.7 | 151.6 |
| SSC | 162.8 | 236.2 | 170.8 | 247.9 | 382.7 | 591.3 | 184.1 | 276.6 | 298.3 | 437.9 |

| Affine | $2^2 \in \mathbb{R}^4$ | | $4^2 \in \mathbb{R}^6$ | | $2^4 \in \mathbb{R}^4$ | | $10^2 \in \mathbb{R}^{15}$ | | $(4,5,6) \in \mathbb{R}^{10}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| Outl. % | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 | 5 | 30 |
| SCC | 0.9 | 1.0 | 1.7 | 2.0 | 5.1 | 2.5 | 6.1 | 13.7 | 5.6 | 6.0 |
| SCC-MS | 0.7 | 0.7 | 1.4 | 1.6 | 2.2 | 2.2 | 5.4 | 6.0 | 4.6 | 4.8 |
| LSA | 8.7 | 16.1 | 11.1 | 20.8 | 28.6 | 54.0 | 21.1 | 32.2 | 38.3 | 54.0 |
| KF | 0.5 | 0.6 | 0.6 | 0.7 | 2.4 | 1.4 | 0.6 | 1.7 | 1 | 1.4 |
| MoPPCA | 0.5 | 0.5 | 0.7 | 0.6 | 2.9 | 1.4 | 1.3 | 1.9 | 1.9 | 2.0 |
| GPCA | 2.4 | 6.9 | 5.1 | 9.8 | 11.2 | 26.1 | 20.2 | 31.9 | 38.4 | 49.9 |
| LBF | 0.5 | 0.6 | 0.5 | 0.6 | 2.2 | 2.8 | 0.7 | 0.8 | 1.2 | 1.5 |
| LBF-MS | 0.4 | 0.5 | 0.4 | 0.5 | 2.0 | 2.7 | 0.5 | 0.6 | 1.0 | 1.3 |
| SLBF | 9.4 | 19.1 | 8.8 | 19.0 | 71.8 | 143.1 | 9.2 | 19.4 | 35.1 | 61.4 |
| SLBF-MS | 10.5 | 21.7 | 10.1 | 21.9 | 79.9 | 175.5 | 10.4 | 21.1 | 40.1 | 66.7 |
| RANSAC (oracle) | 12.0 | 14.4 | 19.6 | 20.6 | 23.9 | 29.5 | 48.2 | 51.5 | 79.7 | 84.9 |
| RANSAC ($\epsilon$ from LBF) | 13.4 | 12.6 | 22.7 | 23.5 | 28.4 | 32.1 | 49.3 | N/A | N/A | 60.6 |
| ALC (oracle) | 13.3 | 25.2 | 15.2 | 39.1 | 61.0 | 119.3 | 18.5 | 43.0 | 39.7 | 92.7 |
| ALC ($\epsilon$ from LBF) | 13.4 | 26.8 | 15.6 | 29.8 | 55.2 | 113.6 | 29.8 | 55.5 | 47.9 | 85.2 |
| SSC | 160.2 | 226.8 | 176.0 | 255.3 | 386.6 | 592.4 | 202.9 | 311.9 | 338.6 | 504.1 |

The RANSAC and ALC algorithms for HLM [14, 16] depends on a user supplied inlier threshold. RANSAC (oracle) and ALC (oracle) uses the oracle inlier bound given by the true noise variance of the model and thus clearly has an advantage over the other algorithms listed. RANSAC ($\epsilon$ from LBF) and ALC ($\epsilon$ from LBF) estimates the inlier threshold by the local best fit flats heuristic of this paper. That is, it fits $N$ neighborhoods for all $N$ points using this heuristic and estimates the least error of approximation by $d$-flat in these $N$ neighborhoods. The inlier bound $\epsilon$ is then the average of these errors. For some cases this algorithm run into error, then the result is reported as N/A. The reason for this is that the RANSAC algorithm for HLM [14] is very sensitive to the estimate of $\varepsilon$ 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. ALC ($\epsilon$ from LBF) may result in a number of clusters different than $K$, though we still use the same identification error as in (9), while comparing the true label with all permutations of the computed label and use the one with smallest error. We remark that due to outliers we could not effectively use the voting procedure for ALC described later.

The simulated data represents various instances of $K$ linear subspaces in $\mathbb{R}^D$. If their dimensions are fixed and equal $d$, we follow [17] and refer to the setting as $d^K \in \mathbb{R}^D$. If they are mixed, then we follow [31] and refer to the setting as $(d_1, \ldots, d_K) \in \mathbb{R}^D$. Fixing $K$ and $d$ (or $d_1, \ldots, d_K$), 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 of the various algorithms is recorded in Table 1. The mean running time is shown in Table 2. 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 $\ell_1$ loss function (see e.g., [27]) 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 supports affine subspace models (the particular data has non-intersecting subspaces, which makes advantageous to some other algorithms, e.g., SSC). The results of RANSAC ($\epsilon$ from LBF) and ALC ($\epsilon$ 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 2 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 is still faster than SSC, which also depends on a spectral matrix.

## 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 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.

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 $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N \in \mathbb{R}^3$ 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 $i^{\text{th}}$ image frame for every $1 \leq i \leq F$ and $1 \leq j \leq N$. Then $\mathbf{z}_j = [\mathbf{z}_{1j}, \mathbf{z}_{2j}, \cdots, \mathbf{z}_{Fj}] \in \mathbb{R}^{2F}$ is the trajectory of the

Table 3. The mean and median percentage of misclassified points for two-motions and three-motions in Hopkins 155 database.

| 2-motion | Checker | | Traffic | | Articulated | | All | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Median | Mean | Median | Mean | Median | Mean | Median |
| 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 | 0.0 |
| 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 | Checker | | Traffic | | Articulated | | All | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Median | Mean | Median | Mean | Median | Mean | Median |
| 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 |

Table 4. The standard deviation to the mean and median percentage of misclassified points for two-motions and three-motions in Hopkins 155 database.

| 2-motion | Checker | | Traffic | | Articulated | | All | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Median | Mean | Median | Mean | Median | Mean | Median |
| 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 | Checker | | Traffic | | Articulated | | All | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Median | Mean | Median | Mean | Median | Mean | Median |
| 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 |

$j^{th}$ feature point across the $F$ frames. The actual task of motion segmentation is to separate these trajectory vectors $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N$ into $K$ clusters representing the $K$ underlying motions.

It has been shown [1] 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 $d = 3$.

We compare our algorithm with the following ones: improved GPCA for motion segmentation (GPCA) [32], $K$-flats (KF) [3] (implemented for linear subspaces), Local Linear Manifold Clustering (LLMC) [15], Local Subspace Analysis (LSA) [13], Multi Stage Learning (MSL) [33], Spectral Curvature Clustering (SCC) [17] and SCC-MS (see description earlier), Sparse Subspace Clustering (SSC) [18], and RANSAC for HLM [14].

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 [34] and [15]). 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

Table 5. Average total computation times for all 155 sequences.

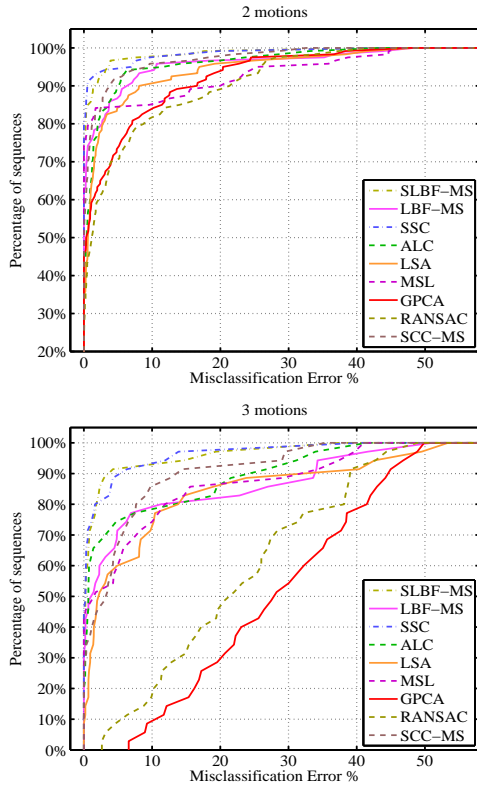| RANSAC | LBF-MS ($4K$,3) | LBF ($4K$,3) | SCC-MS($4K$,3) | SLBF-MS($2F$,3) | SLBF($2F$,3) | SSC-N($4K$,3) |
|--------|------------------|--------------|-----------------|------------------|---------------|----------------|
| 60s | 73s | 91s | 196s | 28min | 31min | 427min |



Figure 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.*

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 6 and 7 and demonstrated in Figure 1.

Our misclassification errors for SCC are different than [35] and [36] and our misclassification errors for SSC are different than [18] (the difference between our and their results differ more than twice the standard deviations of errors obtained here). This can be explained by possible evolutions of the codes since then (at least for SSC). We remark though that the misclassification errors of SCC-MS here are even slightly better than the misclassification errors of SCC in [35].

From Table 3 and Figure 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 algo-

rithms. Besides SLBF/SLBF-MS and SSC-N, only SSC-MS is better than LBF-MS. However, From Table 5, 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 $\ell_1$ energy (as well as the $\ell_2$ 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 errors 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 insist using the same parameters on all four kinds of data (see the third paragraph of Section 3).

From Table 4 we can see that SLBF-MS, SLBF and SSC-N have almost negligible standard deviations in Hopkins 155 database. LBF and LBF-MS are more random, but still have comparable standard deviations with other good algorithms such as SCC/SCC-MS.

### 3.3. Clustering results on Yale Faces B data set

We also test the proposed methods LBF, LBF-MS, SLBF and SLBF-MS and compare them with ALC, SCC, SSC and GPCA on Yale Faces B which is available on http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html. We use 9 subsets of the Yale Face Database B consisting of face images of $2, \ldots, 10$ persons respectively under 64 varying lighting conditions. For computational efficiency, we follow [17] to downsample each image to $120 \times 160$ pixels, so the dimension of the image space is $D' = 120 \times 160$. The 9 subsets contain the faces with the following indices: [5, 8], [1, 5, 8], [1, 5, 8, 10], [1, 4, 5, 8, 10], [1, 2, 4, 5, 8, 10], [1, 2, 4, 5, 7, 8, 10], [ 1, 2, 4, 5, 7, 8, 9, 10], [1, 2, 3, 4, 5, 7, 8, 9, 10] and [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. We apply PCA to the data matrix to reduce the dimension to $D = 20$ for all methods except GPCA. In GPCA, we reduce the dimension to 5 to save running time (it often happened that GPCA got stuck with $D = 20$). We also add another coordinate whose value is 1 to each point so that GPCA is applicable to affine subspaces. By applying [37] to all data pointed we estimated that $d = 2$ (i.e., the local dimension around most data points was 2), though almost any other method for dimension estimation will give the same result since there is hardly any noise in

face data. We thus input $d = 2$ and the correct number of clusters for all algorithms except ALC. ALC (voting with $K$) exactly follows [38, sec. (2.2)], in particular, it chooses $\varepsilon$ from 101 values in the range $10^{-5} - 10^3$ (see the code in http://perception.csl.uiuc.edu/coding/motion/#Software).

We see from Table 6 that our algorithms are very accurate in this data set. SLBF and SLBF-MS are the best algorithms besides ALC ($\epsilon$ from LBF), and LBF and LBF-MS are also great considering the running time. Besides, this data set proves that the local best-fit heuristic provides a good estimation for the noise: ALC ($\epsilon$ from LBF) obtains better results than ALC (voting with $K$) with perfect classification and a much smaller running time.

### 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 $28 \times 28$ 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 Section 3.3. We run 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 8, 10, 9 and 11.

From Table 8 and 9, SLBF and SLBF-MS are the best algorithms among all the methods in terms of classification error, although the misclassification rate increases 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.

### 3.5. Automatic determination of the number of affine sets

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 $F_j$, $j = 1, \ldots, K$ 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,

$$W_K = \sum_{j=1}^{K} \sum_{\mathbf{x} \in C_j} \text{dist}^2(\mathbf{x}, F_j). \quad (10)$$

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 $W_K$ [39]:

$$\text{SOD}(\ln W_K) = \ln W_{K-1} + \ln W_{K+1} - 2 \ln W_K. \quad (11)$$

The optimal $K$ is thus found by

$$K_{\text{SOD}} = \arg \max_K \text{SOD}(\ln W_K), . \quad (12)$$

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

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 [16] and part of GPCA [12] on a number of artificial data sets and real data sets. These experiments run on a machine with Intel Core 2 Quad Q6600 at 2.4GHz and 8 GB memory.

#### 3.5.1   Finding the number of clusters on artificial data

We test the above use of our algorithms for detecting the number of clusters while comparing it with some other methods (three variations of ALC, number of clusters by GPCA and SOD with SSC and SCC). In these experiments, data sets are generated by the Matlab code borrowed from the GPCA [12] 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 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 $\epsilon$ (as in [38]) and choose the estimated $K$ by majority. In ALC ($\epsilon$ from LBF), we choose the average noise in the neighborhood using the local best-fit heuristic as the distortion rate $\epsilon$. In ALC (oracle), we input the true noise level ($\epsilon = 0.05$) as the distortion rate. For GPCA, we use the original idea of [12, 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 [31, 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

Table 6. Mean percentage of misclassified points and mean running time on clustering Yale Faces B data set.

| $K$ | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| LBF | $e\%$ | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.4 | 0.8 | 0.5 |
| | $t(s)$ | 0.60 | 0.89 | 0.83 | 1.40 | 1.47 | 1.96 | 2.54 | 3.19 | 4.00 |
| LBF-MS | $e\%$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.8 | 0.0 | 0.8 | 1.1 | 1.0 |
| | $t(s)$ | 0.32 | 0.48 | 0.54 | 0.90 | 1.04 | 1.47 | 1.98 | 2.56 | 3.26 |
| SLBF | $e\%$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.2 | 0.9 |
| | $t(s)$ | 3.70 | 7.90 | 14.00 | 28.32 | 43.50 | 63.79 | 118.99 | 179.70 | 249.42 |
| SLBF-MS | $e\%$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 | 0.3 |
| | $t(s)$ | 3.26 | 7.26 | 13.45 | 24.09 | 39.85 | 62.57 | 102.13 | 148.99 | 203.37 |
| ALC (voting with $K$) | $e\%$ | 50.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | $t(s)$ | 126.31 | 306.63 | 564.27 | 880.38 | 1251.14 | 1719.52 | 2274.04 | 3043.32 | 4360.81 |
| ALC ($\epsilon$ from LBF) | $e\%$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | $t(s)$ | 3.25 | 7.13 | 12.81 | 20.45 | 29.28 | 40.08 | 52.49 | 66.58 | 82.63 |
| SCC | $e\%$ | 0.0 | 0.0 | 0.0 | 1.1 | 2.7 | 2.1 | 2.2 | 5.7 | 6.6 |
| | $t(s)$ | 0.57 | 0.92 | 1.45 | 2.30 | 2.27 | 4.57 | 6.58 | 10.29 | 7.51 |
| SCC-MS | $e\%$ | 0.0 | 0.0 | 2.9 | 4.1 | 7.5 | 16.7 | 23.0 | 28.2 | 32.5 |
| | $t(s)$ | 0.52 | 0.62 | 0.93 | 1.36 | 1.86 | 2.52 | 3.31 | 4.23 | 5.37 |
| GPCA | $e\%$ | 0.0 | 49.5 | 0.0 | 26.6 | 9.9 | 25.2 | 28.5 | 30.6 | 19.8 |
| | $t(s)$ | 1.42 | 2.72 | 4.91 | 8.08 | 11.71 | 33.11 | 99.49 | 286.36 | 1122.50 |
| K-flats | $e\%$ | 30.4 | 57.4 | 38.1 | 35.6 | 36.4 | 38.2 | 40.7 | 40.5 | 41.2 |
| | $t(s)$ | 0.44 | 0.61 | 0.78 | 1.00 | 1.42 | 1.83 | 2.73 | 2.91 | 3.42 |
| SSC | $e\%$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.4 | 4.6 |
| | $t(s)$ | 36.56 | 56.21 | 80.87 | 107.82 | 137.83 | 174.81 | 219.22 | 276.81 | 570.57 |

Table 7. The standard deviation to the mean percentage of misclassified points on clustering Yale Faces B data set.

| Real $K$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| LBF | 0.0 | 0.2 | 0.0 | 1.3 | 0.3 | 0.1 | 0.1 | 1.2 | 0.4 |
| LBF-MS | 0.0 | 0.1 | 0.0 | 0.2 | 0.2 | 0.0 | 0.1 | 0.1 | 0.2 |
| SLBF | 0.0 | 0.0 | 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 | 0.0 | 0.0 |
| ALC(voting with $K$) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ALC($\epsilon$ from LBF) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| SCC | 0.0 | 0.0 | 0.0 | 0.5 | 1.8 | 3.5 | 3.6 | 4.4 | 5.5 |
| SCC-MS | 0.0 | 0.0 | 9.4 | 8.5 | 7.8 | 9.4 | 7.2 | 7.3 | 6.6 |
| GPCA | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| K-flats | 1.0 | 7.2 | 9.4 | 8.4 | 8.3 | 7.4 | 6.7 | 5.9 | 5.7 |
| SSC | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.2 | 6.0 |

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 12.

As in Table 12, ALC (oracle) and ALC ($\epsilon$ from LBF) work the best for low dimensions ($d = 1, 2, 3$), but in real problems this choice (the noise level) for $\epsilon$ 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.

### 3.5.2 Finding the number of clusters on Yale Faces B data set

We use the Yale Faces B as in Section 3.3 for testing the above algorithms for detecting the number of clusters. The ambient dimension is reduced to $D = 20$ by PCA for all of the methods and the intrinsic dimension of subspaces is set as $d = 2$ (see Section 3.3). For SOD with different clustering algorithms, we let $K_{\max} = 6, 8, 8, 10, 10, 12, 14, 16$ and 18 respectively for 2 to 10 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). The error rate of finding the correct number of clusters and the computation time are recorded in Table 13.

We see from Table 13 that SOD (SLBF), SOD (SLBF-MS), ALC ($\epsilon$ from LBF) and ALC (voting) have perfect detection of $K$; SOD (SLBF) and SOD (SLBF-MS) are faster than ALC (voting) on data with small sizes, but slower with large sizes; LBF provides a good estimation for the distortion rate and makes ALC ($\epsilon$ from LBF) much faster than ALC (voting); SOD (LBF) and SOD (SLBF) have nearly perfect detection and obvious advantage of efficiency; GPCA does not work well on finding the number of clusters.

### 3.5.3 Finding the number of clusters on MNIST data set

We preprocess MNIST exactly the same way as we did in Section 3.4. The ambient dimension is reduced to both

Table 8. 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 ($\epsilon$ 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 |

Table 9. 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 | 2.3 | 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 ($\epsilon$ 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 |

$D = 10$ 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 rate of finding the correct number of clusters and the computation time are recorded in Tables 14 and 15.

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 14 that when we project data to 10-dimensional space, ALC and GPCA fail in most cases, except for ALC ($\epsilon$ from LBF) on digits [3 6 8]. SOD

(SLBF), SOD (SLBF-MS) and SOD (SSC) outperform all others although they are not very efficient.

### 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

Table 10. 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($\epsilon$ 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 11. 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($\epsilon$ 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 |

have $K$ flats; we use these as an initialization for $K$-flats.

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 $x, y, z+.2$ plane, and then 500 more from the $x, y, z+.4$ plane. This data set is designed to favor the use of small neighborhoods. The next data set is three random affine sets with 15% Gaussian noise and 5% outliers, generated using the Matlab code from GPCA, as in Section 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 Figure 2. The error rates of $K$-flats with farthest insertion initialization with fixed neighborhoods of size 10, 20, ..., 160 are plotted against the error rate for farthest insertion with adapted neighborhoods (searched over the same range), averaged over 400 runs in Figure 3. 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 Figure 2 we plot the number of neighbors picked by our algorithm for each point of a realization of data set #3.

## 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 $\ell_2$ $\beta$ numbers; it is proven under certain geometric conditions that good local neighborhoods exist and are found
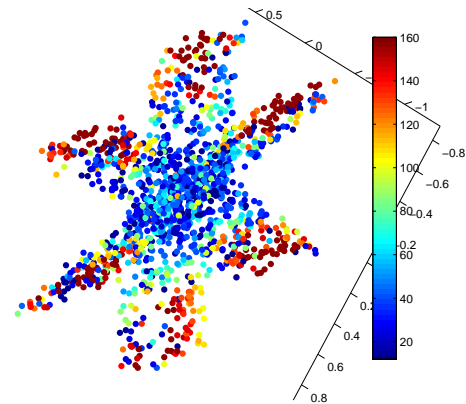


Figure 2. *Color map of neighborhood size obtained by the local best-fit flat heuristic. The color value represents the number of neighbors chosen at that point. Note that the algorithm chooses smaller neighborhoods for points closer to the intersection of the planes.*

by this method. 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 the next step is to adapt the method for multi-manifold clustering. As it is, our method, while quite good at unions of affine sets, cannot successfully handle unions of curved manifolds. We believe 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.

Table 12. 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 $= \pi/8$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $1^6 \in \mathbb{R}^5$ | $2^4 \in \mathbb{R}^5$ | $3^3 \in \mathbb{R}^5$ | $1^6 \in \mathbb{R}^3$ | $2^4 \in \mathbb{R}^4$ | $3^3 \in \mathbb{R}^4$ | $10^2 \in \mathbb{R}^{15}$ | $1^6 \in \mathbb{R}^3$ | $2^4 \in \mathbb{R}^3$ | $3^3 \in \mathbb{R}^4$ | $10^2 \in \mathbb{R}^{15}$ |
| 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 ($\epsilon$ 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 | 0 | 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 | $e\%$ | 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 |

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 Yale Faces B data set.

| Real $K$ | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| SOD (LBF) | $e\%(K)$ | 0.0 | 0.2 | 0.0 | 0.4 | 0.4 | 0.2 | 0.4 | 2.2 | 5.6 |
| | $t(s)$ | 1.22 | 2.38 | 3.19 | 5.32 | 6.41 | 9.92 | 15.25 | 22.17 | 31.19 |
| SOD (LBF-MS) | $e\%(K)$ | 0.0 | 0.2 | 0.2 | 0.0 | 0.2 | 0.2 | 0.4 | 2.6 | 1.0 |
| | $t(s)$ | 0.73 | 1.65 | 2.21 | 3.77 | 4.53 | 7.46 | 11.86 | 18.10 | 25.83 |
| SOD (SLBF) | $e\%(K)$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | $t(s)$ | 69.94 | 167.00 | 160.86 | 364.18 | 408.30 | 772.41 | 1527.42 | 2395.91 | 3818.62 |
| SOD (SLBF-MS) | $e\%(K)$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | $t(s)$ | 69.92 | 167.65 | 160.96 | 364.37 | 409.56 | 777.07 | 1526.85 | 2396.82 | 3826.75 |
| ALC (voting) | $e\%(K)$ | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | $t(s)$ | 127.85 | 305.82 | 567.46 | 902.63 | 1278.08 | 1744.64 | 2284.50 | 2945.04 | 3718.86 |
| ALC ($\epsilon$ from LBF) | $e\%(K)$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | $t(s)$ | 3.25 | 7.13 | 12.81 | 20.45 | 29.28 | 40.08 | 52.49 | 66.58 | 82.63 |
| GPCA | $e\%(K)$ | 0.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| | $t(s)$ | 0.07 | 0.13 | 0.52 | 0.71 | 1.02 | 1.56 | 2.70 | 2.65 | 3.38 |
| SOD (SCC) | $e\%(K)$ | 0.0 | 0.0 | 0.0 | 3.4 | 30.8 | 6.4 | 15.2 | 47.8 | 52.0 |
| | $t(s)$ | 5.51 | 13.27 | 11.02 | 24.64 | 25.12 | 53.43 | 85.35 | 125.24 | 188.25 |
| SOD (SCC-MS) | $e\%(K)$ | 0.0 | 0.0 | 9.4 | 31.0 | 46.6 | 79.0 | 84.4 | 90.0 | 94.6 |
| | $t(s)$ | 4.93 | 10.54 | 10.87 | 20.14 | 21.26 | 34.66 | 54.05 | 83.12 | 120.95 |
| SOD (SSC) | $e\%(K)$ | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 30.0 | 50.0 |
| | $t(s)$ | 163.26 | 355.51 | 511.81 | 868.86 | 1100.86 | 1588.92 | 2524.68 | 3263.78 | 4340.78 |

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=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 | 248.2 | 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 ($\epsilon$ 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 |

Table 15. 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 | $e\%$ | 45.0 | 35.0 | 54.0 | 79.0 | 72.0 | 67.0 | 60.0 |
| (LBF) | $t(s)$ | 22.9 | 23.5 | 22.2 | 24.9 | 56.2 | 54.6 | 51.1 |
| SOD | $e\%$ | 32.0 | 22.0 | 38.0 | 66.0 | 44.0 | 82.0 | 58.0 |
| (LBF-MS) | $t(s)$ | 12.2 | 12.2 | 12.2 | 12.2 | 29.3 | 29.4 | 29.4 |
| SOD | $e\%$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 | 100.0 |
| (SLBF) | $t(s)$ | 204.2 | 198.1 | 207.8 | 295.8 | 864.5 | 766.5 | 706.1 |
| SOD | $e\%$ | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 | 100.0 | 100.0 |
| (SLBF-MS) | $t(s)$ | 213.7 | 201.7 | 176.6 | 259.9 | 748.1 | 640.0 | 681.1 |
| ALC | $e\%$ | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| (voting) | $t(s)$ | 1469.2 | 1445.6 | 1489.2 | 679.0 | 1530.1 | 1528.5 | 3032.4 |
| ALC | $e\%$ | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| ($\epsilon$ from LBF) | $t(s)$ | 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 | $e\%(K)$ | 0.0 | 4.0 | 1.0 | 50.5 | 78.8 | 30.3 | 83.8 |
| (SCC) | $t(s)$ | 14.9 | 10.6 | 11.6 | 11.6 | 24.7 | 26.2 | 25.4 |
| SOD | $e\%(K)$ | 0.0 | 0.0 | 0.0 | 42.4 | 89.9 | 97.0 | 93.9 |
| (SCC-MS) | $t(s)$ | 12.6 | 13.0 | 14.7 | 13.9 | 34.0 | 36.8 | 30.7 |
| SOD | $e\%(K)$ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 | 100.0 |
| (SSC) | $t(s)$ | 426.4 | 417.6 | 409.3 | 413.5 | 823.8 | 821.2 | 836.8 |
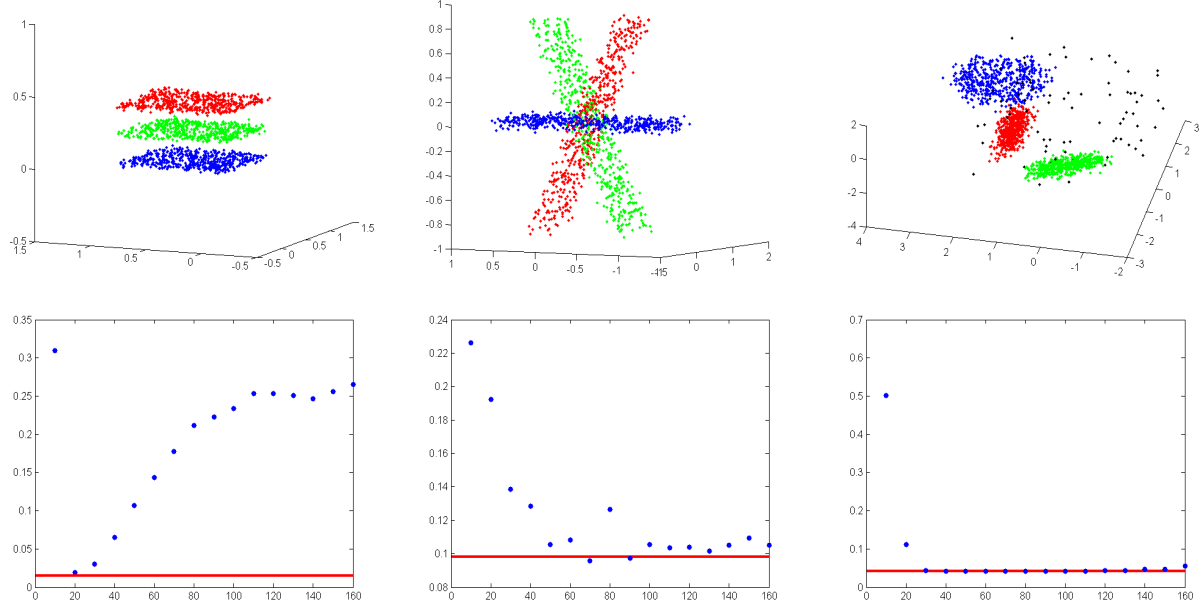


Figure 3. *Using our neighborhood choice to improve initialization of k-flats: the first row is the visualization of three data sets, and the seconds row shows the corresponding figures such that the vertical axis is accuracy, and the horizontal axis is fixed neighborhood size in geometric farthest insertion 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 Section 3.6. Random initialization leads to errors of .4 or greater for all three data sets.*

## A. Proof of Theorem 2.1

*Proof.* In order to demonstrate the basic ideas about the proof, we also use an $L_\infty$ version of the continuous $\beta_2$, which is formed as follows:

$$\beta_\infty(\mathbf{x}, r) = \min_{d-\text{flats } L} \max_{\mathbf{y} \in \Omega \cap B(\mathbf{x},r)} \frac{\text{dist}(\mathbf{x}, L)}{r} .$$

Clearly, the function $\beta_\infty$ is not robust to noise, unlike $\beta_2$.

However, direct computations are easier with $\beta_\infty$ and there is no need in the various comparability constants, which are 1 in this case.

Assume for simplicity that $w_1 = \cdots = w_K = w$ and also assume without loss of generality that $i^* = 1$. Denote the underlying set by $\Omega$, i.e., $\Omega = \cup_{i=1}^K \Omega_i$. Note that if $r \leq w$, then $B(\mathbf{x}, r)$ is full, i.e., $B(\mathbf{x}, r) \setminus \Omega_1 = \emptyset$. By scaling invariance we immediately obtain that $\beta_\infty(\mathbf{x}, r) =$

$\beta_\infty(\mathbf{x}, w) = 1/2$ and $\beta_2(\mathbf{x}, r) = \beta_2(\mathbf{x}, w)$ for all $r \leq w$.

On the other hand, if $w \leq r \leq r_0$, then

$$\beta_\infty(\mathbf{x}, r) = \frac{w}{r}$$

and is decreasing. Clearly, $\beta_\infty$ has a local minimum at $r_0$ since there is a jump at this point in the width of the minimal tube containing $\Omega \cap B(\mathbf{x}, r)$.

If $r \leq w$, then in a similar way to the argument above,

$$\beta_2(x, r) = \frac{D - d}{D + 2}.$$

For $r > w$, the function $\beta_2$ cannot be calculated so easily. However, we can approximate it in the following way. We apply the regions $T_d(\mathbf{x}, r) := B_d(\mathbf{x}, r) \times B_{D-d}(\mathbf{x}, r)$ (where $B_d$ and $B_{D-d}$ are $d$-dimensional and $(D - d)$-dimensional balls respectively) to form the function

$$\widetilde{\beta}_2^2(\mathbf{x}, r)$$
$$= \min_{d-\text{flat}L} \int_{\Omega \cap T_d(\mathbf{x}, r)} \left( \frac{\text{dist}(\mathbf{x}, L)}{r} \right)^2 \frac{d\mathbf{x}}{\text{vol}_D(\Omega \cap T_d(\mathbf{x}, r))}.$$

We simplify this function for $w \leq r \leq r_0$ as follows:

$$\widetilde{\beta}_2(\mathbf{x}, r) = \int_{\Omega_1 \cap T_d(\mathbf{x}, r)} \left( \frac{\text{dist}(\mathbf{x}, L_1)}{r} \right)^2 \frac{d\mathbf{x}}{\text{vol}(\Omega_1 \cap T_d(\mathbf{x}, r))}$$
$$= \frac{\int_{B_{D-d}(\mathbf{x}, w)} ||x||^2 d\mathbf{x}}{r^2 \int_{B_{D-d}(\mathbf{x}, w)} d\mathbf{x}} = \frac{\int_{t=0}^{w} t^2 \cdot t^{D-d-1} dt}{r^2 \int_{t=0}^{w} t^{D-d-1} dt}$$
$$= \frac{D - d}{D - d + 2} \cdot \frac{w^2}{r^2} = C_{d,D} \cdot \frac{w^2}{r^2},$$

and thus note that it is decreasing on $[w, r_0]$.

We note that for $r \geq w$

$$T_d(\mathbf{x}, \sqrt{r^2 - w^2}) \subseteq B(\mathbf{x}, r) \subseteq T_d(\mathbf{x}, r),$$

and clearly if $r \leq \sqrt{r_0^2 - w^2}$: $T_d(\mathbf{x}, r) \setminus \Omega_1 = \emptyset$. Consequently for

$$\nu_0 = \sqrt{1 - \frac{w^2}{r_0^2}},$$

we have that for $r \leq \sqrt{r_0^2 - w^2}$,

$$\nu_0^{D-d+2} \widetilde{\beta}_2(\mathbf{x}, r) \leq \beta_2(\mathbf{x}, r) \leq \nu_0^{d-D} \widetilde{\beta}_2(\mathbf{x}, r),$$

that is, $\widetilde{\beta}_2$ and $\beta_2$ are comparable for $r \leq \nu_0 \cdot r_0$, and the comparability constant approaches one as $w/r_0$ approaches zero.

We conclude by showing that the local minimum of $\beta_2$ occurs nearby $r_0$. For this purpose we will find a sufficiently small constant $\varepsilon$ such that

$$\varepsilon r_0 > w \quad \text{and} \quad f(\varepsilon) \gtrsim 1, \tag{13}$$

where

$$f(\varepsilon) := \frac{\beta_2^2(\mathbf{x}, (1 + \varepsilon)r_0)}{\beta_2^2(\mathbf{x}, r_0)} =$$

$$\frac{\min_{d-\text{flat}L} \int_{\Omega \cap B_d(\mathbf{x}, (1+\varepsilon)r_0)} \text{dist}(\mathbf{x}, L) d\mathbf{x}}{\text{vol}_D(\Omega \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0))}$$
$$\cdot \frac{\text{vol}_D(\Omega \cap B_d(\mathbf{x}, r_0))}{\min_{d-\text{flat}L} \int_{\Omega \cap B_d(\mathbf{x}, r_0)} \text{dist}(\mathbf{x}, L) d\mathbf{x}} \cdot \frac{1}{(1 + \varepsilon)^2}. \tag{14}$$

We estimate $f(\varepsilon)$ by bounding from below its first multiplicative fraction in (14). Assume without loss of generality that $\mathbf{y} \in T_2$. In order to estimate $f(\varepsilon)$ we first notice the following upper bounds on volumes of tubes within balls:

$$\text{vol}_D(\Omega \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0)) \leq \text{vol}_D(\Omega_1 \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0))$$
$$+ (K - 1) \cdot \text{vol}_D(\Omega_2 \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0))$$

We bound the ratio between those volumes as follows

$$g(\varepsilon) = \frac{\text{vol}_D(\Omega_2 \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0))}{\text{vol}_D(\Omega_1 \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0))}$$
$$\gtrsim \frac{\text{vol}_D(B_d(\mathbf{x}, 2\varepsilon^{\frac{1}{2}}r_0) \times B_{D-d}(\mathbf{x}, w))}{\text{vol}_D(B_d(\mathbf{x}, (1 + \varepsilon)r_0) \times B_{D-d}(\mathbf{x}, w))} \gtrsim \varepsilon^{\frac{d}{2}}, \tag{15}$$

where the first inequality was derived using (13).

Next, we bound from below the numerator of the first multiplicative fraction in (14) as follows (while using the notation $\mathbf{z}$ for the center of mass of $\Omega \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0)$):

$$\min_{d-\text{flat}L} \int_{\Omega \cap B_d(\mathbf{x}, (1+\varepsilon)r_0)} \text{dist}^2(\mathbf{x}, L) d\mathbf{x} \gtrsim$$

$$\min_{d-\text{flat } L \text{ containing } \mathbf{z}} \int_{\Omega_2 \cap B_d(\mathbf{x}, (1+\varepsilon)r_0)} \text{dist}^2(\mathbf{x}, L) d\mathbf{x} +$$

$$\min_{d-\text{flat}L} \int_{\Omega_1 \cap B_d(\mathbf{x}, (1+\varepsilon)r_0)} \text{dist}^2(\mathbf{x}, L) d\mathbf{x}. \tag{16}$$

We remark that we applied the assumption $r_0 \gg w$ in neglecting the integral over $\Omega_1 \cap \Omega_2 \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0)$. Equation (15) implies that $\mathbf{z}$ is sufficiently close to $\mathbf{x}$ so that the first term in the RHS of (16) is approximately $\varepsilon r_0^2 \text{vol}_D(\Omega_2 \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0))$. For the second term, the best $L_2$ $d$-flat is $L_1$ (on which $\Omega_1$ is centered) and thus this term is estimated by $w^2 \text{vol}_D(\Omega_1 \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0))$. We can thus estimate the first multiplicative fraction in (14) using $g(\varepsilon)$ of (15) as follows

$$\frac{\min_{d-\text{flat}L} \int_{\Omega \cap B_d(\mathbf{x}, (1+\varepsilon)r_0)} \text{dist}(\mathbf{x}, L) d\mathbf{x}}{\text{vol}_D \Omega \cap B_d(\mathbf{x}, (1 + \varepsilon)r_0)}$$
$$\gtrsim \frac{w^2 + g(\varepsilon)\varepsilon r_0^2}{1 + (K - 1)g(\varepsilon)}. \tag{17}$$

Notice that the second multiplicative fraction in (14) is easily estimated in the following way:

$$\frac{\int_{\Omega \cap B_d(\mathbf{x},r_0)} \text{dist}(\mathbf{x},L)\text{d}\mathbf{x}(1+\varepsilon)^2}{\text{vol}_D(\Omega \cap B_d(\mathbf{x},r_0))} \approx (1+\varepsilon)^2 w^2 \ . \quad (18)$$

Combining (14), (17) and (18), we obtain that $\varepsilon$ needs to satisfy the equation:

$$\frac{C_1 w^2 + C_2 g(\varepsilon)\varepsilon r_0^2}{1+g(\varepsilon)} \gtrapprox (1+\varepsilon)^2 w^2 \ . \quad (19)$$

Equation (13) and (19) imply the requirement: $g(\varepsilon)r_0^2 > C_4 w^2$. In view of (15) and (13) we obtain that $\varepsilon$ can be taken to be in the order of $w/r_0$ if $d \le 4$ and in the order of $(w/r_0)^{4/d}$ if $d \ge 5$. By carefully checking all comparability estimates above we note that they tend to be equalities or strict inequalities when $w/r_0$ approaches zero.

$\square$

# References

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

[2] R. Epstein, P.W. Hallinan, and A.L. Yuille. $5 \pm 2$ eigenimages suffice: an empirical investigation of low-dimensional lighting models. page 108, jun. 1995.

[3] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 11–18, 2003.

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

[5] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[6] P. Bradley and O. Mangasarian. k-plane clustering. *J. Global optim.*, 16(1):23–32, 2000.

[7] P. Tseng. Nearest $q$-flat to $m$ points. *Journal of Optimization Theory and Applications*, 105(1):249–252, April 2000.

[8] T. Zhang, A. Szlam, and G. Lerman. Median $K$-flats for hybrid linear modeling with many outliers. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on Computer Vision*, pages 234–241, Kyoto, Japan.

[9] T. E. Boult and L. G. Brown. Factorization-based segmentation of motions. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 179–186, 1991.

[10] K. Kanatani. Motion segmentation by subspace separation and model selection. In *Proc. of 8th ICCV*, volume 3, pages 586–591. Vancouver, Canada, 2001.

[11] K. Kanatani. Evaluation and selection of models for motion segmentation. In *7th ECCV*, volume 3, pages 335–349, May 2002.

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

[13] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In *ECCV*, volume 4, pages 94–106, 2006.

[14] A. Y. Yang, S. R. Rao, and Y. Ma. Robust statistical estimation and segmentation of multiple subspaces. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, page 99, Washington, DC, USA, 2006. IEEE Computer Society.

[15] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *CVPR 07*, 2007.

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

[17] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *Int. J. Comput. Vision*, 81(3):317–330, 2009.

[18] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 09)*, pages 2790 – 2797, 2009.

[19] P. W. Jones. Rectifiable sets and the traveling salesman problem. *Invent Math*, 102(1):1–15, 1990.

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

[21] G. Lerman. Quantifying curvelike structures of measures by using $L_2$ Jones quantities. *Comm. Pure Appl. Math.*, 56(9):1294–1365, 2003.

[22] K. Fukunaga and D. R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Trans. Comput.*, 20(2):176–183, 1971.

[23] A. V. Little, J. Lee, Y.-M. Jung, and M. Maggioni. Estimation of intrinsic dimensionality of samples from noisy low-dimensional manifolds in high dimensions with multiscale svd. In *Statistical Signal Processing, 2009. SSP '09. IEEE/SP 15th Workshop on*, pages 85–88, 2009.

[24] A. V. Little, Y.-M. Jung, and M. Maggioni. Multiscale estimation of intrinsic dimensionality of data sets. In *Manifold learning and its applications : papers from the AAAI Fall Symposium*, pages 26–33, 2009.

[25] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Randomized hybrid linear modeling by local best-fit flats. pages 1927 – 1934, jun. 2010.

[26] Y.-M. Jung, A.V. Little, M. Maggioni, and L. Rosasco. Multiscale estimation of intrinsic dimensionality of noisy data sets. preprint.

[27] G. Lerman and T. Zhang. Probabilistic recovery of multiple subspaces in point clouds by geometric $\ell_p$ minimization. Submitted April 2010. Available at http://arxiv.org/abs/1002.1994.

[28] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.

[29] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and Hall, 2nd edition, 2001.

[30] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

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

[32] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using powerfactorization and gpca. *Int. J. Comput. Vision*, 79(1):85–105, 2008.

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

[34] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *CVPR*, 2007.

[35] G. Chen and G. Lerman. Motion segmentation by SCC on the Hopkins 155 database. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on Computer Vision*, pages 759–764, Kyoto, Japan.

[36] F. Lauer and C. Schnorr. Spectral clustering of linear subspaces for motion segmentation. pages 678 –685, sep. 2009.

[37] E. Levina and P. J. Bickel. Maximum likelihood estimation of intrinsic dimension. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 777–784. MIT Press, Cambridge, MA, 2005.

[38] S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(RapidPosts), 2009.

[39] S. Yue, X. Wang, and M. Wei. Application of two-order difference to gap statistic. *Trans. Tianjin Univ.*, 14(3):217–221, 2008.