Randomized Hybrid Linear Modeling by Local Best-fit Flats[*]

Teng Zhang[♮] Arthur Szlam[♭] Yi Wang[♮] Gilad Lerman[♮]
♮ School of Mathematics
University of Minnesota
{zhang620, wangx857, lerman}@umn.edu
♭ Courant Institute of Mathematical Sciences
New York University
aszlam@courant.nyu.edu

Abstract

The hybrid linear modeling problem is to identify a set of d-dimensional affine sets in $\mathbb{R}^D$. It arises, for example, in object tracking and structure from motion. The hybrid linear model can be considered as the second simplest (behind linear) manifold model of data. In this paper we will present a very simple geometric method for hybrid linear modeling based on selecting a set of local best fit flats that minimize a global ℓ_1 error measure. The size of the local neighborhoods is determined automatically by the Jones' β_2 numbers; it is proven under certain geometric conditions that good local neighborhoods exist and are found by our method. We also demonstrate how to use this algorithm for fast determination of the number of affine subspaces. We give extensive experimental evidence demonstrating the state of the art accuracy and speed of the algorithm on synthetic and real hybrid linear data.

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

1. Introduction

Many data sets can be modeled as unions of affine subspaces. This Hybrid Linear Modeling (HLM) finds diverse applications in many areas, such as motion segmentation in computer vision, hybrid linear representation of images, classification of face images, and temporal segmentation of video sequences (see e.g., [1, 2]).

Several algorithms have been suggested for solving this problem, for example the K-flats (KF) algorithm or any of its variants [3, 4, 5, 6, 7], Subspace Separation [8, 9, 10], Generalized Principal Component Analysis (GPCA) [1], Local Subspace Affinity (LSA) [11], Agglomerative Lossy Compression and Spectral Curvature Clustering (SCC) [12]. Some algorithms for modeling data by a mixture of more general surfaces have been successfully applied to HLM [13, 14].

In this paper, inspired by [15, 16, 17] and [18, 19], we will describe an extremely straightforward geometric method for hybrid linear modeling that can either be used in a stand alone manner or as an initialization of any of the above methods. The basic idea is that for a data set sampled from a hybrid linear model and a random point of it $\mathbf{x}$: the principal components of a neighborhood of appropriate size of $\mathbf{x}$ often give a good approximation to its nearest subspace. An appropriate neighborhood size needs to be larger than the noise, so that the affine cluster is recognized. However, not too large so that the neighborhood intersects multiple clusters. Such neighborhoods (in which a subspace is clearly distinguished) always exist for points far enough from the intersection of subspaces (i.e., most of points), as long as the following two assumptions are satisfied: Samples are sufficiently dense along local regions of the subspaces and data points sufficiently far from the intersection of subspaces are mostly surrounded by neighbors of the same subspace (this is true when the affine Grassmannian distance between subspaces is sufficiently large and the noise level is sufficiently small).

The contributions of this work are as follows: we make precise the local fit heuristic, using the ℓ_2 version of Jones' β numbers [15, 16, 17], and state a theorem that tells us under certain geometric conditions how to calculate the size of the optimal local neighborhood. Using this, we introduce a new algorithm for affine clustering based on the above heuristic. At each of a randomly chosen subset of the data, we build a candidate flat by calculating the principal components of a large neighborhood which still lies in only one affine cluster. The algorithm then selects among the best fit flats of each of the neighborhoods to build a global model using an ℓ_1 error energy. We show experimentally that this algorithm obtains state of the art accuracies on real and synthetic HLM problems while running extremely fast (often on the order of ten times faster than most of the previously mentioned methods). Note that the two parts of the algorithm are independent and can be used with other al-

[*] This work was supported by NSF grants DMS-0612608, DMS-0811203 and DMS-0915064. Thanks to Peter Jones, Mauro Maggioni and Amit Singer for some brief discussions that motivated our exploration for a multiscale SVD-based HLM algorithm and to the IMA for a stimulating multi-manifold modeling workshop.

gorithms. In particular, we can use the local fit heuristic to initialize other HLM algorithms. We will give experimental evidence to show that the K-flats algorithm [7] is improved by such initialization. We also show how to use this fast algorithm to 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 two parts of the above algorithm, and state a theorem giving conditions that guarantee that good neighborhoods can be found. Section 3 carefully tests the algorithm on both artificial data of synthetic hybrid linear models and real data of motion segmentation in video sequences. 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. Randomized local best fit flats

The algorithm partitions a data set $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$ into K clusters $\mathrm{X}_1, \ldots, \mathrm{X}_K$, with each cluster approximated by a d-dimensional affine subspace, which we refer to as d-flats or flats. We sketch it as follows, while suppressing details that appear later in Algorithms 3 and 2.

Algorithm 1 HLM by randomized local best fit flats

```
Input: $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subseteq \mathbb{R}^D$: data, $d$: dimension
  of subspaces, $C$: number of candidate planes, $K$: number
  of output flats/clusters ($K < C$), other parameters used
  by Algorithms 3 and 2
Output: A partition of X into $K$ disjoint clusters $\{\mathrm{X}_i\}_{i=1}^K$,
  each approximated by a single flat.
  Steps:
  • Pick $C$ random points in X
  • For each of the $C$ points find appropriate local neigh-
    borhoods using Algorithm 3
  • Generate $C$ flats (by PCA) for the $C$ neighborhoods
    of the previous step
  • Choose $K$ flats from the $C$ flats above using Algo-
    rithm 2
  • Partition X by sending points to nearest $K$ flats above
```

The proposed algorithm breaks into two main parts. The first part finds a set of candidate flats. It takes as input the dimension of the flats to be found and the number of candidates to search for. It starts by randomly selecting one point for each candidate flat. The algorithm chooses a scale (that is, a number of neighbors) around each of the seed points. The best fit flats (in L^2 sense) for each of the chosen neighborhoods are collected as candidates. The method for choosing the best scale is described in Section 2.1 and sketched in Algorithm 3.

Algorithm 2 Greedy ℓ_1 candidate selection for HLM by randomized local best fit flats

```
Input: $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subseteq \mathbb{R}^D$: 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 $\tilde{\mathcal{L}}_j = L_j \bigcup \mathcal{L} \setminus L_{A_l}$
      • Set $s_j = \sum_{i=1}^N \min_{L \in \tilde{\mathcal{L}}_j} ||x_i - P_L x_i||$
    end for
    If $\min_j s_j < \sum_{i=1}^N \min_{L \in \{L_{A_1}, ..., L_{A_K}\}} ||x_i - P_L x_i||$,
    set $L_{A_l} := L_{\mathrm{argmin}\, s_j}$
  end for
```

The second part of the algorithm searches for a good set of flats from the candidates in a greedy fashion. A number K of desired flats and a measure of goodness of a K tuple of flats $G = G_X(L_1, ...L_K)$ is chosen; here, it will be the average ℓ_1 distance of each point to its nearest flat. 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.

The simplest choice of G is the sum of the squared distances of each point in X to its nearest flat. In our experiments, we will use an ℓ_1 energy, i.e., summing the distance of every point to its chosen flat. We have experimentally found that this energy is more robust to outliers than least squares error (see also [20] for a similar conclusion with a different implementation of ℓ_1 subspace minimization and [21] for partial theoretical justification). 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.

2.1. Choosing the optimal neighborhood

Choosing the correct neighborhood is crucial for the suc-

Algorithm 3 Neighborhood size selection for HLM by randomized local best fit flats

```
Input: $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subseteq \mathbb{R}^D$: 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
    • Set $\mathcal{N}_k$ to be 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 $\mathcal{N}(\mathbf{x}) := \mathcal{N}_{k-1}$
```

cess of the method. 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 look D dimensional. 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 scaled least squares error for d-flat approximation. In practice, for a neighborhood $\mathcal{N}$ of $\mathbf{x}_0$ the scaled least squares error for d-flat approximation, $\beta_2(\mathcal{N})$, is computed by the formula:

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

where P_L denotes the projection onto the flat L. This notion of scaled error introduced and utilized in [15, 16, 17], and considered recently in [18, 19] for dimension estimation. The procedure we have just described is summarized in Algorithm 3.

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, 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 β_2 introduced earlier when having an underlying continuous set Ω (here it is the union of tubes) in a ball of center $\mathbf{x}$ and radius r is defined as follows:

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

where the minimum is over all d-flats L (see also [17]).

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

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

and

$$r_0 := \mathrm{dist}(\mathbf{y}, \mathbf{x}). \qquad (3)$$

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

$$\beta_2((1 + \varepsilon) \cdot r_0) \gtrapprox \beta_2(r_0) \qquad (4)$$

for sufficiently small ε, 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.

We remark that by imposing an upper bound on the widths of the tubes in the theorem above and a lower bound on the dihedral angles between the flats, then the local condition $r_0 > w_{i^*}$ (required by the theorem) is satisfied at any point $\mathbf{x}$ which has distance larger than order of $\max_{1\leq i\leq K} w_i$ from the intersection of all flats.

2.2. Some technical notes about the proposed algorithm

Note that the first minimum in the Theorem excludes the left endpoint. 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. In the experiments below, we will show the results of the algorithm with both notions of "first" local minimum.

The second technical detail concerns the choice of the random points used for candidate generation. We use the mean shift technique: given a point $\mathbf{x}$, update $\mathbf{x}$ as the center of its neighborhood several times. The method shifts the point to a denser region, resulting in a more accurate estimation of the flats. In the experiments below, we will show the results with and without mean shift biased seed selection.

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 algorithms.

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

$$\text{error}\% = \frac{\text{\# of misclassified inliers}}{\text{\# of total inliers}} \times 100\% \,. \qquad (5)$$

In all the experiments below, the number C in Algorithm 1 is 70 times the number of subspaces, the number p in Algorithm 2 is 3 times the number of subspaces, and the number T in Algorithm 3 is 2. We run experiments with and without mean shifts; the experiments using mean shifts use 10-nearest neighbors and 5 shifts. According to our experience the LBF algorithm is 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.

3.1. Simulated data

We compare our algorithm with the following algorithms: Mixtures of PPCA (MoPPCA) [4], K-flats (KF) [7], Local Subspace Analysis (LSA) [11], Spectral Curvature Clustering (SCC) [12], Random Sample Consensus (RANSAC) [22] and GPCA with voting (GPCA) [2]. We use the Matlab codes of the GPCA, MoPPCA and KF algorithm from http://perception.csl.uiuc.edu/gpca, the SCC algorithm from http://www.math.umn.edu/∼lerman/scc and the LSA, RANSAC algorithms from http://www.vision.jhu.edu/db.

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 [12]), 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 ℓ_2 error for MoPPCA as well as KF. The number of restarts was restricted by the running time and accuracy. RANSAC uses the oracle inlier bound given by the model's noise variance.

The simulated data represents various instances of K linear subspaces in $\mathbb{R}^D$. If their dimensions are fixed and equal d, we follow [12] and refer to the setting as $d^K \in \mathbb{R}^D$. If they are mixed, then we follow [2] 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. Within each subspace, the underlying sampling distribution is the sum of a uniform distribution in a d-dimensional ball of radius 1 of that subspace (centered at the origin for the case of linear subspaces) and 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 (including ours) do not support mixed dimensions natively, we assume each subspace has the maximum dimension in the experiment.

The mean (over 100 instances) misclassification rate of the various algorithms is recorded in Table 1. The mean running time is shown in Table 2. In each of the Tables, our algorithm is labeled LBF (Local Best-fit Flats); our algorithm with mean shifts and using the modified choice of good neighborhood described in section 2.2 is labeled LBFMS.

3.2. Motion segmentation data

We test the proposed algorithm 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

[1] The RANSAC code we use (and most standard versions of RANSAC) depend on a user supplied inlier threshold. The first part of our algorithm can in some sense be considered to be the automatic detection of this inlier threshold; and if this is provided by the user, the initialization we have described is no longer useful, as we would simply pick the largest neighborhood so that the distance from any point to its projection is smaller than the user supplied bound. The experiments in the table use the oracle choice of inlier bound (given by the true noise variance), and so here RANSAC has an advantage over the other algorithms listed.

Table 1. Mean percentage of misclassified points in simulation for linear-subspace cases or affine-subspace case. The proposed algorithm as in Section 2.2 is in the row labeled LBFMS, and the “vanilla” version is in the row labeled by LBF

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
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	2.7	3.0	2.7	2.6	7.0	11.1	1.5	2.1	1.4	1.9
LBFMS	3.1	3.0	2.7	2.8	7.0	11.3	4.3	5.5	2.1	1.9
RANSAC[1]	3.3	2.6	2.3	2.2	8.6	9.8	0.9	6.7	1.8	1.4

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
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	2.1	0.1	1.8	0.5	3.7	0.0	0.5	0.0	0.0
LBFMS	0.4	2.0	0.1	2.6	0.7	6.0	0.0	0.3	0.0	0.0
RANSAC[1]	13.2	12.2	11.5	11.2	31.5	28.4	2.6	9.2	1.1	2.2

Table 2. Mean running time for linear-subspaces cases and affine-subspaces cases. The proposed algorithm as in Section 2.2 is in the row labeled LBFMS, and the “vanilla” version is in the row labeled by LBF.

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
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	0.3	0.3	0.3	0.3	0.9	1.1	0.6	0.6	0.6	0.8
LBFMS	0.3	0.3	0.3	0.3	1.1	1.4	0.4	0.5	0.7	0.9
RANSAC[1]	0.01	0.01	0.02	0.06	0.03	0.06	3.5	3.8	0.9	3.4

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
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	0.3	0.3	0.3	0.3	1.1	1.3	0.5	0.6	0.7	0.9
LBFMS	0.3	0.3	0.3	0.3	1.1	1.5	0.4	0.5	0.7	0.9
RANSAC[1]	0.02	0.1	0.2	0.6	0.2	0.3	3.2	3.7	2.0	3.5

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^{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 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 [8] that under affine camera models and with some mild conditions, 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 $d = 3$, and use affine flats.

We compare our algorithm with the following: improved GPCA for motion segmentation (GPCA) [23], K-flats (KF) [7] (implemented for linear subspaces), Local Linear Manifold Clustering (LLMC) [13], Local Subspace Analysis (LSA) [11], Multi Stage Learning (MSL) [24], Spectral Curvature Clustering (SCC) [12], Sparse Subspace Clustering (SSC) [14], and Random Sample Consensus (RANSAC) [22, 25, 26]. As before, our algorithm is labeled LBF (Local Best-fit Flats); our algorithm with mean shifts and using the modified choice of good neighborhood described in section 2.2 is labeled LBFMS.

For these algorithms, we copy the results from http://www.vision.jhu.edu/data/hopkins155 (they are based on experiments reported in [26] and [13]) and [27], and we just record the mean misclassification rate and the median

Table 3. The mean and median percentage of misclassified points for two-motions and three-motions in Hopkins 155 database. The proposed algorithm as in Section 2.2 is in the row labeled LBFMS, and the “vanilla” version is in the row labeled by LBF

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)	3.31	0.00	3.29	0.00	4.31	0.12	3.40	0.00
LBFMS($4K$,3)	3.05	0.00	0.78	0.00	1.73	0.03	2.34	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
SCC($4K$,4)	1.30	0.04	1.07	0.44	3.68	0.44	1.46	0.16
SSC-N	1.12	0.00	0.02	0.00	0.62	0.00	0.82	0.00

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)	8.42	1.29	14.80	9.21	20.45	20.45	10.38	1.63
LBFMS($4K$,3)	6.87	1.47	1.40	0.00	24.10	24.10	6.76	0.89
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
SCC($4K$,4)	5.68	2.96	2.35	2.07	10.94	10.94	5.31	2.40
SSC-N	2.97	0.27	0.58	0.00	1.42	0.00	2.45	0.20

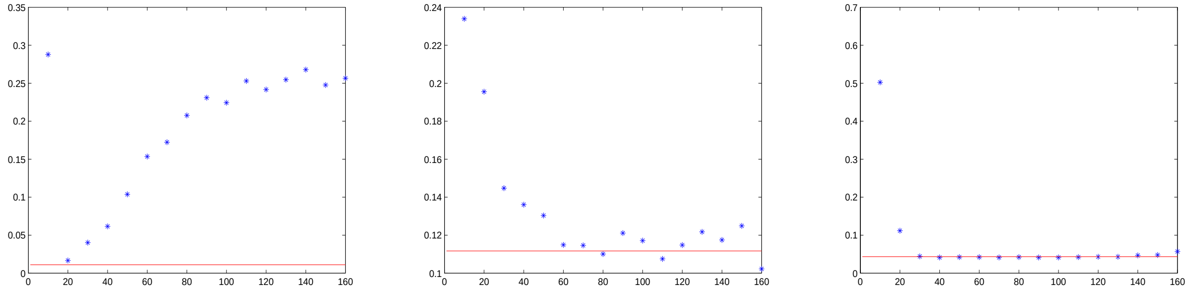

Figure 2. *Using our neighborhood choice to improve initialization of k-flats: the vertical axis is accuracy, and the horizontal axis is fixed neighborhood size in geometric farthest insertion for initialization of K flats. The red line is the result of using adapted neighborhoods. The data sets are #1,#2, and #3 as described in Section 3.4. Random initialization leads to errors of .4 or greater for all three data sets.*

misclassification rate for each algorithm for any fixed K (two or three-motions) and for the different type of motions (“checker”, “traffic” and “articulated”).

3.3. Discussion of Results

From Table 1 we can see that our algorithm performs well in various artificial instances of hybrid linear modeling (with both linear subspace and affine subspace), and its advantage is especially obvious with many outliers and affine subspaces. The robustness to outliers is a result of our use of the ℓ_1 error as loss function, and because of the random sampling. Also unlike many other methods, the proposed method natively supports affine subspace models.

Table 2 shows that the running time of the proposed algorithm is less than the running time of most other algorithms, especially GPCA, LSA and LSCC. The difference is large enough that we can also use the proposed algorithm as an initialization for the others. The algorithm is 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.

From Table 3 we can see that the local best-fit flat algorithm works well for the data set. Of all the methods tested, only SCC and SSC had better accuracy. However LBF ran 4 times faster than SCC and more than 100 times faster than SSC. In many of the cases where SSC performed better than LBF, the ℓ_1 energy (as well as the ℓ_2 energy) was lower for the labels obtained by LBF than the labels obtained by SSC. We thus suspect that good clustering of the Hopkins data requires additional type of clustering (e.g., bottleneck clustering) to be combined with subspace clustering (i.e., hybrid linear modeling).

Table 4. The percentage of incorrectness ($e\%$) and the average computation time t of the three methods SOD (LBF), ALC and GPCA.

		no minimum angle							minimum angle $= \pi/8$			
		$1^6 \in \mathbb{R}^5$	$2^4 \in \mathbb{R}^5$	$3^3 \in \mathbb{R}^5$	$10^2 \in \mathbb{R}^{15}$	$1^6 \in \mathbb{R}^3$	$2^4 \in \mathbb{R}^3$	$3^3 \in \mathbb{R}^4$	$1^6 \in \mathbb{R}^3$	$2^4 \in \mathbb{R}^3$	$3^3 \in \mathbb{R}^4$	$10^2 \in \mathbb{R}^{15}$
SOD (LBF)	e%	17	3	2	0	55	29	19	3	5	5	0
	t	3.51	4.07	3.37	7.31	3.13	3.77	3.85	3.09	3.45	3.32	6.78
ALC	e%	1	0	0	16	34	31	1	0	10	1	13
$\epsilon = 0.05$	t	23.74	43.44	59.14	1370.92	20.49	37.49	53.59	20.22	37.41	54.11	1354.11
GPCA	e%	88	100	100	100	27	100	100	13	100	100	100
	t	0.03	0.09	0.12	1.30	0.06	0.09	0.12	0.04	0.09	0.12	1.30

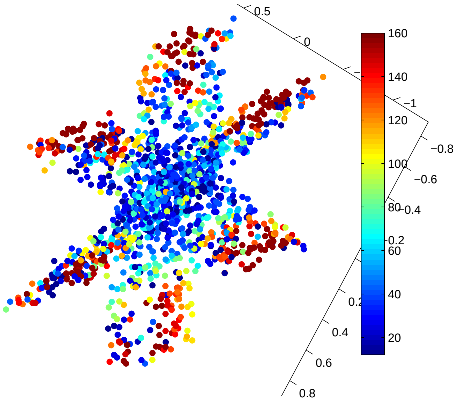

Figure 1. *Data set #3 from Section 3.4. The color value represents the number of neighbors chosen at that point. Note that the algorithm chooses smaller neighborhoods for points closer to the intersection of the planes.*

3.4. Initializing K-flats with good neighborhoods

Here we demonstrate that our choice of neighborhoods 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.

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 1. 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 2. 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 1 we plot the number of neighbors picked by our algorithm for each point of a realization of data set #3.

3.5. Automatic determination of the number of affine sets

In this section we show experimentally that using the elbow method on the least squares errors of the outputs of the randomized best fit flat method can accurately determine the number of affine clusters.

Let W_k be the total mean squared distance of a data set to the flats returned by our algorithm with k affine clusters specified; as k increases, W_k decreases. A classical method for determining the correct number k is to find the "elbow", or the k past which adding more clusters does not significantly decrease the error. We use the Second Order Difference (SOD) formulation of this heuristic [28]:

$$SOD(\ln W_k) = \ln W_{k-1} + \ln W_{k+1} - 2 \ln W_k, \qquad (6)$$

Then the optimal k is found by:

$$k_{opt} = \arg\max_k SOD(\ln W_k). \qquad (7)$$

We compare SOD (LBF), i.e., SOD applying LBF, with ALC [29] and GPCA [2] on a number of artificial data sets. Similarly to Section 3.1, data sets were generated by the Matlab code borrowed from the GPCA package in http://perception.csl.uiuc.edu/gpca with $100d$ samples from each subspace and 0.05 Gaussian noise. For the last four experiments, we restrict the angle between subspaces to be at least $\pi/8$ for separation. All algorithms are given the dimension d and we choose $k_{max} = 10$ in SOD (LBF). For ALC, we use the oracle choice of the parameter ϵ, setting it equal the true noise level. For GPCA, we embed the data to a $d+1$ subspace by PCA and let the tolerance of rank detection be 0.05 [1, 2]. There is no automatic way to choose this tolerance, so we tried different values and picked the one which matched the ground truth the best. Each experiment is repeated 100 times and the error ($e\%$) and the average computation time t (in seconds) are recorded in Table 4.

4. Conclusions and future work

We presented a very simple geometric method for hybrid linear modeling based on selecting a set of local best fit flats that minimize a global ℓ_1 error measure. The size of the local neighborhoods is determined automatically using the ℓ_2 β numbers; it is proven under certain geometric conditions that good local neighborhoods exist and are found 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.

References

[1] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *IEEE TPAMI*, 27(12), 2005.

[2] 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.

[3] A. Kambhatla and T. K. Leen. Fast non-linear dimension reduction. In *6th NIPS*, pages 152–159, 1994.

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

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

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

[7] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *CVPR 03*, volume 1, pages 11–18, 2003.

[8] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998.

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

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

[11] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In *ECCV 06*, volume 4, pages 94–106, 2006.

[12] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *IJCV*, 81(3):317–330, 2009.

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

[14] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR 09*, pages 2790 – 2797, 2009.

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

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

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

[18] 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 *SSP 09*, pages 85–88, 2009.

[19] 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.

[20] T. Zhang, A. Szlam, and G. Lerman. Median K-flats for hybrid linear modeling with many outliers. 2nd international workshop on subspace methods at ICCV 2009.

[21] G. Lerman and T. Zhang. Probabilistic recovery of multiple subspaces in point clouds by geometric ℓ_p minimization. Available at http://arxiv.org/abs/1002.1994.

[22] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395, June 1981.

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

[24] 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.

[25] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London A*, 356:1321–1340, 1998.

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

[27] G. Chen and G. Lerman. Motion segmentation for hopkins 155 database by SCC. 4th IEEE international workshop on dynamical vision at ICCV 2009.

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

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