Multiscale Geometric and Spectral Analysis of Plane Arrangements

Guangliang Chen[*], Mauro Maggioni[*,†]
[*]Mathematics and [†]Computer Science Departments, Duke University, PO Box 90320, Durham, NC 27708, USA
{glchen,mauro}@math.duke.edu

Abstract

Modeling data by multiple low-dimensional planes is an important problem in many applications such as computer vision and pattern recognition. In the most general setting where only coordinates of the data are given, the problem asks to determine the optimal model parameters (i.e., number of planes and their dimensions), estimate the model planes, and cluster the data accordingly. Though many algorithms have been proposed, most of them need to assume prior knowledge of the model parameters and thus address only the last two components of the problem. In this paper we propose an efficient algorithm based on multiscale SVD analysis and spectral methods to tackle the problem in full generality. We also demonstrate its state-of-the-art performance on both synthetic and real data.

1. Introduction

It is of interest in various applications to model data by hybrid linear models, i.e., using a union of low-dimensional planes. In examples such as image processing [10], computer vision [15], and pattern recognition [14], these models have shown promise in solving modeling, clustering and classification tasks. Notwithstanding much recent work [18], the problem of finding a set of planes approximating a point cloud is still open at least from two perspectives: from a theoretical standpoint most existing algorithms do not have finite-sample guarantees, and from a practical standpoint they require that the number of planes and their dimensions be given. Finally, computational efficiency is also important, and several existing algorithms are slow in theory and in practice. We formulate the following problems:

Problem 1. *(**Model Selection**) Given a data set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$ sampled around a collection of K planes $\pi_1, \ldots, \pi_K$ of dimensions $d_1, \ldots, d_K$ respectively (where K and d_k are both unknown), determine the model parameters K, $(d_k)_{k=1}^K$ and $\{\pi_k\}_{k=1}^K$.*

Problem 2. *(**Subspace Clustering**) With the same data as in Problem 1 and prior knowledge on the model parameters $K, (d_k)_{k=1}^K$, cluster the data into K groups corresponding to the model planes.*

The two problems are sometimes confused in the literature, which has mostly focused on Problem 2. We think it is important to keep them separate as their complexity is quite different, and so should be algorithms attempting their solution. Problem 2 has been studied extensively during the past decade, leading to a handful of proposed algorithms [15, 19, 20, 13, 8, 14, 1, 5, 21, 12]; for a comprehensive review of the subspace clustering methods, we refer the reader to [18]. However, even several latest methods [14, 1, 5, 21, 12] require K and/or $(d_k)_{k=1}^K$ as inputs. To the best of our knowledge, the only method that can solve Problem 1 is Agglomerative Lossy Coding (ALC) [13]. Nevertheless, ALC has an unknown tolerance parameter whose different values lead, in theory and in practice, to different numbers of clusters. Moreover, ALC is observed to be a slow method (see Section 3), with an unknown number of iterations needed for convergence, and has no finite-sample guarantees, albeit it seems to perform well with small amounts of data and with ideal settings of the tolerance. Iterative methods [6, 16, 9] that alternate between the subspace estimation and data clustering steps also exist. Typically, these methods are initialized with a random guess of the planes or of the clusters. Though extremely fast, they often converge to a local optimal solution, far from the global optimal one, especially in the cases of affine subspaces and of mixed dimensions. This makes iterative methods more suitable as a post-optimization tool for improving other algorithms, than as a clustering method on their own. Therefore, we will not discuss iterative methods in the rest of the paper.

In this paper, we propose a new algorithm solving Problems 1, 2. Our approach is inspired by ideas from Multiscale Singular Value Decomposition (MSVD) [11] and spectral methods [20, 1]. It only requires as inputs upper bounds $K_{\max}$ and $d_{\max}$ for the number of planes and their dimensions, respectively. Then, it automatically determines the model parameters $(K; d_1, ..., d_K, \pi_1, \ldots, \pi_K)$

with a near-optimal number of samples and computational cost, and, upon request and with additional cost, clusters the data accordingly. Moreover, it is robust to noise and extremely fast. Finally, it may also be used to estimate crucial parameters needed by other state-of-the-art algorithms such as those mentioned above.

The rest of the paper is organized as follows. Section 2 presents our methodology for solving Problems 1 and 2. Numerical experiments are then conducted in Section 3 to test the algorithm against existing methods. Finally, the paper is concluded in Section 4.

2. Methodology

We start by tackling Problem 1. First of all, we discuss the complexity of the problem at hand: the cost of encoding the K planes (in generic position) is $\sum_{1\leq k\leq K} d_k D = O(KD\max_k d_k)$, since in order to encode π_k we need to store d_k vectors in $\mathbb{R}^D$. Ideally, we would like to produce an algorithm with comparable computational and sampling complexity. Note that this is independent of N, and in particular much smaller than the complexity of Problem 2, which requires at the very least visiting every one of the N points in the given data. Our algorithm will indeed have computational complexity $O(K_{\max}d_{\max}(d_{\max}+K_{\max})D)$, where $d_{\max}, K_{\max}$ are given upper bounds on d_k and K, up to some logarithmic factors in $d_{\max}$ and $K_{\max}$, and with a constant that depends only on a notion of "geometric complexity of the problem" and on the size of the noise that affects the points. Our approach will favor configurations of the K planes such that a random point $\mathbf{x} \in \pi_k$ has a large $\mathbb{R}^D$ neighborhood that only intersects π_k, and no other plane. The average size of such neighborhoods is a natural measure of complexity of the underlying geometry that captures the intuition that configurations with planes with many intersections and/or close-by are harder to resolve, i.e., it will require more samples and possibly smaller noise for the planes to be correctly identified.

Our approach to Problem 1 makes use of this assumption and starts by finding $n_0 = O(K_{\max}\log K_{\max})$ good local regions, together with their dimension estimates $(\hat{d}_k)_{k=1}^{n_0}$ and best approximating planes $\{\hat{\pi}_k\}_{k=1}^{n_0}$ by using a Multiscale SVD analysis (MSVD) [11, 4]. Then these planes $\{\hat{\pi}_k\}_{k=1}^{n_0}$ are aligned together by a spectral method to generate the final estimates of the model parameters $(K; d_1, ..., d_K, \pi_1, \ldots, \pi_K)$. At this point, upon request, and with additional cost, we may proceed to solve Problem 2 and assign points to planes. This type of approach was inspired by [11], suggested in [4] and independently in [21].

2.1. Setup and assumptions

Our assumptions are that the probability of sampling from each plane is roughly $1/K$ and that the points on each plane are well-distributed. More precisely, let $\mu_{\mathcal{X}}$ be a probability measure in $\mathbb{R}^D$ which is supported in $\left(\cup_{k=1}^K \pi_k\right) \cap [-M, +M]^D$ where $\{\pi_k\}$ are affine d_k-planes. We shall assume that $\mu_{\mathcal{X}}(\pi_{k_1} \cap \pi_{k_2}) = 0$ if $k_1 \neq k_2$, and that the probability is well-distributed across the planes, i.e., $\exists c_1 : \mu_{\mathcal{X}}(\pi_k) \geq c_1/K$, for all k. Conditioning $\mu_{\mathcal{X}}$ to π_k we obtain a probability measure $\mu_{k,\mathcal{X}}$, for which we assume two types of regularity around a point $\mathbf{x} \in \pi_k$: (a) volume regularity, i.e., $\exists c_2 : \forall r \leq M$, $c_2^{-1} r^{d_k} \leq \mu_{k,\mathcal{X}}(B_{\mathbf{x}}(r)) \leq c_2 r^{d_k}$, where $B_{\mathbf{x}}(r)$ is the $\mathbb{R}^D$-ball centered at $\mathbf{x}$ of radius r; (b) shape regularity, i.e., for any $r > 0$ and a random variable $X_{k,\mathbf{x},r} \sim \mu_{k,\mathcal{X}}$ (restricted on $B_{\mathbf{x}}(r)$), the set of eigenvalues of its covariance satisfy $\lambda(\mathrm{cov}(X_{k,\mathbf{x},r})) \subset \frac{r^2}{d_k}[\lambda_{\min}, \lambda_{\max}]$ for fixed constants $\lambda_{\min}, \lambda_{\max} > 0$. Let $E_{\mathbf{x}} := \{r > 0 : B_{\mathbf{x}}(r) \cap (\cup_{j=1}^K \pi_j) \subseteq \pi_k\}$, and $R_{\mathbf{x}}^* := \sup E_{\mathbf{x}}$. We call the $r \in E_{\mathbf{x}}$ *good scales* at $\mathbf{x}$, and $R_{\mathbf{x}}^*$ the maximal good scale. $R_{\mathbf{x}}^*$ is large if $\mathbf{x}$ is far from other planes. We assume that the points $\mathbf{x}_i$ are corrupted by random Gaussian noise $\eta_i \sim \sigma D^{-\frac{1}{2}} N(0, I_D)$ (albeit assuming subgaussian distributions would be enough), and we will abuse notation to still denote the noisy samples by $\mathbf{x}_i$. Finally, we assume that $\exists c_3 < 1$ (small enough) : $\mu_{k,\mathcal{X}}(R_{\mathbf{x}}^* > 2\sigma) \geq 1 - c_3$, i.e. there is a substantial probability that a $\mu_{k,\mathcal{X}}$-distributed random sample has a maximal good scale above the noise scale.

2.2. Sketch of the analysis

Suppose we sample $\mathbf{x}_1, \ldots, \mathbf{x}_{n_0}$ according to $\mu_{\mathcal{X}}$: as soon as $n_0 \geq c_4 K \log K$ we have with high probability (w.h.p.) $\Omega(n_0 \mu_{\mathcal{X}}(\pi_k)) = \Omega(\frac{n_0}{K}) = \Omega(c_4)$ samples in each π_k (as in the coupon collector's problem, with the K planes representing coupons). If in addition we sample $n \sim c_5 n_0 d_{\max} \log d_{\max}$ points to form a random subset $\mathcal{X}_n$, with $c_5 = c_5(c_2, c_3, \sigma)$ large enough, then w.h.p. a large fraction of such points has a maximal scale $R_{\mathbf{x}}^* > \sigma$, which is large enough to contain $\Omega(d_{\max} \log d_{\max})$ points in $\mathcal{X}_n$. For any $\mathbf{x} \in \pi_k$ among such points, at appropriate scales $\sigma \lesssim r \leq R_{\mathbf{x}}^*$, with only $\Omega(d_k \log d_k)$ noisy samples it is possible to obtain an accurate empirical estimate of $\mathrm{cov}(X_{k,\mathbf{x},r})$ [11]. The MSVD analysis of such matrices, for varying values of r, yields accurate estimates $\hat{d}_k$ and $\hat{\pi}_k$ for d_k and π_k respectively. Moreover, by monitoring an empirical estimate of the fitting error of the data by the planes constructed as we increase n_0, with $O(n)$ points as above we produce a model with $O(K)$ planes, that, w.h.p., is accurate in the sense that $\frac{1}{n}\sum_{\mathbf{x}_i \in \mathcal{X}_n} \min_k \mathrm{dist}^2(\mathbf{x}_i, \hat{\pi}_k)^2 \sim \frac{1}{n}\sum_{\mathbf{x}_i \in \mathcal{X}_n} \min_k \mathrm{dist}^2(\mathbf{x}_i, \pi_k)$ and this error is w.h.p. as small as can be expected. At this point we use spectral clustering on a matrix of assignments of points to planes to estimate K, merge the estimated planes and (if requested to solve Problem 2) make the final point assignment. This reasoning can be made rigorous [3]:

Theorem 2.1. *With the above assumptions there exists* *an algorithm whose inputs are upper bounds $K_{\max}, d_{\max}$ for the number of planes and their dimensions, respectively, such that the following holds: by accessing $n \leq C d_{\max} K_{\max} \log K_{\max} \log d_{\max}$ samples distributed according to $\mu_{\mathcal{X}}$, where $C = C(c_1, c_2, \sigma)$ is a constant, it returns w.h.p. the correct model parameters $(K; d_1, ..., d_K)$, and accurate approximations to $\{\pi_k\}_{k=1}^K$, in time $O(D K_{\max} d_{\max}(d_{\max}+K_{\max}) \log K_{\max} \log d_{\max})$.*

The finite sample and running time guarantees are the strongest among the existing algorithms (most of which have in fact no guarantees), and near optimal. Because of space constraint we cannot provide here the fully quantitative result, nor the most general version of the above statement, nor its proof, and we focus instead on the algorithm itself, its implementation and performance, on synthetic and real data, relative to the current state of the art.

2.3. Multiscale SVD analysis for multiple planes

The first step of the algorithm is to estimate several local planes from random samples from the distribution $\mu_{\mathcal{X}}$. Our technique is based on the Multiscale Singular Value Decomposition (MSVD) [11], originally applied for estimating the intrinsic dimension of a point cloud. Singular values computed in increasingly larger neighborhoods (scales) of each point are used to identify a range of scales within which the singular values corresponding to the local tangent plane, curvature and noise exhibit different growth rates (as functions of scale). Here we only discuss our adaptation of MSVD to the case of plane arrangements, to yield a collection of good local pieces of the underlying planes, together with estimates of their intrinsic dimensions, best fitting planes and approximation errors.

We fix a sequence of $J = O(1)$ positive integers $n_j \sim \lceil j d_{\max} \log d_{\max} \rceil, j = 1, \ldots, J$. We draw a point $\mathbf{x}_i$ according to $\mu_{\mathcal{X}}$, say $\mathbf{x}_i \in \pi_k$, and for each j, let $\mathbf{y}_1, \ldots, \mathbf{y}_{n_j}$ be the n_j nearest neighbors of $\mathbf{x}_i$. We compute the top few singular values $\sigma_{jp}, p = 1, \ldots, d_{\max}$, of the $n_j + 1$ points

$$\mathbf{Y}_j := (n_j+1)^{-1/2} \left[\mathbf{x}_i - \mathbf{m}_j, \mathbf{y}_1 - \mathbf{m}_j, \ldots, \mathbf{y}_{n_j} - \mathbf{m}_j\right], \quad (1)$$

where $\mathbf{m}_j$ is the mean of $\mathbf{x}_i$ and the $\mathbf{y}$'s (points are thought of as column vectors). We let $r_j := \left\|\mathbf{x}_i - \mathbf{y}_{n_j}\right\|$, for $j = 1, \ldots, J$, be the local scales. The σ_{jp} and r_j depend on the particular point $\mathbf{x}_i$, but we have dropped this dependence from the notation. We will use the multiscale singular values $\{\sigma_{jp}\}_j, 1 \leq p \leq d_{\max}$ and the local scales $\{r_j\}_j$ to determine the local dimension at $\mathbf{x}_i$. In [3] we prove that w.h.p. for $\sigma < r_j < R^*_{\mathbf{x}_i}$ the top d_k singular values σ_{jp} grow linearly in r_j (as do their expected values, by our assumptions on $\mathrm{cov}(X_{k,\mathbf{x},r})$), while the remaining σ_{jp} are $O(\sigma)$ (by the assumptions on the noise η_i). At scales $r_j > R^*_{\mathbf{x}_i}$ at least σ_{j,d_k+1} also starts growing linearly, as intersections and points on other planes $\pi_{k'}$, $k' \neq k$, enter the neighborhood of $\mathbf{x}_i$. Algorithm 1 below implements this strategy to infer the local dimensions at a subset of n_0 randomly sampled points from $\mu_{\mathcal{X}}$; Figure 1 illustrates its behavior on a toy data set.

Algorithm 1 Linear-manifolds Multiscale SVD (LMSVD)

```
Input: Upper bound $d_{\max}$ for all $d_k$; sampling parameter $n_0$.
Output: $n_0$ sampled points $\mathbf{x}_i$, maximal good regions $\hat{\mathcal{R}}_i$, local dimensions $\hat{d}_i$, best $\hat{d}_i$-planes $\hat{\pi}_i$ approximating $\hat{\mathcal{R}}_i$ and the least squares errors $\hat{\epsilon}_i$.
  Steps:
    . Randomly sample $n_0$ points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_0}\}$ according to $\mu_{\mathcal{X}}$, let $n_j = j d_{\max} \log d_{\max}$, $1 \leq j \leq J = 50$, $\alpha_0 = 0.3/\sqrt{d_{\max}}$.
    . for $i = 1 : n_0$
        – For each $1 \leq j \leq J$, perform SVD analysis on $\mathbf{x}_i$ and its $n_j$ nearest neighbors using (1).
        – Detect and discard the first few scales $r_j$ where all the $\sigma_{jp}$ grow linearly with slope $\geq \alpha_0$ due to noise (this corresponds w.h.p. to $r_j < \sigma$).
        – Find the maximum $\hat{R}^*_{\mathbf{x}_i}$ of subsequent scales within which the first few $\sigma_{jp}$ grow linearly while the remaining ones are relatively flat, by thresholding slopes at $\alpha_0$.
        – Let $\hat{\mathcal{R}}_i$ be the subset containing $\mathbf{x}_i$ and its nearest neighbors that are within distance $\hat{R}^*_{\mathbf{x}_i}$, and $\hat{d}_i$, the number of the singular values with linear growth in the range, be the local dimension at $\mathbf{x}_i$.
        – Let $\hat{\pi}_i$ be the best $\hat{d}_i$-plane approximating $\hat{\mathcal{R}}_i$, and compute the corresponding least squares error $\hat{\epsilon}_i$.
      end
```

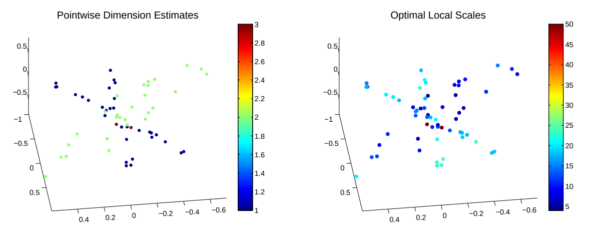
We estimate the model error by the quantity

$$\tau^2 = \frac{D}{n_0} \sum_{i=1}^{n_0} \frac{\hat{\epsilon}_i^2}{D - \hat{d}_i}, \quad (2)$$

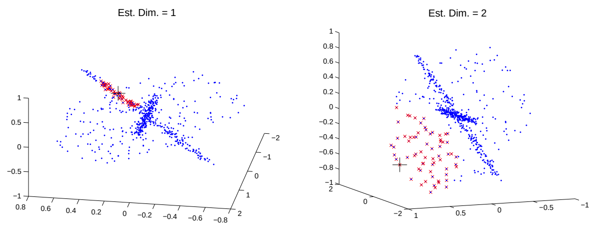
where $\hat{d}_i, \hat{\epsilon}_i$ are the local dimensions and errors returned by Algorithm 1. Observe that an unbiased estimator for the true error may be obtained by using a validation set of size no larger than the set of points used to estimate $\hat{\pi}_i$. τ is (w.h.p.) close to the expected error. It will be used in Section 2.4.2 to estimate K (when not given). We remark that many state-of-the-art algorithms such as ALC [13], SSC [5], and GPCA [14] crucially rely on the parameter τ, therein referred to as a "tolerance level". The ALC method when given the correct error tolerance estimates the right number of clusters K while clustering data, and otherwise fails to do so, in general, as we shall discuss later.

2.4. Plane alignment for modeling data

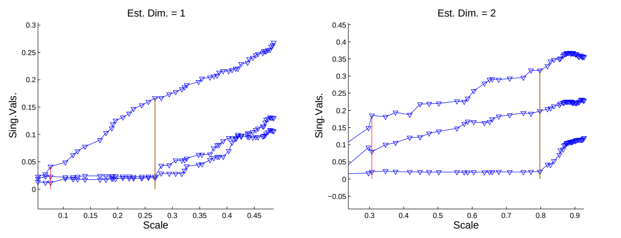
Given the upper bound dimension $d_{\max}$ and the sampling parameter n_0, Algorithm 1 returns n_0 random samples $\mathbf{x}_i$, their optimal local regions $\hat{\mathcal{R}}_i$ and associated parameters $(\hat{d}_i, \hat{\pi}_i, \hat{\epsilon}_i)$. In this section we present how to use these



(a) Dimension estimates of $n_0 = 60$ randomly chosen samples (left) and corresponding values of maximal good scale $R^*_{\mathbf{x}}$ (right).



(b) Two points (indicated by black + symbols) among the ones above, with corresponding dimension (title) and scale estimates (red regions).



(c) Plots of the multiscale singular values (for the two points above) and the detected local dimensions and good scales (between red lines)

Figure 1. Illustration of Algorithm 1 (with $n_0 = 60$) on a data set of two lines and one plane in $\mathbb{R}^3$, each containing 200 points and corrupted with Gaussian noise with $\sigma = .04$.

information to recover the hybrid linear model and correspondingly partition the data in $\mathcal{X}$. We first assume that the number of subspaces K is given to us (Section 2.4.1), and then discuss (in Section 2.4.2) how to estimate K when it is unknown (using the tolerance of (2)). Finally, we formulate an algorithm (Section 2.4.3).

2.4.1 When the number of clusters K is given

We set $n_0 = \Omega(K \log K)$: we can show that (w.h.p.) a constant fraction of the n_0 planes $\hat{\pi}_i$ approximates well one of the true planes π_k. Define $\mathcal{X}_n = \bigcup_{1 \le i \le n_0} \hat{\mathcal{R}}_i \subseteq \mathcal{X}$, the collection of all samples and their nearest neighbors within the optimal scales. We recall that the cardinality n of $\mathcal{X}_n$ is $O(n_0 d_{\max} \log d_{\max})$. This is the data that we will use, together with the planes $\hat{\pi}_i$, to estimate the model.

We define an $n \times n_0$ affinity matrix

$$\mathbf{A}_{ij} := e^{-\frac{\mathrm{dist}^2(\mathbf{x}_i, \hat{\pi}_j)}{2\hat{\epsilon}_j^2}}. \tag{3}$$

Note that $\hat{\epsilon}_j$, the local error estimate, serves as a tuning parameter in the Gaussian kernel, and is locally adaptive. Also, each row $\mathbf{A}_{i:}$ maps the point $\mathbf{x}_i$ to a feature vector in $\mathbb{R}^{n_0}$ encoding the distances from $\mathbf{x}_i$ to the planes $\hat{\pi}_j$. We expect points generated in the same plane to be clumped together in the feature space, and conversely points from different planes in almost orthogonal directions; this is exemplified in the top left plot of Figure 2.

We next follow the corresponding steps of the SCC algorithm [2] to partition the data in $\mathcal{X}_n$ into K subsets respecting the model. That is, after proper normalization of the matrix $\mathbf{A}$, we extract its top K left singular vectors and use them as columns to form a matrix $\mathbf{U}$. The rows of $\mathbf{U}$ (regarded as points in $\mathbb{R}^K$), again properly normalized, are used as new coordinates of the data in $\mathcal{X}_n$. We then apply the K-means algorithm to these rows and obtain K clusters $\mathcal{X}_k$ of $\mathcal{X}_n$, We use the $\mathcal{X}_k$'s to provide updated estimates for the model parameters. For example the intrinsic dimension $\hat{d}_k$ of π_k is estimated through an internal voting procedure, i.e., it is set to be the most frequent dimension of the samples $\mathbf{x}_i$ that are assigned to this cluster. Finally, we let $\hat{\pi}_k$ be the $\hat{d}_k$ dimensional PCA plane of the cluster $\mathcal{X}_k$, and partition the original data in $\mathcal{X}$ by assigning points to their closest planes $\hat{\pi}_k$.

2.4.2 When the number of clusters K is unknown

When we do not know K, we will apply the strategy in the preceding section to the data in $\mathcal{X}_n$ (but with $n_0 = \Omega(K_{\max} \log K_{\max})$), for $K = k$ starting at $k = 1$ and increasing it by one at every iteration (or, better, by binary search). In each iteration we compute the corresponding k planes $\hat{\pi}_{jk}, 1 \le j \le k$ of dimensions $\hat{d}_{jk}$ approximating the clusters $\hat{\mathcal{X}}_{jk}$, and error

$$e^2(k) = \frac{D}{n} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \hat{\mathcal{X}}_{jk}} \frac{\mathrm{dist}^2(\mathbf{x}_i, \hat{\pi}_{jk})}{D - \hat{d}_{jk}}. \tag{4}$$

We gradually increase k until we find the first k so that $e(k)$ is below the tolerance τ estimated in (2): $\hat{K} = \min\{k : e(k) \le \tau\}$. This $\hat{K}$ is expected to be the true number of clusters: for $k < \hat{K}$, the corresponding k-plane model cannot accurately fit the data because their dimensions are chosen among local dimension estimates, thus always underfitting the data and leading to large errors $e(k)$. Note also that we never need to check those $k > K$.

2.4.3 The Multiscale Analysis of Plane Arrangements (MAPA) algorithm

We present in Algorithm 2 our solution to Problems 1 and 2 and illustrate it in Figure 2 on the data of Figure 1.

Algorithm 2 MAPA

```
Input: Upper bounds $d_{\max}, K_{\max}$.
Output: Model parameters $(\hat{K}; \hat{d}_1, \ldots, \hat{d}_{\hat{K}}; \hat{\pi}_1, \ldots, \hat{\pi}_{\hat{K}})$, and associated clustering.
  Steps:
    . Replace the original data $\mathcal{X}$ with a random subset, which with abuse of notation we still denote with $\mathcal{X}$, of size $\min\{100 K_{\max} d_{\max} \log K_{\max} \log d_{\max}, N\}$.
    . Apply Algorithm 1 to obtain $n_0 := 20\, K_{\max} \log K_{\max}$ local regions $\hat{\mathcal{R}}_i$ and their associated statistics $(\hat{d}_i, \hat{\pi}_i, \hat{\epsilon}_i)$. Also, compute the tolerance $\tau$ in (2).
    . Set $k = 1$ and $d = \mathrm{mode}\{\hat{d}_i \mid 1 \leq i \leq n_0\}$. Compute $d$-dimensional PCA plane for $\mathcal{X}$; let $e(1)$ be as in (4).
    . If $e(1) \leq \tau$, stop and return; otherwise form $\mathbf{A}$ as in (3) and normalize it to $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}$, where $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{A}'\mathbf{1})$. Let $\mathbf{U} := [\mathbf{u}_1, \ldots, \mathbf{u}_{K_{\max}}]$ the matrix of the top left singular vectors of $\mathbf{L}$.
    . while $e(k) > \tau$
        – Increment $k$ by 1 and let $\mathbf{U}_k = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$.
        – Normalize the row vectors of $\mathbf{U}_k$ to have unit length to produce the matrix $\mathbf{V}_k$, and apply $K$-means to the row vectors of $\mathbf{V}_k$ to find $k$ clusters $\{\hat{\mathcal{X}}_{jk}\}_{j=1}^{k}$.
        – Let the dimension $\hat{d}_{jk}$ of $\hat{\mathcal{X}}_{jk}$ be the mode number of the $\hat{d}_i$'s of the sampled points assigned to $\hat{\mathcal{X}}_{jk}$.
        – Compute the best fitting planes $\hat{\pi}_{jk}$ of clusters $\hat{\mathcal{X}}_{jk}$ and approximation error $e(k)$ using (4).
      end
    . Return $\hat{K} = k, \hat{d}_j = \hat{d}_{jk}, \hat{\pi}_j = \hat{\pi}_{jk}$. If also solving Problem 2, use $\hat{\pi}_j$ to cluster the original data in $\mathcal{X}$ by assigning points to their nearest planes.
```

2.5. Algorithmic complexity

For the solution of Problem 1, we assume the algorithm has access, at the cost $O(D)$, to any data point, and do not include the data storage in the space requirements. We also drop $\log$ factors in the following calculation. The space requirement then is $O(K_{\max}D)$, driven by the cost of storing the $n_0 = \Omega(K_{\max})$ estimated planes. The total computational cost of the algorithm is $O(K_{\max}d_{\max}(d_{\max} + K_{\max})D)$; in particular, it is independent of N, and is only marginally higher than $O(Kd_{\max}D)$, the cost of just encoding the planes. It is computed as follows: if we let $n = O(K_{\max}d_{\max})$, then in time $O(n_0 nD)$ we may compute the distances from n points to n_0 points, in time $O(n_0(d_{\max} + d_{\max}^2 D))$ we can find the $O(d_{\max})$ nearest neighbors of each of the n_0 points and perform MSVD, in time $O(nn_0 d_{\max}D)$ we can construct $\mathbf{A}$ and in time $O(nn_0K_{\max})$ we can compute $\mathbf{U}$, and in time $O(K_{\max}nD)$ we may compute $K_{\max}$-means (as in [2]). In order to solve Problem 2, we simply add the cost of assigning points to planes, which is $O(NKd_{\max}D)$.

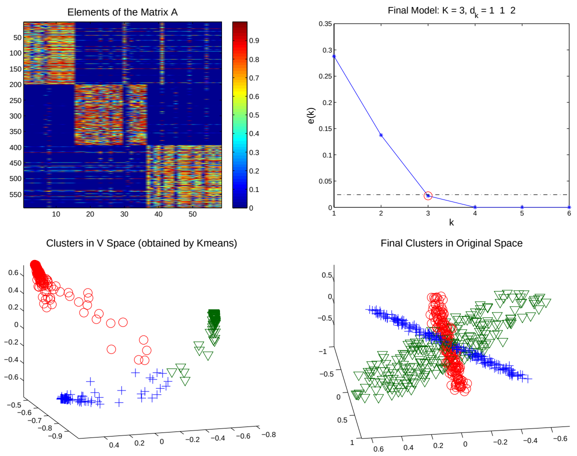


Figure 2. Illustration of Algorithm 2 on the data of Figure 1. Note that the errors $e(k) = 0$ when $k > 3$, indicating that MAPA stopped at $k = 3$ and thus did not examine those larger k.

3. Experiments

We extensively test the MAPA algorithm on simulated and real data and compare it with ALC [13], GPCA-voting [14], SCC [2], SSC [5], LRR [12], and LBF [21]. Among these algorithms, only ALC estimates a model given a tolerance level (though it is not completely clear how ALC would infer the subspace dimensions once the clusters are formed).[1] The LBF algorithm estimates the number of clusters only when all d_k are equal and known. Therefore, for each simulated or real data set, we will compare MAPA only with ALC and LBF (for the latter we input the maximum of the dimensions when not all equal) in terms of the K-*modeling error*, i.e., frequency of incorrect identification of the model parameter K, while reporting the d_k-*modeling error* for MAPA alone, i.e., frequency of incorrect identification of the dimensions $d_1, \ldots, d_K$. (Note that the d_k-modeling error is no smaller than the K-modeling error.) In the meantime, we also report the *clustering error* and *running time* of all algorithms (run on Core 2 Duo 8400 3.0Ghz machines with 4GB of RAM). The Matlab code of MAPA, together with links to the other algorithms' webpages, can be found at http://www.math.duke.edu/~mauro/code.html.

3.1. Simulations

We generate many instances of artificial data using code from the GPCA-voting package. We denote a collection of planes of dimensions $d_1, \ldots, d_K$ in $\mathbb{R}^D$ by $(d_1, \ldots, d_K; D)$. We will test MAPA against other methods in the following six instances: $(1, 1, 2; 3)$, $(1, 2, 2; 3)$, $(2, 2, 2; 3)$, $(1, 1, 2, 2; 3)$, $(1, 2, 3; 4)$, and $(1, 1, 3, 3; 6)$.

[1]The paper suggests to threshold the singular values of the clusters in order to estimate their dimensions, however, neither it nor the Matlab code provides any further detail regarding how to implement it in a robust way.

		$(1,1,2;3)$		$(1,2,2;3)$		$(2,2,2;3)$		$(1,2,1,2;3)$		$(1,2,3;4)$		$(1,1,3,3;6)$	
		L	A	L	A	L	A	L	A	L	A	L	A
d_k	MAPA	0	0	0	0	0	0	1	0	0	0	7	0
K	MAPA	0	0	0	0	0	0	0	0	0	0	0	0
	ALC	0	6	0	15	0	20	0	24	0	1	0	0
	LBF	81	23	0	2	1	0	77	15	71	19	72	30
$e_{\%}$	*REF*	$3.2_{\pm 0.8}$	$1.9_{\pm 1.2}$	$3.8_{\pm 0.9}$	$3.1_{\pm 1.6}$	$5.5_{\pm 1.2}$	$3.6_{\pm 1.3}$	$4.6_{\pm 0.8}$	$3.6_{\pm 1.5}$	$3.3_{\pm 0.8}$	$2.1_{\pm 1.1}$	$2.5_{\pm 0.5}$	$0.0_{\pm 0.0}$
	MAPA	$3.2_{\pm 0.7}$	$1.9_{\pm 1.1}$	$3.8_{\pm 0.8}$	$3.0_{\pm 1.6}$	$5.6_{\pm 1.2}$	$3.6_{\pm 1.3}$	$4.8_{\pm 0.8}$	$3.7_{\pm 1.4}$	$3.4_{\pm 0.8}$	$2.0_{\pm 1.2}$	$2.6_{\pm 1.1}$	$0.0_{\pm 0.0}$
	ALC	$2.0_{\pm 0.7}$	$1.3_{\pm 5.4}$	$3.2_{\pm 1.2}$	$4.4_{\pm 10.0}$	$8.3_{\pm 2.0}$	$7.6_{\pm 12.6}$	$4.0_{\pm 1.2}$	$7.4_{\pm 12.7}$	$0.5_{\pm 0.3}$	$0.1_{\pm 0.2}$	$0.8_{\pm 0.3}$	$0.0_{\pm 0.0}$
	SCC	$3.5_{\pm 1.0}$	$0.8_{\pm 0.8}$	$4.0_{\pm 1.1}$	$2.4_{\pm 1.6}$	$5.6_{\pm 1.1}$	$3.5_{\pm 1.6}$	$6.9_{\pm 5.6}$	$2.4_{\pm 2.1}$	$2.1_{\pm 0.8}$	$0.3_{\pm 0.4}$	$25_{\pm 17.9}$	$0.0_{\pm 0.0}$
	GPCA	$6.0_{\pm 9.4}$	$5.4_{\pm 9.8}$	$4.0_{\pm 1.0}$	$9.6_{\pm 9.9}$	$5.7_{\pm 1.1}$	$5.1_{\pm 2.8}$	$11.8_{\pm 12.9}$	$24_{\pm 13.9}$	$4.2_{\pm 4.9}$	$5.2_{\pm 7.8}$	$19.9_{\pm 14.8}$	$0.0_{\pm 0.0}$
	LBF	$33_{\pm 3.6}$	$11.0_{\pm 13.1}$	$5.6_{\pm 1.2}$	$4.8_{\pm 6.6}$	$6.0_{\pm 4.6}$	$3.8_{\pm 1.5}$	$27_{\pm 3.8}$	$9.2_{\pm 8.5}$	$31_{\pm 7.2}$	$10.0_{\pm 12.5}$	$19.5_{\pm 10.3}$	$7.5_{\pm 11.5}$
	SSC	$29_{\pm 16.6}$	$23_{\pm 13.6}$	$28_{\pm 12.2}$	$34_{\pm 13.1}$	$35_{\pm 11.4}$	$40_{\pm 10.8}$	$40_{\pm 12.1}$	$39_{\pm 11.2}$	$17.5_{\pm 11.5}$	$15.4_{\pm 12.2}$	$4.2_{\pm 8.0}$	$1.1_{\pm 1.7}$
	LRR	$39_{\pm 5.6}$	$27_{\pm 11.3}$	$48_{\pm 5.4}$	$34_{\pm 12.2}$	$55_{\pm 3.9}$	$34_{\pm 11.7}$	$53_{\pm 3.3}$	$39_{\pm 8.1}$	$39_{\pm 7.0}$	$23_{\pm 12.7}$	$18.8_{\pm 5.4}$	$19.5_{\pm 13.0}$
t	MAPA	.69	.72	.64	.68	.60	.64	0.97	1.02	.70	.77	1.26	1.53
	ALC	15.4	27.1	15.2	30.2	14.6	33.8	29.3	57.7	16.4	40.2	34.2	83.8
	SCC	1.42	1.19	2.08	1.94	2.74	2.56	4.45	4.06	3.26	3.04	5.12	5.04
	GPCA	3.44	4.18	3.74	7.36	1.39	2.04	5.97	11.65	12.0	12.9	32.1	34.6
	LBF	10.5	11.5	10.4	11.3	17.4	19.3	12.2	13.5	17.9	19.4	25.2	27.4
	SSC	167	120	171	126	165	124	251	180	179	134	298	222
	LRR	240	236	272	267	288	284	694	684	290	286	731	678

Table 1. Comparison among various algorithms in six instances of hybrid linear modeling, including both linear (L) and affine (A) data. The experiment in each instance is repeated 100 times, so that the means for $e_{\%}$ (in percentage), t (in seconds) and standard deviation for $e_{\%}$, as well as modeling error rates for K, d_k, can be reported. The MAPA algorithm is applied using $n_0 = 20 \cdot K$. All the other algorithms are given the truth for the necessary model and tuning parameters. The clustering errors of all algorithms are compared with a reference (REF) algorithm which directly assigns points to the nearest ground-truth planes.

In each instance $(d_1, \ldots, d_K; D)$, we first randomly generate a collection of K linear subspaces of dimensions $d_1, \ldots, d_K$ in $\mathbb{R}^D$. We then randomly draw 200 samples from each subspace, and corrupt them with 4% Gaussian white noise. We refer to such data as *linear data*, as opposed to *affine data*, i.e., data sampled from affine subspaces, generated by translating each cluster of the linear data by a random vector $\mathbf{c} \in \mathbb{R}^D$, with $\mathbf{c} \sim N(0, I_D)$.

We apply the MAPA algorithm to such linear and affine data with $n_0 = 20\,K$. We examine both the K-modeling and d_k-modeling errors, and in addition, for the estimates K, d_k and an associated clustering of the data, compute a clustering error $e_{\%}$ (i.e., percentage of misclassified points). We also apply the ALC algorithm: in order to ensure the best possible results for ALC, we (a) set its tolerance parameter equal to the *true* model error; (b) use the optimal coding length function depending on whether the planes are linear or affine. For LBF, we use $\max_k d_k$ as dimension of the planes, and as above provide the linear/affine information. We use the default values for the other parameters in LBF, in particular, $K_{\max} = 10$.

We repeat the above experiment, for the three algorithms, 100 times and record in Table 1 the modeling errors, the average clustering errors (with standard deviation), and the running times. Observe that both MAPA and ALC achieve excellent results in all instances, while LBF performs well only in two cases (where the dimensions are the same, or can be regarded the same). It seems that ALC always does worse in the affine case, which is the easier case for MAPA (due to better separation). A significant advantage of MAPA is its fast speed (at least 20 times faster than ALC with a single tolerance), parallelizability, and of course the fact that it can solve Problem 1 much faster than Problem 2. MAPA is essentially parameter free (it only requires $K_{\max}, d_{\max}$); in contrast, ALC has the tolerance level as a crucial parameter, possibly hard to acquire in most practical applications.

Meanwhile, we report in the same table the clustering errors and running times of SCC, GPCA, SSC and LRR on the same data generated above. We give all the necessary true model parameters to each of these algorithms. We set the sampling parameter $c = 100K$ in SCC and the tuning parameter $\lambda = 0.01$ in LRR, as used in the corresponding papers. The SSC algorithm also contains an important tolerance parameter for which we provide the true model error. We use the different versions of the SSC code to deal with linear and affine data accordingly.

Observe that SCC also achieves excellent clustering results in all scenarios, with a relatively fast speed. The SSC algorithm only works well in the last case where the ambient dimension D is relatively high, so that the subspaces are nearly independent of each other, a known necessary assumption for SSC. Also, note that SSC and LRR are the slowest methods (several hundred seconds), primarily due to the need to solve sparse coding problems per point.

3.2. Applications to real data

The algorithm could be applied to problems wherever one needs to model data using a union of subspaces. However, due to page limit, we only study two applications, *motion segmentation* and *face clustering*, in this paper.

3.2.1 Motion segmentation with affine camera models

We investigate the motion segmentation problem already studied in a few papers [2, 14, 15]. Suppose that a camera is tracking a dynamic scene and captures a video sequence of F frames of the scene, and also that N feature points have been selected from the moving objects. Then, given only the trajectories of the feature points along the sequence, the problem is to segment the different motions. [15, 14] show that this is a subspace clustering problem, since under affine camera models the trajectory vectors corresponding to different moving objects across F image frames live in distinct linear subspaces of dimension at most 4 in $\mathbb{R}^{2F}$, or affine subspaces of dimension at most 3 within those linear subspaces. The bounds 4 (linear) and 3 (affine) have been used by the algorithms [2, 14] as the common dimension of the subspaces in this setting.

However, the dimensions of the planes are still not precisely known. Moreover, the number of motions K is revealed as input to those algorithms. Here, we use MAPA to infer the best model for the motion data, in addition to clustering the motions. We use as examples the three Kanatani sequences that are originally produced in [15] and studied in [2, 14]. These three data sets are also part of a large database of 155 video sequences [17], but due to page limit we will not test our algorithm on this extensive dataset. As a preprocessing step to suppress noise, we apply PCA to project the data onto $\mathbb{R}^{10}$ in which we then test our method against the ALC algorithm.

We first apply MAPA to the projected data with $n_0 = N$ in order to avoid randomness: the results are reported in Figure 3. The number of motions K is correctly identified, the underlying planes are determined to all have dimension 2, and the clusters are also perfectly recovered (up to a few errors in the third example). The model errors are estimated to be $.0507, .0405, .2391$ using (2) for the three sequences, respectively. These numbers will be supplied to ALC as its tolerance levels in the experiment below.

We next apply the ALC algorithm to the three sequences and compare modeling and clustering errors. When ALC is provided with the tolerance parameter estimated by MAPA, it produces the correct K and has zero clustering error for sequences 1 and 3, while overestimating K for sequence 2. We also test a few other values to further study the sensitivity of the tolerance parameter. We find that for sequence 2 ALC returns $K = 3$ whenever $\tau \leq 0.0560$, but $K = 2$ when $\tau \geq 0.0565$ (see Figure 3); for sequences 1 and 3, $\tau = .001, .01$, respectively, lead to incorrect K.

3.2.2 Clustering of facial images

We next consider the problem of clustering a collection of images of human faces in fixed pose under varying illumination conditions. A well-known such dataset is the Yale Face Database B [7]. In general we know that for a Lambertian object the set of all its images under a variety of lighting conditions approximately span a low-dimensional linear subspace [9]. Moreover, images of different objects lie in different subspaces, so that this problem may be tackled by segmenting an arrangement of linear subspaces.

We use the frontal face images of all ten human subjects in the Yale database. Overall, there are 640 images (64 images per subject), of size 640×480. We would like to separate these images into 10 groups, one per subject. We apply the same preprocessing as in [14], i.e., downsample the images to 160×120 and apply SVD to project them into $\mathbb{R}^{30}$. We apply both MAPA and ALC on the preprocessed images to estimate the number of subjects in the collection and cluster the facial images. For MAPA, we set $n_0 = N$ to avoid randomness. It correctly identifies 10 groups and obtains a zero clustering error. In addition, the dimensions of the planes are estimated to be a mixture of 2 and 3: $d_k = 3, 3, 3, 3, 3, 2, 3, 2, 3, 3$. These results are summarized in Figure 4. Also, MAPA estimates that $\tau = 1119$ which is supplied to ALC as tolerance, in which case ALC also correctly identifies $K = 10$ together with zero clustering error. However, with another tolerance $\tau = 100$ ALC returns 11 clusters.

4. Conclusions

We presented an efficient and effective algorithm for estimating plane arrangements. It starts by finding many local pieces of the underlying clusters, via a *multiscale* approach, and then aligns their best approximating planes, using a *spectral* approach, to recover the plane arrangement. It has a computational complexity essentially comparable to that of encoding the answer to the problems; in particular it is independent of the number of points when solving Problem 1, and it estimates the parameters of the model rather than requiring them as inputs. The algorithm gives state-of-the-art results when compared with the current best algorithms, on both synthetic and real data, and is faster.

Acknowledgements

GC thanks Gilad Lerman for forwarding him preprints and codes of his work [21] and for useful discussions, in particular, for pointing out the use of the point-to-plane distance in the affinity of (3). The authors are grateful for the support from NSF 0650413, 0808847 and 0803293, and ONR N00014-07-1-0625.

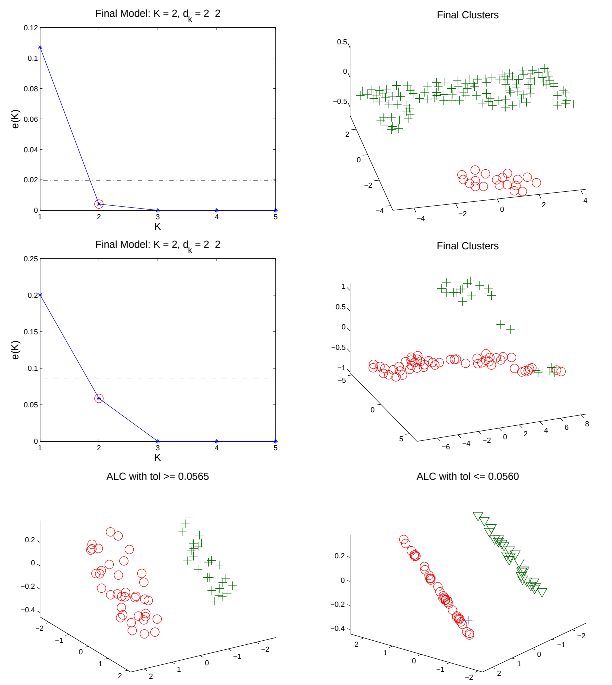


Figure 3. Results obtained by MAPA on sequences 1 and 3 (first two rows) and by ALC on sequence 2 (last row) of the Kanatani dataset. MAPA achieves perfect result on sequence 2 (not shown).

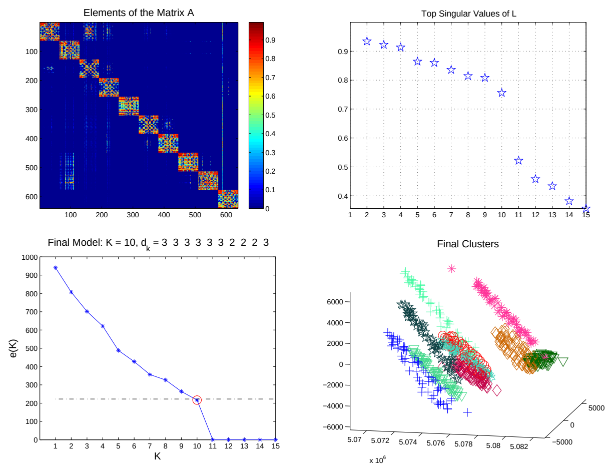


Figure 4. Results obtained by MAPA (with $n_0 = N$) on all 10 subjects in the Yale database using only their frontal face images. MAPA determined the best model to be 10 planes of mixed dimensions 2 and 3, and achieved zero clustering error.

References

[1] G. Chen and G. Lerman. Foundations of a multi-way spectral clustering framework for hybrid linear modeling. *Found. Comput. Math.*, 9(5):517–558, 2009. 2825

[2] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *Int. J. Comput. Vision*, 81(3):317–330, 2009. 2828, 2829, 2831

[3] G. Chen and M. Maggioni. Multiscale geometric methods for data sets III: Multiple planes. *In preparation*. 2826, 2827

[4] G. Chen, A.V. Little, M. Maggioni and L. Rosasco. Some recent advances in multiscale geometric analysis of point clouds, In *Wavelets and Multiscale Analysis: Theory and Applications*, 2011, submitted 3/12/2010. 2826

[5] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR*, pages 2790–2797, June 2009. 2825, 2827, 2829

[6] M. Fischler, and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981. 2825

[7] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. PAMI*, 23(6):643–660, 2001. 2831

[8] A. Goh, and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *CVPR*, pages 1–6, Minneapolis, MN, June 2007. 2825

[9] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *CVPR*, volume 1, pages 11–18, 2003. 2825, 2831

[10] W. Hong, J. Wright, K. Huang, and Y. Ma. A multi-scale hybrid linear model for lossy image representation. In *ICCV*, pages 764–771, 2005. 2825

[11] A. Little, Y. Jung, and M. Maggioni. Multiscale estimation of intrinsic dimensionality of data sets. In *AAAI Fall Symposium Series*, November 2009. 2825, 2826, 2827

[12] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *ICML*, pages 663–670, 2010. 2825, 2829

[13] Y. Ma, H. Derksen, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *PAMI*, 29(9):1–17, 2007. 2825, 2827, 2829

[14] 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. 2825, 2827, 2829, 2831

[15] Y. Sugaya and K. Kanatani. Multi-stage unsupervised learning for multi-body motion segmentation. *IEICE Trans. Inf. & Syst.*, E87-D(7):1935–1942, 2004. 2825, 2831

[16] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999. 2825

[17] R. Tron and R. Vidal. A benchmark for the comparison of 3-D motion segmentation algorithms. In *CVPR*, volume 1, pages 1–8, June 2007. 2831

[18] R. Vidal. Subspace Clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, March 2011. 2825

[19] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *PAMI*, 27(12):621–628, 2005. 2825

[20] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In *ECCV*, 4:94–106, 2006. 2825

[21] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Randomized hybrid linear modeling by local best-fit flats. In *CVPR*, pages 1927–1934, June 2010. 2825, 2826, 2829, 2831