A New Approach To Two-View Motion Segmentation Using Global Dimension Minimization

Bryan Poling · Gilad Lerman

Received: date / Accepted: date

Abstract We present a new approach to rigid-body motion segmentation from two views. We use a previously developed nonlinear embedding of two-view point correspondences into a 9-dimensional space and identify the different motions by segmenting lower-dimensional subspaces. In order to overcome nonuniform distributions along the subspaces, whose dimensions are unknown, we suggest the novel concept of global dimension and its minimization for clustering subspaces with some theoretical motivation. We propose a fast projected gradient algorithm for minimizing global dimension and thus segmenting motions from 2-views. We develop an outlier detection framework around the proposed method, and we present state-of-the-art results on outlier-free and outlier-corrupted two-view data for segmenting motion.

Keywords Global Dimension · Empirical Dimension · Subspace Clustering · Hybrid-Linear Modeling · Motion Segmentation · Outliers · Robust Statistics

Supp. webpage: http://math.umn.edu/~lerman/gdm

1 Introduction

A classic problem in computer vision is that of feature-based motion segmentation from two views. In this problem one has two images, taken at different times, of a 3D scene. The scene is assumed to consist of multiple, independently moving rigid bodies. The goal is to identify the different moving objects and estimate a motion model for each one of them. For this purpose, one automatically tracks the locations of visually-interesting “features” in the scene, which are visible in both views (e.g., via Lucas Kanade type algorithm [4]). Each feature is represented as a pair of 2-vectors, holding the image coordinates of the feature in the two different views; such a pair is referred to as a point correspondence. The mathematical problem of feature-based, two-view motion segmentation is to both segment the point correspondences according to the rigid objects to which they belong, and estimate a motion model for each object.

A basic strategy to solve the feature-based, two-view motion segmentation problem is to first cluster point correspondences and then estimate the single-body motions within clusters (well-known methods for single-body motion estimation are described in [22,30]). This procedure was suggested in [17], while clustering point correspondences with K-means or spectral clustering, and in [40], while alternating between clustering and motion segmentation via an EM procedure. Both clustering strategies of [17] and [40] are based primarily on spatial separation between the clusters, however, different clusters in this setting may intersect each other (e.g., when motions share a symmetry).

Due to this problem, some algebraic methods have been developed for directly solving for the motion parameters, while eliminating the clustering of point correspondences [46,55]. Another solution is to segment feature trajectories by taking into account their geometric structures, which may be different than spatial separation (the feature trajectories in 2-views are the 4-dimensional vectors concatenating the 2 point correspondences of the same feature from 2 views).

Costeira and Kanade [14] showed that under the affine camera model, feature trajectories in n-views (for n ≥ 2
these are vectors of length $2n$) within each rigid body lie on an affine subspace of dimension at most 3. This observation has given rise to several feature-based segmentation schemes for n-views, which are based on clustering subspaces; we refer to such clustering as Hybrid Linear Modeling (HLM). Many algorithms have been suggested for solving the HLM problem, for example, the K-flats (KF) algorithm or any of its variants \cite{9,19,23,49}, methods based on direct matrix factorization \cite{8,14,24,25}, Generalized Principal Component Analysis (GPCA) \cite{45,32,33}, Local Subspace Affinity (LSA) \cite{47}, RANSAC (for HLM) \cite{48}, Agglomerative Lossy Compression (ALC) \cite{31}, Spectral Curvature Clustering (SCC) \cite{12}, Sparse Subspace Clustering (SSC) \cite{15,16}, Local Best-Fit Flats (LBF and its spectral version SLBF) \cite{50,51} and Low-rank Representation (LRR) \cite{29,28}. Some theoretical guarantees for particular HLM algorithms appear in \cite{11,26,37,38,3,2}. Two recent reviews on HLM are by Vidal \cite{44} and Aldroubi \cite{1}.

For the more general and realistic model of the perspective camera, it can be shown that feature trajectories from two-views lie on quadratic surfaces of dimension at most 3 (in $\mathbb{R}^4$) (see $\S$ 2). Arias-Castro et al. \cite{2} suggested clustering the quadratic surfaces of point correspondences (in $\mathbb{R}^4$) using Higher Order Spectral Clustering (HOSC) for manifold clustering. They demonstrated competitive results on the outlier-free database of \cite{35}, when assuming that the clusters are of dimension 2. However, their results are not competitive for incorporating dimension 3 and they did not provide any numerical evidence that the dimension of the surfaces was 2 and not 3.

A different approach for clustering these particular quadratic surfaces can be obtained by embedding point correspondences into “quadratic coordinates” and then clustering subspaces. More precisely, if a point correspondence $(x, y, (x', y'))$ is mapped into $(x, y, \mathbb{R}^d)$, where $\mathbb{R}^d$ denotes the Kronecker product, then these quadratic surfaces are mapped into linear subspaces of dimensions at most 8, which are determined by the fundamental matrices \cite{22,30,10} of the different motions. Chen et al. \cite{10} have used this idea for clustering such quadratic mappings of point correspondences by the Spectral Curvature Clustering (SCC) algorithm \cite{12} (they showed that instead of performing the actual mapping, one can apply the kernel trick). They claimed that other HLM algorithms (that at that time) did not work well for such embedded data.

The drawback of applying SCC to this quadratic mapping of point correspondences in $\mathbb{R}^9$ is that SCC does not work well with subspaces of mixed dimensions, and the subspace dimensions must be known a-priori. Unfortunately, the subspaces in this application have mixed and unknown dimensions (see $\S$ 2). What makes SCC successful for this application is the fact that it takes into account some global information of the subspaces (i.e., for d-dimensional subspaces it uses affinities based on arbitrary $d + 2$ points, and in particular, far-away points). This helps SCC deal with nonuniform sampling along subspaces with local structure very different than the global one (see $\S$ 2). On the other hand, local methods (e.g., \cite{51,13}) often do not work well in this setting.

The purpose of this paper is to develop an HLM algorithm that can successfully cluster the quadratically-embedded point correspondences in $\mathbb{R}^9$. In particular, it exploits the global structure of the underlying subspaces, i.e., their “dimensions”. We remark that earlier works \cite{5,15,20,21} used dimension estimators to segment data clusters according to their intrinsic dimension \cite{5} and \cite{18} used box counting estimators of some fractal dimensions and \cite{20,21} used the statistical estimator of \cite{27}. However, their methods do not distinguish well subspaces of the same dimension. Here on the other hand, we aim to minimize “dimensions” within tentative clusters, instead of estimating them. Thus, we take into account the effect of tentative clusters on these “dimensions” and try to optimize accordingly the appropriate choice of clusters.

For this purpose, we propose a class of empirical dimension estimators, and a corresponding notion of global dimension for a mixture of subspaces (a function of the estimated dimensions of its constituent parts). We propose the global dimension minimization (GDM) algorithm, which is a fast projected gradient method aiming to minimize the global dimension among all data partitions. We also build an outlier detection framework into this development to allow for corrupted data sets. We demonstrate state-of-the-art results for two-view motion segmentation (via quadratic embedding), both in the outlier-free and outlier-corrupted cases. We even show that these results are competitive with the state-of-the-art results for multiple-views, i.e., using all frames of a video sequence (obtained under the affine camera model). To motivate the use of global dimension, we prove that for special settings and choice of parameters, the global dimension is minimized by the correct partition of the data (representing the underlying subspaces). We then discuss what to do in more general settings.

The paper is organized as follows: $\S$ 3 briefly explains how the problem of 2-view motion segmentation can be formulated as a problem in HLM; $\S$ 4 introduces global dimension and explains why its minimization can solve the HLM problem under some conditions; $\S$ 5 develops a fast projected gradient method for minimizing global dimension; $\S$ 6 develops an outlier detection/rejection framework for global dimension minimization; $\S$ 7 demonstrates numerical results on real-world 2-view data sets for both outlier-removed and outlier-corrupted data; finally, $\S$ 8 concludes this work. The appendix contains proofs of the key results in the paper.
2 Formulating 2-View Motion Segmentation as a Problem in HLM

One way of formulating the motion segmentation problem in terms of HLM is by exploiting the Affine Motion Subspace. Costa and Kanade [14] demonstrated that when a set of features all come from a single rigid body, then under the assumptions of the affine camera model, the corresponding feature trajectories lie in an affine subspace of dimension 3 or less. One can use this fact to partition the set of features by clustering their trajectories into subspaces. This is a popular formulation of the segmentation problem, even when dealing with only two views of a scene.

The formulation involving the affine motion subspace has the advantage that the feature trajectories tend to be nicely distributed in their respective subspaces, and the different subspaces all have nearly the same dimensions. This formulation has the drawback that it requires an affine camera model. The affine camera assumption breaks down when viewing objects close to the camera, or when looking at objects at significantly different ranges. The consequence of this is that the trajectories from a rigid body do not lie within a subspace, but rather in a manifold which is only locally approximated by a subspace of dimension at most 3.

When dealing with 2-view segmentation, there is another approach, based on a more general camera model, which avoids this problem of distortion. This approach assumes a perspective camera, and relies on the fundamental matrix [22] for a rigid body.

Indeed, if $F = (F_{i,j})_{i,j=1}^3$ is the fundamental matrix for a rigid body, and $x_h = (x, y, 1)^T$ and $x'_h = (x', y', 1)^T$ together form a point correspondence (in standard homogenous coordinates) from that body, then

$$x'_h^T F x_h = 0,$$

which is algebraically equivalent to

$$\text{vec}(F) \cdot \nu = 0,$$  \hspace{1cm} (1)

where

$$\text{vec}(F) = (F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33})^T$$

and

$$\nu = (x'x', x'y', x'x, y'y, y'y', x, y, 1)^T = (x, y, 1)^T \otimes (x', y', 1)^T.$$

We refer to the vector $\nu$ as the nonlinear or Kronecker embedding of a point correspondence (recall that $\otimes$ is the Kronecker product). The vectors obtained through this nonlinear embedding for feature points on the same rigid object lie in a linear subspace of $\mathbb{R}^9$ of dimension at most 8. Indeed, [1] says that there is a vector $\text{vec}(F) \in \mathbb{R}^9$ orthogonal to all of the feature trajectories in this set (it also shows that the linear embedding $(x, y, x', y')$ lies on a 3-dimensional quadratic manifold). However, the subspace dimension can decrease due to two different reasons. First of all, if there are very few points (per motion), then they may span a lower-dimensional subspace. The second cause is degeneracy in the 3D configuration of the features. If all world points and both camera centers live on a ruled quadratic surface, then their corresponding subspace has dimension 7 or less. In particular, if all world points (but not necessarily the camera centres) are coplanar, the corresponding subspace will have dimension no larger than 6 (see [22, pg. 296]). Therefore, to make use of this embedding, the hybrid-linear modeling algorithm being employed must be tolerant of subspaces of mixed dimension.

Since the perspective camera assumption is accurate in a much broader range of situations than the affine camera model, subspaces are more apparent with the nonlinear embedding than with the linear embedding. However, the nonlinear embedding distorts the original sampling and results in lower-dimensional structures (of dimension at most 3) within the higher dimensional subspaces (of typical dimensions 6, 7 or 8), which is a serious obstacle for many HLM algorithms, especially ones using local spatial information.

3 Global Dimension

From here on, we will be considering 2-view motion segmentation under the perspective camera model, i.e. using the Kronecker embedding. We will present a global HLM method, which is well-suited for handling the data which results from this embedding. We begin our development by providing some intuitive motivation for our approach.

Imagine that we have access to an oracle, who for any set of vectors in $\mathbb{R}^D$, can provide for us a good, robust estimate

\footnote{A surface $S$ is ruled if through every point of $S$ there exists a straight line that lies on $S$.}
of the dimensionality of the set. Now, suppose we have a set of vectors which are sampled from a hybrid-linear distribution. Consider a general partition of the data set, and define the “vector of set dimensions” for that partition to be the vector of oracle-provided approximate dimensions of each respective set in the partition. Our inspiration is the observation that for most partitions one may happen upon, each set in the partition will typically contain points from many of the underlying subspaces. The associated vector of set dimensions will contain relatively large numbers, and the p-norm of this vector will be large. The p-norm of the vector of set dimensions will be referred to as the global dimension of the partition. The best way to make the global dimension small, it would seem, is to try and decrease all of the elements of the vector of set dimensions by grouping together vectors that come from common subspaces. This notion will be made precise and we will show, in fact, that under certain conditions, the natural partition of the data set (the one where point assignment agrees with subspace affiliation) is a global minimizer of the global dimension function.

Our approach to HLM will be to find the partition of a data set that yields the lowest possible global dimension. In this section, we develop the global dimension objective function in two parts. In we suggest a new class of dimension estimators that can perform the role of the oracle in our discussion above. In we define global dimension and explain why we expect its minimizer to reveal the clusters corresponding to the underlying subspaces. A fast algorithm for this minimization will be later described in

### 3.1 On Empirical Dimension

We present here a class of dimension estimators depending on a parameter $\epsilon \in (0, 1]$. For $u = (u_1, \ldots, u_k)$ and any $p > 0$, we use the notation $|u|^p$ to mean $|u_1|^p + \cdots + |u_k|^p$ (even for $p = \epsilon < 1$, where $|\cdot|^p$ is not a norm). For a given set of $N$ vectors in $\mathbb{R}^P$, we denote by $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{N \times D})^T$ the vector of singular values of the $D \times N$ data matrix $A$ (the matrix whose columns are the data vectors).

For $\epsilon \in (0, 1]$ the empirical dimension, denoted by $\hat{d}_\epsilon(v_1, v_2, \ldots, v_N)$ (or simply $\hat{d}_\epsilon$) is defined by

$$
\hat{d}_\epsilon(v_1, v_2, \ldots, v_N) := \frac{\|\sigma\|^\epsilon}{\left(\frac{1}{\epsilon}\right)^\epsilon}.
$$

When $\epsilon = 1$, this is sometimes called the “effective rank” of the data matrix.

The following theorem explains why $\hat{d}_\epsilon$ is a good estimator for dimension. Put simply, it says two things. First, if we rotate and/or uniformly scale our set of vectors by a non-zero amount, then the empirical dimension of the set does not change. Second, in the absence of noise, empirical dimension never exceeds true dimension, but it approaches true dimension in the limit (as the number of measurements goes to infinity) for spherically symmetric distributions. From now on we refer to $d$-dimensional subspaces as $d$-subspaces.

**Theorem 1** For $\epsilon \in (0, 1]$, $\hat{d}_\epsilon$ possesses the following properties:

1. $\hat{d}_\epsilon$ is invariant under dilations (i.e., scaling).
2. $\hat{d}_\epsilon$ is invariant under orthogonal transformations.
3. If $\{v_i\}_{i=1}^N$ are contained in a $d$-subspace of $\mathbb{R}^D$, then $\hat{d}_\epsilon \leq d$.
4. If $\{v_i\}_{i=1}^N$ are i.i.d. samples from a sub-Gaussian probability measure, which is spherically symmetric within a $d$-subspace and non-degenerate then $\lim_{N \to \infty} \hat{d}_\epsilon(v_1, \ldots, v_N) = d$ with probability 1.

To gain some intuition into the definition of empirical dimension, consider taking a large set of samples from a spherically symmetric distribution supported by a $d$-subspace. Call the covariance matrix for this distribution $Q$. As the number of samples becomes large, the empirical covariance matrix approaches $Q$, which has the first $d$ elements on the main diagonal all equal (call the value $\alpha^2$), and 0’s everywhere else. The empirical dimension of the set of vectors involves the singular values of the data matrix, which are approaching the square roots of the eigenvalues of $Q$. Hence, as the number of samples increases, we get:

$$
\hat{d}_\epsilon(v_1, \ldots, v_N) \to \frac{\|\alpha_1, \alpha_2, \ldots, \alpha_0, \ldots, 0\|}{\|\alpha_1, \alpha_2, \ldots, \alpha_0, \ldots, 0\|} = \frac{d^{1/\epsilon}}{d^{(1-\epsilon)/\epsilon}} = d.
$$

Thus, for any value of $\epsilon$ in $(0, 1]$, the empirical dimension approaches the true dimension of the set as the number of measurements increases.

If we look at a distribution that is not spherically symmetric, but still supported by a $d$-subspace, then empirical dimension tends to under-estimate the true dimension of the distribution, even as the number of samples approaches infinity. This is actually desirable behavior. If we take a spherically symmetric distribution in a $d$-subspace and imagine the process of collapsing it in one direction until it lies in a $(d-1)$-subspace, then true dimension behaves discontinuously. The true dimension of a large set of samples will

---

2. In a noiseless case this would return the dimension of the linear span of the set of vectors.
3. “Effective rank” is sometimes defined differently. See [56].

4. A measure is spherically symmetric within a $d$-subspace if it is supported on this subspace and invariant to rotations within this subspace.
5. A measure is non-degenerate on a subspace if it does not concentrate mass on any proper subspace. In our setting the measure is also assumed to be spherically symmetric, and this assumption is equivalent to assuming the measure does not concentrate at the origin.
equal \( d \) until the collapsing is complete; at that point the dimension will instantly drop to \( d - 1 \). Empirical dimension smoothly drops from \( d \) to \( d - 1 \) during this collapsing process. It is in this setting that we see the necessity of the parameter \( \varepsilon \). This parameter controls how quickly the empirical dimension drops from \( d \) to \( d - 1 \) in this process. More generally, a low value of \( \varepsilon \) results in a "strict" dimension estimator (meaning that it will not under-estimate dimension easily, even when distributions are asymmetric). When \( \varepsilon \) is large (approaching 1), empirical dimension is a lenient dimension estimator. It is much more tolerant of noise, but it may consequently under-estimate the dimension of highly asymmetric distributions. The trade-off is that when dealing with noisy data or distributions only approximately supported by linear subspaces, a stricter estimator can mistakenly interpret noise or distortion as energy in new directions, thereby causing an over-estimate of dimension. Numerical experiments (e.g., Fig. 2) have shown that values of \( \varepsilon \) between 0.3 and 0.7 seem to provide reasonable estimators, which tend to agree with our intuitive notion of dimension.

is not “tuned” to individual data sets, but is chosen based on the properties of the application as a whole and the nature of the embedding.

### 3.2 On Global Dimension

Assume we are provided a data set \( X \) in \( \mathbb{R}^D \) (in our application \( D = 9 \) with the nonlinear embedding) and a partition of it \( \Pi = (\Pi_1, \Pi_2, \ldots, \Pi_K) \) for some \( K \in \mathbb{N} \) (i.e., \( \{\Pi_i\}_{i=1}^K \) are disjoint subsets of \( X \) whose union is \( X \)). We also assume that \( X \) lies on a union of \( K \) subspaces and denote the “correct” (or natural) partition of the data (where each subset contains only points from a single underlying subspace) by \( \Pi_{Nat} \). For a fixed \( \varepsilon \in (0, 1] \), \( \{\hat{d}_{\varepsilon,i}\}_{i=1}^K \) are the empirical dimensions of the sets \( \{\Pi_i\}_{i=1}^K \). We seek to minimize a function based on these dimensions to recover \( \Pi_{Nat} \). To this end we define global dimension (GD). When thinking of this function, we take the set of data vectors to be fixed and given, and we view GD as a function of partitions, \( \Pi \), of the set of data vectors. For a fixed \( p \in (0, \infty) \) (we discuss the meaning of \( p \) later) we define GD as follows:

\[
GD(\Pi) = \|(\hat{d}_{\varepsilon,1}, \hat{d}_{\varepsilon,2}, \ldots, \hat{d}_{\varepsilon,K})\|_p = \left(\sum_{i=1}^K \hat{d}_{\varepsilon,i}^p\right)^{1/p}.
\]  

Our strategy for recovering \( \Pi_{Nat} \) will be to try and find the partition of the data set that minimizes \( GD(\Pi) \). Intuitively, by trying to minimize the \( p \)-norm of the vector of set dimensions, we are looking for a partition where all of the set dimensions are small. Imagine trying to minimize this objective function by hand, and starting with a partition close to, but not equal to \( \Pi_{Nat} \). If there is a point assigned to the wrong cluster, then removing it from the set it is currently assigned to should result in a significant drop in the dimension of that particular set. Re-assigning that point to the correct set, on the other hand, will have little impact on the dimension of the target set because the point will lie approximately in the span of other points already in the set.

Thus, such a change would cause a significant drop in one of the set dimensions, without disturbing the other sets, and the global dimension will decrease. This would suggest that amongst partitions that are close to it, \( \Pi_{Nat} \) yields the lowest global dimension. Additionally, if one considers a “random”, or usual partition, then each set in that partition will tend to contain vectors from many different subspaces. Each set will have a large dimension, and the global dimension will exceed that of \( \Pi_{Nat} \). This would suggest that minimizing global dimension may be a reasonable objective if we want to recover \( \Pi_{Nat} \).

Unfortunately, there can exist certain special partitions of a data set that result in low global dimension (in some cases even lower than that of \( \Pi_{Nat} \)). For example, let us
choose $p = 1$, so that the global dimension of a partition is simply the sum of the dimensions of its constituent parts. Now consider 3 lines in the plane, and a data set consisting of many points sampled from each line. In this case $\Pi_{Nat}$ will consist of three sets. Each set will contain only points from a single line. The dimension of each set in $\Pi_{Nat}$ is 1. Hence, $GD(\Pi_{Nat}) = 3$. On the other hand, if we consider the “degenerate” partition, that simply puts all points in a single set, then since we are in $\mathbb{R}^2$, the dimension of that set, and hence the global dimension of the partition, is 2.

The above example is actually rather special. Consider the same data set, but set $p$ to a large value instead of 1. When $p$ is large, the global dimension approximately returns the largest value from $\{d_i\}_{i=1}^K$. Now consider minimizing this quantity, subject to the constraint that the partition contains no more than 3 sets. Minimizing global dimension in this setting penalizes partitions consisting of fewer, higher-dimensional sets instead of multiple, more balanced sets. Specifically, the global dimension of the degenerate partition is again approximately 2, while the global dimension of $\Pi_{Nat}$ is approximately 1, since that is the maximum dimension of its constituent sets. In fact, as shown in the next theorem, using large $p$ effectively resolves the issue of special partitions yielding lower global dimension than $\Pi_{Nat}$.

We will consider the setting where we have $K$ distinct linear subspaces of $\mathbb{R}^D$, each of dimension $d < D$. Call these subspaces $\{L_k\}_{k=1}^K$. Assume we have a collection of non-degenerate measures $\{\mu_k\}_{k=1}^K$ supported by these subspaces (so that $\mu_k$ is supported by $L_k$, $k = 1, 2, ..., K$). Let $\{v_n\}_{n=1}^N$ be a set consisting of $N_i$ i.i.d. points from each $\mu_k$ (so $N = N_1 + N_2 + ... + N_K$). We require that $N_k > d$ for each $k$ so that each subspace is adequately represented in the data set. Let $GD_{True}$ be global dimension for a fixed parameter $p$, defined using true dimension as the “dimension estimator” for a set. That is, $GD_{True}(\Pi) = \| (d_{True}(\Pi_1), \ldots, d_{True}(\Pi_K)) \|_p$ where the $\Pi_k$ are the constituent sets of the partition $\Pi$ and $d_{True}(\bullet)$ returns the true dimension of its parameter set. Then, we get the following result:

**Theorem 2** Let $\{L_k\}_{k=1}^K$, $\{\mu_k\}_{k=1}^K$, $\{v_n\}_{n=1}^N$ satisfy the conditions above. If
\[
p > \ln(N)/(\ln(d+1) - \ln(d)),
\]then amongst all partitions of $\{v_n\}_{n=1}^N$ into $K$ or fewer sets, the natural partition is almost surely (w.r.t $\{\mu_k\}_{k=1}^K$) the unique minimizer of $GD_{True}$.

The weakness of the above theorem is that it requires all of the intrinsic subspaces to have the same dimension. In practice, the global dimension objective function appears to be rather robust to subspaces with mixed dimensions. If there is a large difference in dimension between two subspaces in a dataset, then the minimum of global dimension tends to be very near $\Pi_{Nat}$, the only difference being that a few points from the higher-dimensional set are re-assigned to lower-dimensional sets to balance out the set dimensions.

Theorem 2 gives us a quantitative way of selecting an appropriate value of $p$ for our application. Specifically, if we identify the largest number of clusters we will need to address ($K$) and an upper bound for $d$, then the right-hand side (RHS) of (5) gives us a lower bound on the value of $p$. As long as $p$ is larger than this bound, then Theorem 2 ensures that the natural partition (uniquely) minimizes global dimension. Table I exemplifies the values of the RHS of (5) for different values of $d$ and $K$. We do not want to choose $p$ extravagantly large because of the potential for numerical issues when taking large powers. In our setting, we want to be able to handle up to 4 sets and we will use one less than the ambient dimension as an upper bound for the intrinsic dimension ($d = 8$). According to Theorem 2 we should select $p \geq 11.77$. In all of our experiments we set $p = 15$ to give us a safety margin. We did some tests on one of our motion segmentation databases (outlier-free RAS) to see how sensitive the minimizer of global dimension is to $p$ in practice. We found that values as low as $p = 10$ result in nearly identical performance to $p = 15$, and we don’t start to see significant degradation in results in the other direction until $p = 25$.

**Table 1**: Values of the RHS of (5) for various values of $K$ and $d$. Theorem 2 ensures that $\Pi_{Nat}$ is the unique minimizer of $G_p$ when $p$ is larger than these values.

| $K$ | 2   | 3   | 4   | 5   | 6   | 7   | 8   |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 2   | 5.89| 5.19| 4.50| 3.80| 3.11|     |     |
| 3   | 9.33| 8.23| 7.13| 6.03| 4.92|     |     |
| 4   | 11.77| 10.38| 8.99| 7.60| 6.21|     |     |

4 A Fast Algorithm for Minimizing Global Dimension

Global dimension is defined on the set of partitions of a data set. With a discrete domain, finding ways of quickly minimizing the objective function is non-trivial. In this section we briefly introduce a method, which we will call Global Dimension Minimization (GDM) for doing exactly this.

GDM is based on the gradient projection method [6 §2.3]. In order to apply a gradient-based method, we need to re-formulate the problem so that we have a smooth objective function over a convex domain. To do this we employ the notion of fuzzy assignment. Rather than trying to assign each data point a label, identifying it with a single cluster, we allow each point to be associated with every cluster simultaneously, in varying amounts. Specifically, we assign each data point $v_j$ a probability vector where the $i$'th coordinate holds the strength of $v_j$'s affiliation with cluster $i$. Assuming we have a data set of $N$ points in $\mathbb{R}^D$, and we seek $K$ clusters,
we need \( N \) probability vectors of length \( K \) to encode the soft partition of the data. This membership information will be stored in a membership matrix, \( M \), where each column is a probability vector. Element \((i, j)\) of the matrix \( M \) holds the strength of \( v_j \)'s affiliation with cluster \( i \).

The next step is to extend the definition of global dimension so that it is defined on soft partitions in a meaningful way. In its original formulation, to evaluate the global dimension of a partition, we would break up the data set into parts, based on the partition, and estimate the dimension of each part using empirical dimension. To extend this to soft partitions, we estimate the dimension of the \( k \)th set in a partition by scaling each data point by its respective affiliation strength to set \( k \) (\( v_n \) is multiplied by \( M_{(k,n)} \)). We then use empirical dimension to estimate the dimension of the scaled set. In essence, each point is now included in each dimension estimate. However, if a point is scaled so that it lays near the origin when considering a given set, it has little impact on the estimated dimension of that set. In fact, if we look at the global dimension of a soft partition that assigns each data point entirely to a single set (\( M \) has only 1's and 0's in it), then the global dimension of that soft partition, using our new definition, agrees with the global dimension of the corresponding “hard partition”, using our original definition. Thus, this change is a reasonable extension of the original definition to soft partitions. Our extended definition of global dimension is:

\[
GD = \| (d^1_1, d^2_1, \ldots, d^K_1) \|_p
\]

(6)

where \( d^2_k = \hat{d}_k(M_{(k,1)}v_1, M_{(k,2)}v_2, \ldots, M_{(k,N)}v_N) \).

With this modified formulation, global dimension is an almost-everywhere differentiable function defined over the Cartesian product of \( N \) \( K \)-dimensional probability simplices. One can check that this is a convex domain (the product of convex sets is convex). A natural approach to minimizing a problem of this sort is the gradient projection method [32,33]. In this method, we begin at some initial state, compute the gradient of the objective function, take a step in the direction opposite the gradient, and then project our new state back into the domain of optimization. This is repeated until our state converges.

The method of global dimension can be computed, but we need some notation first. For \( i = 1, \ldots, K \), we denote by \( A_k \) the \( D \)-by-\( N \) matrix whose \( j \)th column equals \( M_{(k,j)}v_j \) for \( j = 1, 2, \ldots, N \) (i.e., \( A_k \) is the data matrix scaled according to weights for cluster \( k \)). Let \( A_{k1} = U_k \Sigma_k (V_k)^T \) be the thin SVD of \( A_k \), and \( \sigma_k \) denote the vector of elements from the diagonal of \( \Sigma_k \). Let \( \delta = \varepsilon/(1 - \varepsilon) \). Define

\[
D_k = \left( \frac{\| \sigma_k \|_f - \varepsilon \| \sigma_k \|_\delta}{\| \sigma_k \|_\delta^2} \right) (\Sigma_k)^{f-1} - \left( \frac{\| \sigma_k \|_f - \varepsilon \| \sigma_k \|_\delta}{\| \sigma_k \|_\delta} \right) (\Sigma_k)^{f-1}.
\]

(7)

**Theorem 3** The derivative of global dimension w.r.t. an arbitrary element of the membership matrix \( M \) is given by:

\[
\frac{\partial GD}{\partial M_{(k,n)}} = V_{k(n)} \left( \| d^1_k \|_{p-1} \| d^2_k, d^3_k, \ldots, d^K_k \|_p \right)^{-p} D_k (U_k)^T A_{(k,n)}.
\]

(8)

A proof of Theorem 3 is included in the appendix. This theorem allows us to evaluate the gradient vector of global dimension. As was mentioned before, in an iteration of the gradient projection method we take a step in the direction opposite the gradient. Computing a good step size is frequently a challenging task, but here we are fortunate. Our domain has a meaningful natural scale, since it is formed as a product of probability simplices. Intuitively, our step size should be large enough to move us across the entire space in a reasonable number of steps, but small enough that any individual membership vector can move only a fraction of the way across its own simplex in one step. In practice, we scale each step so that the membership vectors most affected by the step move a distance of .3 on average. This seems to work well in general.

Finally, one can check that projecting onto the domain of optimization can be accomplished by individually projecting each column of \( M \) onto the standard \( K \)-dimensional probability simplex.

We have outlined a projected gradient descent method for minimizing global dimension. The above method forms the core of the GDM algorithm. However, since the global dimension function is non-convex (and hence may contain multiple local minimums), it is important to achieve reasonably good initialization. Our initialization strategy is inspired by ALC [31]. We start with a “trivial” partition where each point is in its own set, and we randomly select many pairs of sets in the partition. For each pair, we hypothetically merge the two sets and measure the resulting global dimension. We select the pair that results in the lowest global dimension when merged and we effect that merge. We then repeat the process iteratively until we have the desired number of sets in our partition (in each step the number of sets in the partition decreases by 1). After initialization, the projected gradient descent algorithm is run until convergence (or for a fixed, but large number of iterations). Thresholding is performed to recover a “hard partition” from our soft partition (point \( j \) is assigned to cluster \( i \) if \( M_{(i,j)} \) is the largest element from column \( j \) of \( M \)). After this is done, we perform a final genetic stage to clean up small errors which may have occurred in any of the previous stages. This is done by taking each point and hypothetically re-assigning it to each different cluster (while all other point assignments are kept fixed) and retaining the assignment that results in the lowest global dimension. This is repeated a few times or until no single-point re-assignments reduce global dimension.
This primarily helps with placing points that lie near the intersections of different subspaces (their fuzzy assignments may associate them almost equally to 2 different subspaces, making them difficult to place). Finally, we run this entire process several times and return the best partition of all runs (as measured by global dimension).

Algorithm 1 GDM Algorithm for HLM

Input: \( X = \{x_1, x_2, \ldots, x_N\} \subseteq \mathbb{R}^D \), data, \( K \): number of clusters, \( p \): global dimension parameter, \( \varepsilon \): empirical dimension parameter, \( n_1, n_2, n_3 \): number of iterations (default: \( n_1 = n_2 = 10, n_3 = 30 \))

Output: A partition, \( \Pi \), of \( X \) into \( K \) disjoint clusters

for \( i = 1 : n_1 \) do
  \( \Pi := \) Partition of \( X \) where each point is in its own set.
  while number of sets in \( \Pi \) greater than \( K \) do
    randomly choose several pairs of sets.
    for each pair, measure the effect on global dimension if the pair is merged.
    Merge the pair of sets which results in the lowest global dimension.
  end while
  convert \( \Pi \) to a soft partition, encoded in membership matrix \( M \).
for \( j = 1 : n_2 \) do
  compute gradient of global dimension, \( VGD \).
  \( p = \) average magnitude of largest 10% of columns of \( VGD \).
  project each column of \( M \) onto the standard \( k \)-dimensional probability simplex.
end for
  convert \( M \) back to a “hard partition”, \( \Pi \), by thresholding.
for \( j = 1 : n_3 \) do
  for \( n = 1 : N \) do
    check if re-assigning point \( n \) to some other cluster decreases global dimension.
    if so, re-assign point \( n \) to that cluster.
  end for
end for
return partition from all runs with lowest global dimension.

4.1 Complexity of GDM

A thorough analysis of the computational complexity is not included here; this is a short summary of the computational aspects involved. The main numerical component of GDM is computing \( VGD \). For a single iteration its complexity is \( O(K \cdot N \cdot D^2) \). Our choice of \( p \) requires a sorting procedure and is thus of order \( O(N \cdot \log(N)) \) operations for a single iteration. The initialization of the algorithm via ALC-type procedure \([31]\) requires \( O(n_1 \cdot N \cdot \log(N) \cdot D^2) \) operations. Also, the last genetic step has the following complexity \( O(n_1 \cdot n_2 \cdot K \cdot D^2 \cdot N^2) \) (without taking advantage of incremental SVD). In theory, we can make the algorithm linear in the number of points \( N \), by randomly initializing it, removing the genetic “clean-up” step and changing how we select our step size\(^6\). We have good numerical evidence, even with large \( N \), that this can result in good accuracy and speed for artificial data. Regardless, for the values of \( N \) in our application the algorithm is sufficiently fast and these additional steps help improving accuracy, especially for points which are nearby several clusters (whose percentage is not negligible when \( N \) is small).

5 Detecting and Rejecting Outliers with GDM

In practice, it turns out that the GDM algorithm described above is naturally robust to a small number of outliers (in that they do not tend to affect the classification of inliers), but no instruments were put in place for explicitly detecting or rejecting these outlying points. In this section, we introduce a modification to GDM that allows for explicit outlier detection and rejection. The guiding intuition is that an outlier has the property that if the true hybrid-linear structure is reflected in a partition, then no matter which group we assign the outlier to, it causes a significant increase in the empirical dimension of that group. This, in turn, results in a significant increase in global dimension. In other words, if we have a partition that reflects the true hybrid-linear structure of the data set, then there is no good place to put an outlier. If the algorithm was given the option of paying a fixed, low price for the right to ignore a given point, it would make sense for it to exercise this option on outliers, and only segment inliers.

We propose modifying the global dimension objective function, and the accompanying variational development in the following way:

\[
GD(M) = \alpha \|M_{1:}\|_1 + \|\hat{d}_{\varepsilon,2,\ldots,\hat{d}_{\varepsilon,K+1}}^T\|_p
\]

where:

\[
\hat{d}_{\varepsilon,k} = \hat{d}_{\varepsilon}(M_kv_1,M_kv_2,\ldots,M_kv_N).
\]

This modification adds an additional “cluster” to the problem (call it cluster 1), and we treat it differently than the others. Clusters 2 through \( K + 1 \) contribute to the global dimension in the same way that they did in the original development. Cluster 1 contributes to the cost function the sum of the membership strengths of all data points to this cluster. This is the “fuzzy assignment” version of the following notion: we allow the algorithm to pay a fixed price, \( \alpha \), for the right to ignore any particular data point (not assign it to any true cluster).

\(^6\) This is assuming that we will not require more iterations to get close enough to the minimum that we can apply thresholding. In our experiments the number of needed iterations does not appear to grow with \( N \), but we do not have any results to guarantee this.
5.1 Modification to GDM

The proposed modification to the objective function only trivially changes the state space (now it is the product of \( N K + 1 \)-dimensional probability simplexes, as opposed to \( K \)-dimensional simplexes). Thus, our method of projecting states onto the convex domain is effectively the same. The change to the objective function does mean that we must re-evaluate the gradient of global dimension. The computation is very similar to the unmodified version, and the result is:

\[
\frac{\partial GD}{\partial m^n_1} = \alpha m^n_1^{n-1} 
\]

(11)

and, for all \( k > 1 \)

\[
\frac{\partial GD}{\partial m^n_k} = V_{k(n,::)} \hat{\beta}_{p_{K}}^{p_{K}-1} \left( \hat{\beta}_{p_{k}}^{p_{k}} + \ldots + \hat{\beta}_{p_{K}}^{p_{K}} \right)^{\frac{1}{\pi}-1} D_k U^T_k v_n,
\]

(12)

where the notation and constants are as defined in [4]. Thus, the necessary modifications to GDM are:

1. Update the evaluation of the objective function \( GD \) according to (9).
2. Update the initialization of the state vector to include an outlier group.
3. Update the state projection routine to accommodate additional dimensions in domain.
4. Update the evaluation of \( VGD \) according to [11] and [12].

5.2 Practical Implementations of Outlier Rejection

We have described an idea for how to handle outliers, but it introduces a new parameter, \( \alpha \). It is not immediately clear how one should choose this parameter, and how sensitive the results will be to it. In theory one would need to choose an outlier cost, \( \alpha \), that is not so high that nothing is ever assigned to the outlier group, but not so low that large quantities of inliers are assigned to this group. The appropriate values would likely depend on multiple quantities, like intrinsic dimension, noise level, and distortion of the underlying subspaces. These are quantities that can vary not just between applications, but also from data set to data set for a single application. Applying the suggested modification exactly as proposed (and trying to “tune” this parameter) would therefore lead to an unreliable and unpredictable algorithm. We refer to this approach as GDM-Naive, and Figure 3 illustrates why this method is unsound. Instead, we propose two variations of this method, which lead to more reliable solutions.

1. **GDM Known-Fraction**: Run the proposed algorithm with a fixed, low value of \( \alpha \) (we use \( \alpha = 0.01 \)) but stop before the threshold step. Rank the data points according to their membership strengths to the outlier group. Remove a pre-set fraction of the data set (the part that most strongly affiliates with the outlier group). Continue with the classic (non-outlier version) of the variational algorithm on the surviving points only, this provides the inlier segmentation. The points that were removed are labelled outliers.

2. **GDM Model-Reassign**: Run method (GDM Known-Fraction). Fit subspaces of appropriate dimension (round the empirical dimension) to each set in the resulting partition. Re-assign all points (including those that were decided to be outliers) according to their distances from each subspace. Call a point an outlier if it is more than some fixed distance, \( \kappa \), from all of the subspaces.

Each of the proposed methods handles the task of selecting \( \alpha \), but introduces a new parameter. For method this is the percentage of the data set to throw out. For method the new parameter is the maximum distance a point can be from a subspace to be considered an inlier. Both of these parameters are more natural than selecting \( \alpha \). In a noisy environment, one may have an idea, based on experiments, of what percentage of the data set will be outliers, or what the inlier modelling error tends to be. Additionally, when using the “Model-Reassign” method, one could find the average and variance of the residuals, \( \mu \), and \( \sigma \) respectively, when fitting subspaces to the inlier clusters. These quantities can be used to come up with a reasonable value of \( \kappa \) for a given application (\( \mu + r \sigma \) for some \( r \)). One could also find these values on a per-cluster basis and have a different outlier threshold for each cluster.

6 Results on Real-World Data

6.1 Performance in the Absence of Outliers

We tested the GDM algorithm on 2 motion segmentation databases. First, we used the outlier-free RAS database [35] [10] and compared with many leading methods in 2-view segmentation. We noticed that some of the HLM methods performed better when using the linearly embedded point correspondences than with the nonlinear embedding. Therefore, in Table 2 we present each of the competing HLM algorithms twice. Where “Linear” appears, the algorithm was run on the feature trajectories in \( \mathbb{R}^4 \). Where “Nonlinear” appears, the algorithm was run on the Kronecker products (in \( \mathbb{R}^9 \)) of the standard homogeneous coordinates of...
Large $\alpha$: Nothing ends up in outlier group.

Medium $\alpha$: Only outliers end up in outlier group.

Low $\alpha$: Some inliers end up in outlier group.

The three images here illustrate the problem with GDM-Naive. Each triangle represents the probability simplex containing the fuzzy assignment vectors for a fictitious data set. The fuzzy assignment for each point is plotted after many iterations of GDM. Points in red are inliers and points in green are outliers. The quantization regions (for the threshold step) are numbered 1-3. One can see that if $\alpha$ is not chosen correctly, points can be quantized into the wrong cluster. On the other hand, the outlier ranking of a point (the # of points closer the outlier corner) is a more stable quantity. (Color figure online)

Fig. 3: Graphical depiction of the problem with GDM-Naive

Table 3: Average runtimes (per file) of HLM-based methods on non-linearly embedded (outlier-free) RAS data.

| Method | Runtime (seconds) |
|--------|------------------|
| GDM    | 12.7             |
| SCC    | 2.3              |
| MAPA   | 5.6              |
| SSC    | 89.5             |
| SLBF   | 4.0              |
| LRR    | 0.8              |

From Table 3 and Figure 5 we can see that GDM performs very competitively on this database. There is only a single file (#8) on which GDM exhibits significant error. This file contains features from two bent magazines as well as a rigid background. Since the bent magazines are clearly non-rigid, our model assumptions are not met (see Fig. 6). There were two methods in the comparison that had a lower average misclassification error than GDM (“SCC Linear” and “SLBF Linear”). This is because they perform significantly better on file (#8). Both of these are spectral methods, accompanied by the linear embedding, and are therefore better able to handle the manifold structure that results from the non-rigidity of the objects in this file. Amongst the other files however, GDM performs better on average than both of these two methods (see the last column of Table 3). Comparing just the HLM-based methods on the non-linearly-embedded data, GDM performs better than any other method, with the most perfect classifications and the fewest number of files with significant errors. Figure 5 more clearly emphasizes this superb performance amongst methods using the nonlinear embedding.

We also performed experiments on the Hopkins155 database [41]. For 2-view segmentation we extracted the first and last frame of each sequence and performed 2-view...
It is also interesting to note that our results for 2-views are comparable to the reference results with n-views. That is, the results of GDM are the best one can expect with pure linear modeling given many views and assuming an affine camera model. GDM for n-views gave comparable results and we thus did not include it. On the other hand, both SLBF-MS and SSC-N are able to obtain better results with n-views and this may be because their machinery of spectral clustering (together with good choices of spectral weights) allows them to take into account some of the manifold structure and nearness of points (information beyond linear modeling).

6.2 Performance in the Presence of Outliers

We tested the methods suggested in [5.2] on the outlier-corrupted RAS database [53]. The performance of classic GDM (no outlier rejection machinery) is also presented on this database, as is the performance of GDM on the corresponding outlier-free database (for comparison purposes).

We also show results from three competing methods for segmenting motion with outliers: RAS [33], HOSC [2], and LRR [29,28] with outlier rejection performed by identifying the largest columns of $E$, as suggested in [28, pg. 9]. The details of this experiment, including parameter values, are given in [5.8].

It is non-trivial to fairly compare different algorithms in the presence of outliers. Each method generally has at least one parameter for controlling how it handles outliers. This parameter balances the desire for a high outlier detection rate with a desire for a low false alarm rate (these two quantities are invariably correlated). Using any popular metric for evaluating segmentation accuracy (like misclassification rate for true inliers\textsuperscript{8}), the performance of each algorithm will depend substantially on its outlier handling parameter.

In general terms, if an algorithm is allowed to discard points as outliers more freely, then the accuracy on the surviving points will improve. Thus, if one method is more conservative than another in discarding points as outliers, the results

\textbf{Table 2:} Misclassification Rates (given as % Error) on the outlier-free RAS database.

| Method/Embedding | File Number | Average w/o File 88 |
|------------------|-------------|---------------------|
|                  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | Average |
| GDM Nonlinear    | 0.85 | 0.00 | 1.57 | 0.65 | 0.00 | 0.00 | 0.00 | 12.76 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.22 |
| SCC Linear       | 0.85 | 0.00 | 1.18 | 0.00 | 0.00 | 1.32 | 0.00 | 0.00 | 1.18 | 0.00 | 0.00 | 0.00 | 0.00 | 0.85 |
| SCC Nonlinear    | 0.85 | 0.00 | 24.11 | 0.65 | 0.00 | 0.00 | 19.18 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.18 |
| MAPA Linear      | 0.85 | 3.65 | 1.18 | 0.62 | 0.00 | 13.00 | 13.97 | 1.29 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.87 |
| MAPA Nonlinear   | 0.85 | 20.55 | 23.65 | 0.65 | 0.00 | 21.92 | 26.52 | 17.73 | 0.00 | 0.00 | 13.97 | 14.34 | 0.00 | 0.89 |
| SLBF Linear      | 0.85 | 0.00 | 1.18 | 0.00 | 0.00 | 6.25 | 22.92 | 0.00 | 0.00 | 0.00 | 14.64 | 13.54 | 4.40 | 0.85 |
| SLBF Nonlinear   | 0.85 | 0.00 | 1.12 | 1.94 | 0.00 | 0.00 | 19.18 | 0.00 | 0.00 | 10.57 | 0.00 | 13.97 | 0.00 | 0.00 |
| LRR Linear       | 5.08 | 24.66 | 1.20 | 0.00 | 9.04 | 0.00 | 13.97 | 9.29 | 14.12 | 6.59 | 1.18 | 0.00 | 0.00 | 0.85 |
| LRR Nonlinear    | 1.27 | 29.66 | 1.18 | 2.38 | 2.38 | 2.74 | 0.00 | 29.12 | 0.00 | 0.00 | 8.93 | 14.81 | 18.68 | 8.47 |
| HOSC d=2         | 0.85 | 0.00 | 24.11 | 1.61 | 0.00 | 0.00 | 22.94 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.00 |
| HOSC d=3         | 1.27 | 23.74 | 24.41 | 2.25 | 0.00 | 19.18 | 12.15 | 19.59 | 23.75 | 0.00 | 1.43 | 1.01 | 17.58 | 11.45 |
| Average          | 1.22 | 0.26 |

\textsuperscript{8} “True inliers” are points that are inliers according to ground truth.
will likely be skewed in favor of one method over the other. It is therefore important when looking at segmentation accuracy to think in terms of accuracy for a given true positive rate (TPR) and false positive rate (FPR):

\[
\text{TPR} = \frac{\# \text{of outliers that were identified as outliers}}{\# \text{of outliers in dataset}} \times 100, \\
\text{FPR} = \frac{\# \text{of inliers that were identified as outliers}}{\# \text{of inliers in dataset}} \times 100.
\]

There are two aspects of these algorithms we wish to compare. The first is outlier detection performance (how good is each method at distinguishing between inliers and outliers). The second is segmentation performance, where we evaluate how good each method is at segmenting motions in the presence of outliers.

To compare the outlier detection performance of multiple methods, a common tool is the ROC curve, which parametrically plots the TPR vs. FPR as a function of the outlier parameter for a method. A “random classifier” that randomly labels points as inliers or outliers will have an ROC curve lying along the line TPR = FPR. An ideal classifier will follow the line TPR = 1. Hence, methods can be compared by seeing which ROC curve is highest over the broadest range of FPRs (or over the FPRs one is interested in). The ROC curves for GDM (using the Model-Reassign outlier detection method and varying k), LRR (by varying λ), RAS (by varying “outlierFraction”), and HOSC (by varying α), are presented in Fig. 7.

GDM was again run using the nonlinear embedding of the data. HOSC was run with the linear embedding and LRR was run with the nonlinear embedding since these were the cases that yielded the best performance in the outlier-free tests for each algorithm (see §8.4 for more details). From Fig. 7 we can see that GDM is very competitive at detecting outliers on this database. At low FPRs GDM yields comparatively excellent performance. At higher FPRs HOSC has a moderate advantage at outlier detection vs. GDM. However, it will be seen later (Table 5) that HOSC is not competitive at segmentation in the presence of outliers. Furthermore, the presented HOSC results were prepared using \(d = 2\) (see §8.4), instead of \(d = 3\) as argued for by its authors. Using \(d = 3\) gave worse results and made the algorithm take an extremely long time to execute.

The TPR and FPR for a robust segmentation algorithm cannot generally be controlled independently or arbitrarily. Thus, for a comparison of segmentation accuracy, one must select “reasonable” parameters for each method, which correspond to the same general region of ROC space. It should be understood that since the TPR and FPR cannot be controlled exactly for each method, any such comparison is inherently unfair, and by manipulating outlier parameters the results can be skewed somewhat in any direction.

For the purpose of fairly comparing GDM with other methods, we must select only one of the suggested outlier detection schemes for GDM (“GDM - Known Fraction” or “GDM Model-Reassign”). To effectively use “GDM - Known Fraction”, one must either know roughly what fraction of his or her data are going to be outliers, or be in a situation where over-rejecting points as outliers is acceptable.

### Table 4: The mean and median percentage of misclassified points for two-motions and three-motions in Hopkins 155 database with comparisons to state-of-the-art n-views. Winning results amongst the 2-view methods are bold-faced in each category.

| Category | Checker | Traffic | Articulated | All |
|----------|---------|---------|-------------|-----|
| 2-view   | Mean    | Median  | Mean        | Median  | Mean    | Median  |
| GDM      | 2.79    | 0.00    | 1.78        | 0.00   | 2.66    | 0.00    |
| MAPA     | 12.85   | 14.07   | 6.49        | 6.93   | 7.15    | 5.33    |
| SCC (d=7)| 2.79    | 0.00    | 1.97        | 0.00   | 3.42    | 0.00    |
| SLBF (d=6)| 8.18 | 1.39    | 3.98        | 0.53   | 4.73    | 0.40    |
| SLBF-MS (2F,3) | 1.28 | 0.00    | 0.21        | 0.00   | 0.94    | 0.00    |
| SSC-N (4K,3) | 1.29 | 0.00    | 0.29        | 0.00   | 0.97    | 0.00    |
| REF      | 2.76    | 0.49    | 0.30        | 0.00   | 1.71    | 0.00    |

| 3-view   | Mean    | Median  | Mean        | Median  | Mean    | Median  |
| GDM      | 5.37    | 3.23    | 4.23        | 2.69    | 5.32    | 5.32    |
| MAPA     | 21.89   | 19.49   | 13.15       | 13.04   | 9.04    | 9.04    |
| SCC (d=7)| 8.05    | 5.85    | 4.67        | 5.45    | 5.85    | 5.85    |
| SLBF (d=6)| 14.08| 12.80   | 7.93        | 6.75    | 4.79    | 4.79    |
| SLBF-MS (2F,3) | 3.33 | 0.39    | 0.24        | 0.00    | 2.13    | 2.13    |
| SSC-N (4K,3) | 3.22 | 0.29    | 0.53        | 0.00    | 2.13    | 2.13    |
| REF      | 6.28    | 5.06    | 1.30        | 0.00    | 2.66    | 2.66    |

GDM was again run using the nonlinear embedding of the data. HOSC was run with the linear embedding and LRR was run with the nonlinear embedding since these were the cases that yielded the best performance in the outlier-free tests for each algorithm (see §8.4 for more details). From Fig. 7 we can see that GDM is very competitive at detecting outliers on this database. At low FPRs GDM yields comparatively excellent performance. At higher FPRs HOSC has a moderate advantage at outlier detection vs. GDM. However, it will be seen later (Table 5) that HOSC is not competitive at segmentation in the presence of outliers. Furthermore, the presented HOSC results were prepared using \(d = 2\) (see §8.4), instead of \(d = 3\) as argued for by its authors. Using \(d = 3\) gave worse results and made the algorithm take an extremely long time to execute.

The TPR and FPR for a robust segmentation algorithm cannot generally be controlled independently or arbitrarily. Thus, for a comparison of segmentation accuracy, one must select “reasonable” parameters for each method, which correspond to the same general region of ROC space. It should be understood that since the TPR and FPR cannot be controlled exactly for each method, any such comparison is inherently unfair, and by manipulating outlier parameters the results can be skewed somewhat in any direction.

For the purpose of fairly comparing GDM with other methods, we must select only one of the suggested outlier detection schemes for GDM (“GDM - Known Fraction” or “GDM Model-Reassign”). To effectively use “GDM - Known Fraction”, one must either know roughly what fraction of his or her data are going to be outliers, or be in a situation where over-rejecting points as outliers is acceptable.
Table 5: Misclassification Rates (given as % Error) of inliers on the RAS database. All but ‘Classic GDM - clean’ are misclassification rates when run on the outlier-corrupted datasets. ‘GDM - clean’ gives the performance of the unmodified GDM algorithm, when run on the outlier-removed datasets (included as a reference).

| File Number | Average w/o File #8 |
|-------------|---------------------|
| 0 (w/o File #8) | 2.98 |
| 1           | 0.00                |
| 2           | 0.00                |
| 3           | 4.33                |
| 4           | 1.29                |
| 5           | 0.95                |
| 6           | 0.00                |
| 7           | 0.00                |
| 8           | 12.63               |
| 9           | 0.00                |
| 10          | 0.00                |
| 11          | 2.02                |
| 12          | 17.58               |
| 13          | 3.12                |

Table 6: True Positive Rate (TPR) and False Positive Rate (FPR) for each method in our segmentation comparison in Table 5. GDM - Model Reassign, RAS, LRR, and HOSC were each tuned to achieve a false positive rate in the range of 0.01 to 0.08.

| Method                      | TPR  | FPR  |
|-----------------------------|------|------|
| GDM - Model-Reassign        | 0.56 | 0.01 |
| GDM - Classic               | NA   | NA   |
| RAS                         | 0.74 | 0.08 |
| LRR                         | 0.49 | 0.04 |
| HOSC (d=2)                  | 0.71 | 0.06 |
| GDM - clean                 | NA   | NA   |

One can see from Table 5 that “GDM Model-Reassign” causes an overall improvement in segmentation accuracy (vs “GDM - Classic”) in the presence of outliers. There were several files where the outliers cause the classic GDM method to misclassify large fractions of the data sets (files 4, 8, and 11 have inlier misclassification rates over 20%). On these files the error rates of “GDM Model-Reassign” are dramatically lower. There are some files where the outlier detection framework appears to hurt performance, but in most of these cases the degradation is slight. The results for GDM are better in most cases (and on average) than the competing methods, although there are a few files where GDM is outperformed by a small margin. Unlike the strong outlier detection performance of HOSC discussed earlier, the segmentation capabilities of HOSC appear very intolerant to outliers (if even a few outliers slip through, segmentation suffers).

7 Conclusions

We presented a new approach to 2-view motion segmentation, which is also a general method for HLM. Its development was motivated by the main obstacle of recovering multiple subspaces within the nonlinear embedding of point correspondences into $\mathbb{R}^9$; namely, the nonuniform distributions along subspaces (of unknown dimensions). The idea was to minimize a global quantity, i.e., global dimension. Unlike KSCC [10], which also exploits global information, this approach does not make an a-priori assumption on the
dimensions of the underlying subspaces. We formulated a fast method to minimize this global dimension, which we referred to as GDM. We demonstrated state-of-the-art results of GDM for 2-view motion segmentation.

We carefully explained the meaning of the two main parameters in our algorithm, \( p \) and \( \varepsilon \), and the trade-offs they express. We gave a theoretical basis for selecting an appropriate value of \( p \). Needless to say that these parameters are fixed throughout the paper. We described a preliminary theory which motivated the notion of global dimension, and we justified why it makes sense as an objective function in our application.

Finally, we presented an outlier detection/rejection framework for GDM. We explored two complimentary implementations of this framework, and we presented results demonstrating that it is competitive at handling outliers in this application.

8 Appendix

8.1 Proof of Theorem 11

We prove the four properties of the statement of the theorem. For simplicity we assume that \( D < N \). That is, the number of data points is greater than the dimension of the ambient space. This is the usual case in many applications.

Proof of Property 1: Clearly, scaling all data vectors by \( \alpha \neq 0 \) results in scaling all the singular values of the corresponding data matrix by \( \alpha \). Furthermore, this results in scaling by \( \alpha \) both the numerator and denominator of the expression for the empirical dimension for any \( \varepsilon > 0 \). Therefore, the empirical dimension is invariant to this scaling.

Proof of Property 2: The singular values of a matrix (in particular the data matrix) are invariant to any orthogonal transformation of this matrix and thus the empirical dimension is invariant to such transformation.

Proof of Property 3: If \( \{v_i\}_{i=1}^N \) are contained in a \( d \)-subspace, then since these form the columns of \( A \), \( \text{rank}(A) \leq d \). Since \( U \) and \( V \) are orthogonal, \( \text{rank}(A) = \text{rank}(\Sigma) \). In particular, \( A \) has at most \( d \) singular values. Let \( \sigma \) be the vector of singular values of \( A \), and let \( 1_{\sigma} \) be the indicator vector of \( \sigma^2 \).

The generalized Hölder’s Inequality [19, pg. 10] states that if:

\[
p_1, p_2 \in (0, \infty] \quad \text{and} \quad \frac{1}{p_1} + \frac{1}{p_2} = \frac{1}{r}
\]

then

\[
\|f_1 f_2\|_r \leq \|f_1\|_{p_1} \|f_2\|_{p_2} \quad \text{for any functions } f_1 \text{ and } f_2.
\]

To apply this result to vectors, we view them as functions over the set \( \{1, 2, \ldots, D\} \) with counting measure.

Let \( p_1 = 1 \), \( p_2 = \frac{r}{r-1} \), \( r \in \varepsilon \). Also let \( f_1 = 1_{\sigma} \), \( f_2 = \sigma \). These values satisfy (13). We therefore get:

\[
\left\| \sigma \right\|_{\frac{r}{r-1}} \leq \left\| 1_{\sigma} \right\|_1 = (\# \text{ of non-zero sing. values of } A) \leq d.
\]

Proof of Property 4: By hypothesis, the data vectors \( \{v_i\}_{i=1}^N \) are i.i.d. and sampled according to probability measure \( \mu \), where \( \mu \) is sub-Gaussian, non-degenerate, and spherically symmetric in a \( d \)-subspace of \( \mathbb{R}^D \). We define the \( n \)th data matrix:

\[
A_n = \begin{bmatrix}
\uparrow & \uparrow & \uparrow & \cdots & \uparrow \\
\downarrow & \downarrow & \downarrow & & \\
v_1 & v_2 & v_3 & \cdots & v_n
\end{bmatrix}.
\]

Then \( \Sigma_n := \frac{1}{n} A_n A_n^T \) is the \( n \)th sample covariance matrix of our data set. Also, let \( \nu \) be a random variable with probability measure \( \mu \). Then \( \Sigma := E[\nu \nu^T] \) is the covariance matrix of the distribution. A consequence of \( \mu \) being spherically symmetric in a \( d \)-subspace is that after an appropriate rotation of space, \( \Sigma \) is diagonal with a fixed constant in \( d \) of its diagonal entries and 0 in all other locations. We are trying to prove a result about empirical dimension, which is scale invariant and invariant under rotations of space. Because of these two properties we can assume that the appropriate rotation and scaling has been done so that \( \Sigma \) is diagonal with value 1 in \( d \) diagonal entries and 0 in all others. Without any loss of generality, we assume that the first \( d \) diagonal entries are the non-zero ones.

Let \( \sigma_n = (\sigma_{n,1}, \sigma_{n,2}, \ldots, \sigma_{n,D})^T \), \( n \geq D \), denote the vector of singular values of the matrix \( A_n \). Our first task will be to show that \( \frac{\sigma_n^2}{\sqrt{n}} \) converges in probability (as \( n \to \infty \)) to the vector:

\[
(1, 1, \ldots, 1, 0, \ldots, 0)^T.
\]

To accomplish our task, we will first relate \( \sigma_n \) to the vector of singular values of \( \Sigma_n \), and then use a result showing that \( \Sigma_n \) converges to \( \Sigma \) as \( n \to \infty \).

It is clear that the vector of singular values of \( \Sigma_n \), which we will denote by \( \psi \), is given by:

\[
\psi = \frac{1}{n} (\sigma_{n,1}^2, \sigma_{n,2}^2, \ldots, \sigma_{n,D}^2)^T.
\]

Next, we will need the following result regarding covariance estimation. This is Corollary 5.50 of [43], adapted to be consistent with our notation.
Lemma 1 (Covariance Estimation): Consider a sub-Gaussian distribution in $\mathbb{R}^D$ with covariance matrix $\Sigma$. Let $\gamma \in (0, 1)$, and $r \geq 1$. If $n > C(t/\gamma)^2 D$, then with probability at least $1 - 2e^{-t^2D}$, $\|\Sigma_n - \Sigma\|_2 \leq \gamma$, where $\| \cdot \|_2$ denotes the spectral norm (i.e., largest singular value of the matrix). The constant $C$ depends only on the sub-Gaussian norm of the distribution.

In our problem, we are applying this lemma to the distribution $\mu$. Let $\gamma \in (0, 1)$ be given. If

$$n > C(t/\gamma)^2 D,$$  

then $\|\Sigma_n - \Sigma\|_2 \leq \gamma$ with probability at least $1 - 2e^{-t^2D}$. The 2-norm of the difference of two matrices bounds the differences of their individual singular values. We will use the following result to make this precise:

Lemma 2: Let $\sigma_i(\bullet)$ denote the $i$th largest singular value of an arbitrary $m$-by-$n$ matrix. Then: $|\sigma_i(B + E) - \sigma_i(B)| \leq \|E\|_2$, for each $i$.

Because $\Sigma$ is diagonal with only values 1 and 0 on the diagonal, the singular values of $\Sigma$ are simply these diagonal values. We will use $I_{i \in 1:d}$ to denote the $i$th singular value of $\Sigma$.

Setting $B = \Sigma_n$ and $E = \Sigma - \Sigma_n$, in lemma 2 we get: $\|\Sigma_n - \Sigma\|_2 \leq \gamma$ \Rightarrow \|1/n\sigma_{i\leq d} - I_{i \in 1:d}\| \leq \|\Sigma_n - \Sigma\|_2 \leq \gamma$, for each $i$. This implies that:

$$\frac{\sigma_{i\leq d}}{\sqrt{n}} \in \begin{cases} [\sqrt{1 - \gamma}, \sqrt{1 + \gamma}], & \text{if } i \leq d; \\ [0, \sqrt{\gamma}], & \text{if } i > d. \end{cases}$$ (19)

Notice that as $\gamma \rightarrow 0$, $\sigma_{i\leq d}/\sqrt{n}$ approaches $I_{i \in 1:d}$. Specifically, for any desired tolerance, $\eta > 0$, and any desired certainty, $\xi$, $n$, can be chosen large enough that with probability greater than $\xi$, $\forall i \in 1:d, \frac{\sigma_{i\leq d}}{\sqrt{n}} < \eta$, simultaneously for each $i$. It follows from this that the vector $\frac{\sigma_{i\leq d}}{\sqrt{n}}$ converges in probability to $I_{i \in 1:d}$ as $n \rightarrow \infty$.

Finally, $\hat{d}_{k,n} = \frac{\|\sigma_{i\leq d} - 1/n\|_2}{\||\sigma_{i\leq d} - 1/n\|_2|} = \frac{(1/n)|\sigma_{i\leq d}|}{(1/n)|\sigma_{i\leq d}|} = \frac{\sigma_{i\leq d}}{\sqrt{n}}$. Thus, $\hat{d}_{k,n}$ is a continuous function of the vector $\frac{\sigma_{i\leq d}}{\sqrt{n}}$. Hence, since $\frac{\sigma_{i\leq d}}{\sqrt{n}}$ converges to $I_{i \in 1:d}$ as $n \rightarrow \infty$, $\hat{d}_{k,n}$ converges in probability to

$$\left(\frac{\|1,1,\ldots,0,0,\ldots\|_2}{\|1,1,\ldots,1,0,0,\ldots\|_2}\right) = \frac{d^2}{d^2} = d^2 - \frac{d^2}{d^2} = d. \quad (20)$$

8.2 Proof of Theorem 2

Recall that $\Pi_{Nat}$ denotes the natural partition of the data set. First, we notice that $GD(\Pi_{Nat}) = \|(d_1, d_2, \ldots, d_k)\|_p$, where $d_k$ is the true dimension of set $k$ of the partition. Notice that $d_k$ cannot exceed $d$ since $\mu_k$ is supported by $L_k$, a $d$-subspace. Furthermore, since $\mu_k$ does not concentrate mass on subspaces it is a probability 0 event that all $N_k$ points from $L_k$ exist in a proper subspace of $L_k$. Thus, for the natural partition, $d_k$ is almost surely $d$, for each $k$. Hence, $GD(\Pi_{Nat})$ is almost surely $\|(d, d, \ldots, d)\|_p = (Kd^p)^{1/p} = K^{1/p}d$.

Next, we will find a lower bound for the global dimension of any non-natural partition of the data, and show that if $p$ meets the hypothesis criterias, the lower bound we get is greater than $K^{1/p}d = GD(\Pi_{Nat})$. To accomplish this we need the following lemma.

Lemma 1 If $\Pi \neq \Pi_{Nat}$ then $\Pi$ almost surely has one set with dimension at least $d + 1$.

Before proving the lemma, observe that a consequence is that if $\Pi \neq \Pi_{Nat}$, then with probability $1$:

$$GD(\Pi) \geq \|(d, d, \ldots, d)\|_p \geq d + 1.$$

Then, from our hypothesis:

$$p > \ln(K)/(\ln(d+1) - \ln(d)) \Rightarrow \left(\frac{d+1}{d}\right)^p > K \Rightarrow d + 1 > K^{1/p}d.$$ (22)

Hence,

$$GD(\Pi) \geq d + 1 > K^{1/p}d = GD(\Pi_{Nat}).$$ (23)

Thus, if we show Lemma 1 we require an simpler lemma:

Lemma 2 If a set $Q$ in $\Pi$ has fewer than $d$ points from a subspace $L_k$, then either $Q$ has dimension at least $d + 1$ or adding another point from $L_k$ to $Q$ (an R.V. $X$ with probability measure $\mu_k$, independent from all other samples) will almost surely increase the dimension of $Q$ by 1.

Proof If $\dim(Q) \leq d$ then $Q$ has dimension strictly less than the ambient space ($\mathbb{R}^D$). Observe that $\operatorname{span}(Q)$ is a linear subspace of $\mathbb{R}^D$, which a.s. does not contain $L_k$. We cannot have proper containment since $\dim(L_k) = d \geq \dim(Q)$. Also, we have fewer than $d$ points from $L_k$ in $Q$, and each other point in $Q$ lies in $L_k$ with probability 0 (All $\mu_k$ do not concentrate mass on subspaces). Thus, $\operatorname{span}(Q)$ a.s. does not equal $L_k$.

Therefore, if we intersect $L_k$ with $\operatorname{span}(Q)$ we get a proper subspace of $L_k$; call it $\bar{L}$. We note that $\mu_k(\bar{L}) = 0$. Then, with probability $1$, $\operatorname{dim}(\bar{L}) = d + 1$.
since $\mu_i$ does not concentrate on subspaces. Thus, since $X$ has probability measure $\mu_i$, $X$ a.s. lies outside the intersection of $L_i$ and $\text{span}(Q)$. It follows that if we add $X$ to $Q$, the dimension of $Q$ a.s. increases by 1.

Now we prove Lemma 1. We will assume all sets in $\Pi$ have dimension less than $d + 1$ and pursue a contradiction. By hypothesis, our set $\{v_n\}_{n=1}^N$ contains at least $d + 1$ points from each subspace $L_i$. Since $\Pi \neq \Pi_{\text{cut}}$, there is some subspace $L^*$ whose points are assigned to 2 or more distinct sets in $\Pi$. Let $v^*$ be a point from $L^*$. Now, choose $d$ points from each $L_i$ and denote this collection of $Kd$ points $\{y_1, y_2, \ldots, y_{Kd}\}$. When making this selection, ensure that $v^*$ is not chosen and that of the points selected from $L^*$, not all of them are assigned to the same set in $\Pi$ as $v^*$.

Notice that $\Pi$ induces a partition on $\{y_1, y_2, \ldots, y_{Kd}\}$.

Select any point $y_i$ and remove it from the set $\{y_1, y_2, \ldots, y_{Kd}\}$. Since we are assuming that each set in $\Pi$ has dimension less than $d + 1$, Lemma 2 implies that the set in $\Pi$ to which $y_i$ belongs will have its dimension decrease by 1. Now select another point $y_j$ and remove it. Lemma 2 still applies and so the set to which $y_j$ belonged will have its dimension decrease by 1. We can repeat this until all $Kd$ points have been removed. Since each removal decreases the dimension of some set in $\Pi$ by 1 it follows that before any removals the sum of the dimensions of all sets in $\Pi$ was at least $Kd$. Since each of the $K$ sets in $\Pi$ had dimension $d$ or less, we conclude that in fact each set must have dimension exactly $d$.

Now, consider our set $\{y_1, y_2, \ldots, y_{Kd}\}$ and add in $v^*$. By our choice of $v^*$, Lemma 2 implies that its addition a.s. increases the dimension of its target set in $\Pi$ by 1 (to $d + 1$). Adding in remaining points from $\{v_n\}_{n=1}^N$ will only increase the dimensions of the sets in $\Pi$. Thus, we almost surely have a set of dimension at least $d + 1$ in $\Pi$, contradicting our hypothesis.

8.3 Proof of Theorem 3

Recall that the soft partition is stored in a membership matrix $M$. Specifically, the $(k, n)$'th element of $M$, denoted $m_{nk}^\ast$, holds the “probability” that vector $v_n$ belongs to cluster $k$. Thus, each column of $M$ forms a probability vector.

Hence, global dimension is a real-valued function of the matrix $M$. We will think of the membership matrix as being vectorized, so that the domain of optimization can be thought of as a subset of $\mathbb{R}^{NK}$. However, we will not explicitly vectorize the membership matrix. Thus, when we talk about the gradient of global dimension, we are referring to another $K$-by-$N$ matrix, where the $(k, n)$'th element is the derivative of global dimension w.r.t. $m_{nk}^\ast$.

To differentiate global dimension we must be able to differentiate the singular values of a matrix w.r.t. each element of that matrix. A treatment of this is available in [24].

To begin, recall the definition of $GD$:

$$GD = \begin{bmatrix} \frac{\partial \hat{d}_1}{\partial \varepsilon} & \cdots & \frac{\partial \hat{d}_p}{\partial \varepsilon} \end{bmatrix}^T = \left((\hat{d}_1)^p + (\hat{d}_2)^p + \ldots + (\hat{d}_p)^p\right)^{1/p}. \tag{24}$$

We will denote the thin SVD (only $D$ columns of $U$ and $V$ are used) of $A_k$:

$$A_k = U_k \Sigma_k V_k^T. \tag{25}$$

Also, we will let $\sigma_j$ refer to the $(j, j)$'th element of $\Sigma_j$. Then, using the chain rule:

$$\frac{\partial GD}{\partial m_k^n} = \frac{\partial GD}{\partial \hat{d}_1} \frac{\partial \hat{d}_1}{\partial m_k^n} + \frac{\partial GD}{\partial \hat{d}_2} \frac{\partial \hat{d}_2}{\partial m_k^n} + \ldots + \frac{\partial GD}{\partial \hat{d}_p} \frac{\partial \hat{d}_p}{\partial m_k^n}. \tag{26}$$

From [24] we can compute $\frac{\partial GD}{\partial \hat{d}_j}$ rather easily:

$$\frac{\partial GD}{\partial \hat{d}_j} = \frac{1}{p} ((\hat{d}_1)^p + (\hat{d}_2)^p + \ldots + (\hat{d}_p)^p)^{-1/2} \hat{d}_j ((\hat{d}_1)^p + (\hat{d}_2)^p + \ldots + (\hat{d}_p)^p)^{-1}. \tag{27}$$

Next, we expand the other components of (26):

$$\frac{\partial \hat{d}_j}{\partial m_k^n} = \sum_{j=1}^D \frac{\partial \hat{d}_j}{\partial \sigma_j} \frac{\partial \sigma_j}{\partial m_k^n}. \tag{28}$$

We now use the definition of $\hat{d}_j$ to compute the first factor of each term as follows:

$$\frac{\partial \hat{d}_j}{\partial \sigma_j} = \frac{\|\sigma_i\|_\delta \frac{\partial}{\partial \sigma_j} \left((\sigma_j)^{\frac{1}{\delta}} + \ldots + (\sigma_D)^{\frac{1}{\delta}}\right)}{\|\sigma_i\|_\delta^{2/\delta}} = \frac{\|\sigma_i\|_\delta \left((\sigma_j)^{\frac{1}{\delta}} + \ldots + (\sigma_D)^{\frac{1}{\delta}}\right)}{\|\sigma_i\|_\delta^{2/\delta}} \frac{\sigma_j^{1/\delta} - (\sigma_j)^{\epsilon - 1}}{\sigma_j}$$

$$= \left(\frac{1}{\|\sigma_i\|_\delta^{2/\delta}}\right) \frac{\|\sigma_i\|_\delta \|\sigma_i\|_\epsilon^{-\epsilon} (\sigma_j)^{\epsilon - 1}}{\|\sigma_i\|_\delta^{2/\delta}} - \left(\frac{1}{\|\sigma_i\|_\delta^{2/\delta}}\right) \frac{\|\sigma_i\|_\delta \|\sigma_i\|_\epsilon^{1-\epsilon} (\sigma_j)^{\epsilon - 1}}{\|\sigma_i\|_\delta^{2/\delta}}$$

$$= C_1 (\sigma_j)^{\epsilon - 1} - C_2 (\sigma_j)^{\delta - 1}, \tag{29}$$

where $C_1$ and $C_2$ are constants.
where

\[
C_1 = \left( \frac{||\sigma_i|| - \epsilon}{||\sigma_i||} \right) \quad \text{and} \quad C_2 = \left( \frac{||\sigma_i|| - \delta}{||\sigma_i||} \right),
\]

(30)

Next, we must evaluate the second factor in each term of (28). Recall that \(\sigma_j^i\) is the \(j\)th largest singular value of the matrix \(A_i\). To achieve the next step, we must observe that each singular value of \(A_i\) depends, in general, on each element of the matrix \(A_i\). We can then compute the derivative of each element of the matrix \(A_i\) w.r.t. each membership variable, \(m_k^i\). We will denote the \((\alpha, \beta)\)th element of the matrix \(A_i\) by \(A_i(\alpha, \beta)\). Using the chain rule:

\[
\frac{\partial \sigma_j^i}{\partial m_k^i} = \sum_{\beta=1}^{N} \frac{\partial \sigma_j^i}{\partial A_i(\alpha, \beta)} \frac{\partial A_i(\alpha, \beta)}{\partial m_k^i}.
\]

(31)

A powerful result \([34, \text{eqn. 7}]\) allows us to express the partial derivative of each singular value, \(\sigma_j^i\), w.r.t. a given matrix element in terms of the already-known SVD of \(A_i\):

\[
\frac{\partial \sigma_j^i}{\partial A_i(\alpha, \beta)} = U_i(\alpha, j) V_i(\beta, j).
\]

(32)

The second factor in each term of (31) can be evaluated directly from the definition of \(A_i\):

\[
\frac{\partial A_i(\alpha, \beta)}{\partial m_k^i} = \begin{cases} 0, & \text{if } n \neq \beta \text{ or } i \neq k; \\ v_n \cdot e_\alpha, & \text{if } n = \beta \text{ and } i = k, \end{cases}
\]

(33)

where \(e_\alpha\) denotes the \(\alpha\)th standard basis vector (1 in position \(\alpha\) and 0's everywhere else).

We are now in a position to work backwards and construct the partial derivative of GD w.r.t. \(m_k^i\). In what follows, \(\delta_k\) is equal to 1 if \(i = k\) and 0 otherwise (this is not to be confused with the un-subscripted \(\delta\), which is shorthand for \(\epsilon / ((1 - \epsilon))^\epsilon\)). Also, for notational convenience, we use Matlab notation to represent a row or column of a matrix \(B_{(m,:)}\) and \(B_{(:,n)}\), respectively. (33) Fists compute \(\partial \sigma_j^i / \partial m_k^i\) as follows:

\[
\frac{\partial \sigma_j^i}{\partial m_k^i} = \sum_{\alpha=1}^{D} \frac{\partial \sigma_j^i}{\partial A_i(\alpha, \beta)} \frac{\partial A_i(\alpha, \beta)}{\partial m_k^i} = \sum_{\alpha=1}^{D} U_i(\alpha, j) V_i(n, \alpha) (v_n \cdot e_\alpha) \delta_k.
\]

(34)

Then from (28), we get

\[
\frac{\partial d_k^i}{\partial m_k^i} = \sum_{j=1}^{D} \left( C_1 (\sigma_j^i)^{\epsilon-1} - C_2 (\sigma_j^i)^{\delta-1} \right) V_i(n, j) (U_i(:, j) \cdot v_n) \delta_k.
\]

(35)

Now we can write:

\[
\frac{\partial d_k^i}{\partial m_k^i} = C_1 \sum_{j=1}^{D} \frac{d_k^i}{\partial A_i(\alpha, \beta)} \frac{\partial A_i(\alpha, \beta)}{\partial m_k^i} V_i(n, j) (U_i(:, j) \cdot v_n) \delta_k = C_2 \sum_{j=1}^{D} \frac{d_k^i}{\partial A_i(\alpha, \beta)} \frac{\partial A_i(\alpha, \beta)}{\partial m_k^i} V_i(n, j) (U_i(:, j) \cdot v_n) \delta_k.
\]

(36)

We now simplify the components of (36). After some manipulation, and using the notation

\[
(\Sigma_j^i)_{\epsilon-1} = \left[ \begin{array}{ccc} (\sigma_1^i)^{\epsilon-1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & (\sigma_D^i)^{\epsilon-1} \end{array} \right],
\]

(37)

we can write

\[
\sum_{j=1}^{D} \left( (\sigma_j^i)^{\epsilon-1} V_i(n, j) (U_i(:, j) \cdot v_n) = V_i(n, :) (\Sigma_j^i)^{\epsilon-1} (U_j^T v_n), \right.
\]

(38)

Similarly, we can simplify part of the second term of (36):

\[
\sum_{j=1}^{D} \left( (\sigma_j^i)^{\delta-1} V_i(n, j) (U_i(:, j) \cdot v_n) = V_i(n, :) (\Sigma_j^i)^{\delta-1} (U_j^T v_n). \right.
\]

(39)

Substituting (38) and (39) into (36) we get

\[
\frac{\partial d_k^i}{\partial m_k^i} = C_1 \left( V_i(n, :) (\Sigma_j^i)^{\epsilon-1} (U_j^T v_n) - C_2 V_i(n, :) (\Sigma_j^i)^{\delta-1} (U_j^T v_n) \right)
\]

(40)

With this expression we are ready to evaluate \(28\) as follows:

\[
\frac{\partial GD}{\partial m_k^i} = \sum_{j=1}^{D} \frac{\partial GD}{\partial d_k^i} \frac{\partial d_k^i}{\partial m_k^i} = \sum_{j=1}^{D} \frac{\partial GD}{\partial d_k^i} \left[ C_1 \left( V_i(n, :) (\Sigma_j^i)^{\epsilon-1} (U_j^T v_n) - C_2 V_i(n, :) (\Sigma_j^i)^{\delta-1} (U_j^T v_n) \right) \right]
\]

(41)
We re-write (40) as follows:

$$\frac{\partial GD}{\partial m_k} = V_{k(n,:)} \left( (d_k^p)^{p-1} \| d_k^p \|_p \| U_k^T \|_p^p \right) A_{(\cdot, n)}$$

(42)

8.4 Experiment Setup

For our comparison on the outlier-free RAS database, we include the following methods: GDM, SCC [12] from www.math.umn.edu/~lerman/sc, MAPA [13] from www.math.duke.edu/~glchen/mapa.html, SSC [15] (version 1.0 based on CVX) from wwwvision.jhu.edu/code, SLBF (& SLBF-MS) [51] from www.math.umn.edu/~lerman/lbf, LRR [29] from sites.google.com/site/guangcanliu, RAS [35] (obtained directly from the authors), and HOSC [2] from www.math.duke.edu/~glchen/hosc.html. For each method in our comparisons (outlier-free and our tests with outliers) the implementation of each algorithm is that of the original authors. Most of these codes were found on the respective authors’ websites, although some codes were obtained from the authors directly when they could not be found online. As a matter of good testing methodology, we ran each method 10 times on each file. This is because we want to avoid capturing any fluke occurrences of any method, but instead seek the “usual case” results (this is important for repeatability of the results). Of the 10 runs for a given file and method, the median error is reported. For deterministic methods, we get the same exact results for a given file and method, the median error is reported.

For our comparison on the outlier-free Hopkins 155 database, the algorithms that were selected for the comparison were run once on each of the 155 data files. The mean and median performance for each category is reported. GDM was run with $n_1 = 30$ to improve reliability. All other parameters were left fixed, and (as before) the non-linearly embedded data was used. Each competing 2-view method was run on the non-linearly embedded data with the same parameters that gave the best performance on the RAS database. The competing n-view methods have their parameters given in the results tables.

For our outlier comparison on the corrupted RAS database, we ran GDM with $n_1 = 30$ (same as for the Hopkins 155 database). For the naive approach, we used $\alpha = 0.02$. For “GDM - Known Fraction” we rejected 20% of the dataset. For “GDM - Model Reassign” we used $\kappa = 0.05$.

For our comparison on the outlier-free Hopkins 155 database, the algorithms that were selected for the comparison were run once on each of the 155 data files. The mean and median performance for each category is reported. GDM was run with $n_1 = 30$ to improve reliability. All other parameters were left fixed, and (as before) the non-linearly embedded data was used. Each competing 2-view method was run on the non-linearly embedded data with the same parameters that gave the best performance on the RAS database. The competing n-view methods have their parameters given in the results tables.

For our outlier comparison on the corrupted RAS database, we ran GDM with $n_1 = 30$ (same as for the Hopkins 155 database). For the naive approach, we used $\alpha = 0.02$. For “GDM - Known Fraction” we rejected 20% of the dataset. For “GDM - Model Reassign” we used $\kappa = 0.05$.

The code for GDM can be found on our supplemental webpage. For each of the algorithms used in our comparisons, we have made an effort to provide (on the supplemental webpage) the code or a link to where the code can be found.

Acknowledgements This work was supported by NSF grants DMS-09-15064 and DMS-09-56072. GL was partially supported by the IMA during their annual program on the mathematics of information (2011-2012) and BP benefited from participating in parts of this program and even presented an initial version of this work at an IMA seminar in Spring 2012. We thank the anonymous reviewers for their thoughtful comments and Tom Lou for his helpful suggestions in regards to our algorithm for minimizing global dimension. A very preliminary version of this work was submitted to CVPR 2012, we thank one of the anonymous reviewers for some insightful comments that made us modify the GDM algorithm and its theoretical support.

References

1. Aldroubi, A.: A review of subspace segmentation: Problem, non-linear approximations, and applications to motion segmentation.
46. Vidal, R., Ma, Y., Soatto, S., Sastry, S.: Two-view multibody structure from motion. International Journal of Computer Vision 68(1), 7–25 (2006)

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

48. Yang, A.Y., Rao, S.R., Ma, Y.: Robust statistical estimation and segmentation of multiple subspaces. In: CVPRW ’06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop, p. 99. IEEE Computer Society, Washington, DC, USA (2006). DOI http://dx.doi.org/10.1109/CVPRW.2006.178

49. Zhang, T., Szlam, A., Lerman, G.: Median K-flats for hybrid linear modeling with many outliers. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on Computer Vision, pp. 234–241. Kyoto, Japan (2009). DOI 10.1109/ICCVW.2009.5457695

50. Zhang, T., Szlam, A., Wang, Y., Lerman, G.: Randomized hybrid linear modeling by local best-fit flats. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 1927–1934 (2010). DOI 10.1109/CVPR.2010.5539866

51. Zhang, T., Szlam, A., Wang, Y., Lerman, G.: Hybrid linear modeling via local best-fit flats. International Journal of Computer Vision 100, 217–240 (2012). DOI 10.1007/s11263-012-0535-6