Efficient Higher-Order Clustering on the Grassmann Manifold

Suraj Jain
Microsoft Research
suraj.jain28@gmail.com

Venu Madhav Govindu*
Department of Electrical Engineering
Indian Institute of Science, Bengaluru, INDIA
venu@ee.iisc.ernet.in

Abstract

The higher-order clustering problem arises when data is drawn from multiple subspaces or when observations fit a higher-order parametric model. Most solutions to this problem either decompose higher-order similarity measures for use in spectral clustering or explicitly use low-rank matrix representations. In this paper we present our approach of Sparse Grassmann Clustering (SGC) that combines attributes of both categories. While we decompose the higher-order similarity tensor, we cluster data by directly finding a low dimensional representation without explicitly building a similarity matrix. By exploiting recent advances in online estimation on the Grassmann manifold (GROUSE) we develop an efficient and accurate algorithm that works with individual columns of similarities or partial observations thereof. Since it avoids the storage and decomposition of large similarity matrices, our method is efficient, scalable and has low memory requirements even for large-scale data. We demonstrate the performance of our SGC method on a variety of segmentation problems including planar segmentation of Kinect depth maps and motion segmentation of the Hopkins 155 dataset for which we achieve performance comparable to the state-of-the-art.

1. Introduction

In computer vision, the most common scenario requiring higher-order clustering arises when observation vectors are drawn from multiple subspaces of a given vector space or when data fits multi-dimensional parametric forms. For instance, see Fig. 1 where the observed points need to be clustered into 5 independent circles. Evidently, such clustering cannot be achieved using conventional methods that measure distances between or similarities of point pairs.

*Corresponding Author. This work was carried out when Suraj Jain was a graduate student at the Indian Institute of Science.

1.1. Review of Literature

Since the literature on clustering is very extensive, in the following we will limit ourselves to a brief review of methods that are of directly relevance to our approach for higher-order clustering. For a recent survey of methods for subspace clustering the reader may refer to [16]. All the methods we consider use spectral clustering, see [12] for a tutorial on spectral clustering and [17] for a discussion of related methods in computer vision. The methods of interest can be classified into two categories. Methods in the first category work with higher-order similarity measures [8, 3, 1, 13]. [8] defines a similarity measure for an n-tuple of data points which leads to a multiway similarity tensor. A similarity matrix is estimated by sampling columns from the flattened form of the similarity tensor and spectral clustering is applied. The Spectral Curvature Clustering (SCC) method of [3], *inter alia*, presented an iterative sampling technique that significantly improved upon the uniform sampling of [8].

While the spectral clustering approach works with a few eigen-vectors of the similarity matrix, our second category of methods explicitly model the low-rank nature of similarity representations. In a dataset, even when the number of points (N) is large, often the number of clusters (K) is far smaller, i.e. $K \ll N$. Methods that utilise the low-rank property of similarity matrices include [6, 18, 11, 7]. In [18], the low-rank representation of observed data is combined with a locality property whereby points in a neighbourhood tend to lie in the same linear subspace. In [11] (LRR), subspace clustering is achieved by finding the lowest-rank representation of the data matrix using itself as the dictionary. Similarly, the Sparse Subspace Clustering (SSC) method of [6] expresses each observation as the sparsest linear combination of the other observations that act as a dictionary. The Low-Rank Subspace Clustering (LRSC) method of [7] has a similar goal of building a sparse representation, except that LRSC assumes that the observations are obtained by adding noise to a clean dictionary. All these methods exploit recent ad-

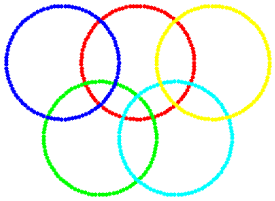


Figure 1. The overlapping circles cannot be clustered into 5 different geometric models using standard methods.

vances in convex optimization and can achieve the desired clustering in the presence of outliers. While the methods of [18, 11, 6] can effectively solve the clustering problem, they are computationally expensive. For instance, SSC solves an optimization problem for each data point which can become prohibitively large as N grows. Additionally, while using different approaches, these methods eventually build a similarity matrix for spectral clustering, implying that the memory and computational load for these methods can become very large. All of these factors imply that despite the desirable property of robustness to outliers, the methods of [18, 11, 6] cannot solve the clustering problem for large datasets.

Our approach of Sparse Grassmann Clustering (SGC) presented in this paper shares attributes with both of the abovementioned categories. For representing the relationships between data points, we use the higher-order similarities of [8, 3] that is discussed in Sec. 2. For efficient computation, we develop an approximation to the similarity measures that is presented in Sec. 3. Instead of applying spectral clustering to a similarity matrix, we utilise its low-rank properties to directly solve for clustering without explicitly building a similarity matrix. This allows for an efficient and scalable method that can handle large amounts of data. In Sec. 4 we develop our arguments for this purpose that relies on the geometry of the Grassmann manifold and discuss the GROUSE algorithm that provides an efficient method for estimation on the Grassmann manifold. In Sec. 5 we present our method of Sparse Grassmann Clustering (SGC) and the results of applying our method on a variety of problems are presented in Sec. 6.

2. Higher-Order Grouping using Tensors

In this Section, we summarise the higher-order tensor decomposition of similarity relationships as presented in [8, 3]. We are given data points $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ in $\mathbb{R}^D$ where N is the number of observations and D is the dimensionality of the observed feature vectors. Further, we assume that data is to be classified into K clusters, where $K \ll N$. The $N \times N$ similarity matrix $\mathbf{S}$ uses the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$,

$$\mathbf{S}(i,j) = e^{-\frac{\phi^2(\mathbf{x}_i,\mathbf{x}_j)}{2\sigma^2}} \tag{1}$$

where $\phi(.,.)$ is an appropriate distance measure between two feature vectors. The spectral clustering problem is solved by using the K leading eigen-vectors $U = \{\mathbf{u}_1, \cdots, \mathbf{u}_K\}$ of $\mathbf{S}$ or equivalently in terms of a normalized Laplacian form. While there are many variants, a common solution uses the rows of the $N \times K$ matrix U, i.e. the i-th row of U is the feature vector for $\mathbf{x}_i$ which are clustered using K-means [12].

The similarity measure of Eqn. 1 cannot account for higher-order similarities. Consider the illustrative example in Fig. 1 of points belonging to $K = 5$ circles, i.e. each point $\mathbf{x}_i$ is drawn from a 3-parameter geometric model (two for centre and one for radius). For clustering data into circles, $\phi(\mathbf{x}_i, \mathbf{x}_j)$ will always be equal to zero since we can always find a circle that passes through any two points $\mathbf{x}_i$ and $\mathbf{x}_j$. In general, if we wish to cluster data according to a d-parameter model, we need $n > d$ independent observations. Similarly, for fitting data to a d-dimensional linear subspace, we need $n > d + 1$ observations as a d-dimensional linear subspace needs $(d + 1)$ parameters to specify it. Therefore, to cluster such data, we need to test whether an n-tuple of points belongs to the same cluster for $n \geq d + 2$. In other words, similarity or affinity cannot be defined for two points at a time but has to be defined for a set of n points at a time. For an n-tuple of points indexed by $\mathcal{S} = \{i_1, \cdots, i_n\}$, i.e. $\mathbf{x}_S = \{\mathbf{x}_{i_1}, \cdots, \mathbf{x}_{i_n}\}$ we can define the n-tuple similarity or affinity tensor as

$$\mathcal{P}(i_1, \cdots, i_n) = \begin{cases} e^{-\lambda \left| \frac{\phi(\mathbf{x}_S)}{\sigma} \right|^p}, & \text{if } \mathcal{S} \text{ are distinct} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $\phi(\mathbf{x}_S)$ is the residual error of fitting a d-dimensional model to the n-tuple of distinct points $\mathbf{x}_S$, i.e. $\mathcal{P}(i_1, \cdots, i_n)$ is the probability the n-tuple $\mathbf{x}_S$ belongs to the same cluster. In Eqn. 2, σ is the expected noise level for the fit, λ is an appropriate scale factor and the exponent p allows for a greater variety of probability models than the Gaussian distribution for $p = 2$.

Evidently, the problem of clustering using the $N \times N \times \cdots \times N$ similarity tensor $\mathcal{P}$ cannot be addressed by spectral clustering developed for similarity matrices. In [8], the problem of higher-order grouping was solved by decomposing the similarity tensor $\mathcal{P}$ into a two-dimensional similarity matrix $\mathbf{S}$. Utilising the fact that $\mathcal{P}$ is an n-dimensional super-symmetric tensor, [8] showed that conventional spectral clustering can be applied to the similarity matrix $\mathbf{S} = \mathbf{P}\mathbf{P}^T$ where $\mathbf{P}$ is the flattened matrix form of

the tensor $\mathcal{P}$, i.e.

$$\mathbf{S} = \mathbf{P}\mathbf{P}^T = \sum_{j=1}^{N_c} \mathbf{P}(:,j)\mathbf{P}(:,j)^T \tag{3}$$

From Eqn. 3 we note that $\mathbf{S}$ is constructed by adding up the individual rank-1 outer product forms of the columns of $\mathbf{P}$. Each column of $\mathbf{P}$ is obtained by fixing $(n-1)$ indices of the tensor $\mathcal{P}$ and varying the n-th index from 1 to N. Accordingly, each column index 'j' corresponds to a unique choice of $(n-1)$ distinct indices $\mathcal{I} = \{i_1, \cdots, i_{n-1}\}$ and the n-tuple similarity is defined as

$$\begin{aligned} \mathbf{P}(i,j) &= \mathcal{P}(i,\mathcal{I}), \forall i \notin \mathcal{I} \\ \Rightarrow \mathbf{P}(i,j) &= Prob(\{\mathbf{x}_i, \mathbf{x}_\mathcal{I}\}|cluster) \end{aligned} \tag{4}$$

where $Prob(.|cluster)$ is the probability that the set of points belong to the same cluster as defined by Eqn. 2. Since the index set $\mathcal{I}$ corresponds to a selection of $(n-1)$ distinct points, we have $N_c = {}^N C_{n-1}$ choices of $\mathcal{I}$, each of which corresponds to a column of $\mathbf{P}$. It must also be noted that the probability of a set of points belonging to the same cluster is independent of their ordering, i.e. $\mathcal{P}(i_1, \cdots, i_n) = \mathcal{P}(\pi(i_1, \cdots, i_n))$ for all permutations $\pi(.)$. Owing to this property of $\mathcal{P}$, known as super-symmetry, we can choose to fix any of the $(n-1)$ indices, and that $\mathbf{S}$ is independent of the ordering of the column index 'j' in Eqn. 3.

Although $\mathbf{P}$ is of an exceedingly large size $N \times N_c$, the similarity matrix $\mathbf{S} = \mathbf{P}\mathbf{P}^T$ is a low-rank matrix of effective rank equal to $K \ll N$. As a result, [8] argued that the low-rank matrix $\mathbf{S}$ can be efficiently estimated by randomly selecting a few columns of $\mathbf{P}$ which are one-dimensional 'fibres' of $\mathcal{P}$, i.e.

$$\mathbf{S} \approx \sum_{j \in \mathcal{C}} \mathbf{P}(:,j)\mathbf{P}(:,j)^T \tag{5}$$

where $\mathcal{C}$ is a subset of integers selected from between 1 and N_c. In [8], the author proposed to randomly select columns of $\mathbf{P}$ with uniform probability to construct the similarity matrix $\mathbf{S}$. Although efficient, such uniform sampling suffers from some serious drawbacks. We distinguish between *pure* and *mixed* columns as columns that draw $\mathcal{I}$ from a single cluster or multiple clusters respectively. Ideally, we would like to construct $\mathbf{S}$ only from pure columns but a uniform sampling strategy fails in this regard, especially for large N. Thus, in Fig. 1 if each circle has 100 points and we let $n = 5$, then only $\frac{5 \times {}^{100}C_4}{{}^{500}C_4} = 0.76\%$ of columns are pure, implying that uniform sampling would not work well. To ameliorate this situation, the SCC method of [3] proposed an iterative method to refine the sampling of columns. SCC starts by uniformly sampling the columns and clustering the data. Subsequently, a fixed number (say 1000) of columns are drawn from within the current cluster. Using these columns, the clusters are updated and the process is repeated till convergence. This approach progressively refines the sampling to increasingly draw from the set of pure columns. The reader may refer to [3] for details. While the SCC method of [3] does effectively address the sampling drawbacks of [8], both these methods suffer from some significant limitations that we address in this paper.

3. Approximating Similarity Measures

It will be noted from Eqn. 5 that although the approximation for $\mathbf{S}$ uses fewer columns of $\mathbf{P}$ than the full estimate of Eqn. 3, the computational load of estimating the entries in an individual column of $\mathbf{P}$ can be significant. We recall here that a single column of $\mathbf{P}$ is nothing but the entries in the similarity tensor $\mathcal{P}(i,\mathcal{I})$, where $\mathcal{I}$ is the selected set of $(n-1)$ distinct indices and i is the index of the column. Now for all $i \notin \mathcal{I}$, computing $\mathcal{P}(i,\mathcal{I})$ involves estimating the probability of n points belonging to a single cluster. Let us compare two individual entries in a given column, say the r-th and s-th entries, i.e. $\mathcal{P}(r,\mathcal{I})$ and $\mathcal{P}(s,\mathcal{I})$ respectively. These two similarity measures differ only to the extent that 1 out of n data points are different, i.e. $\mathbf{x}_r$ is replaced by $\mathbf{x}_s$. The remaining $(n-1)$ data points indexed by $\mathcal{I}$ are common to all estimates in a given column of $\mathbf{P}$. As a result, for the $N-(n-1)$ non-zero entries in a column of $\mathbf{P}$, all similarity measures are estimated by changing only one point $\mathbf{x}_i$ out of n points. Such repeated estimation on n-tuple data points with $(n-1)$ common data points is wasteful and the similarity measure can be efficiently approximated in the following manner.

Since $n \geq (d+2)$, the data points indexed by $\mathcal{I}$ will themselves fit the parametric model as $(n-1) \geq d+1$ data points are sufficient to fit a d-dimensional parametric or subspace model. We denote the parameters of the fitted model for a given set of data points as $\theta(\mathcal{I})$ indicating that the parameters θ are obtained by fitting the data indexed by the set $\mathcal{I}$, i.e. $\theta(\mathcal{I})$ denotes the model estimated for data points $\{\mathbf{x}_{i_1}, \cdots, \mathbf{x}_{i_{n-1}}\}$. Therefore, we can efficiently approximate the similarity $\mathcal{P}(i,\mathcal{I})$ as the likelihood of the individual points $\mathbf{x}_i$ belong to the parametric model defined by the index set $\mathcal{I}$, i.e.,

$$\begin{aligned} \mathcal{P}(i,\mathcal{I}) &= Prob(\mathbf{x}_i, \mathbf{x}_{i_1}, \cdots, \mathbf{x}_{i_{n-1}}|cluster) \\ &\approx Prob(\mathbf{x}_i|\theta(\mathcal{I})) \end{aligned} \tag{6}$$

It will be noted that Eqn. 6 results in a significant reduction in computational load since for each column in $\mathbf{P}$ we compute a single θ and then score all other points in X with respect to θ. Consider our example in Fig. 1 with $n = 5$ for $N = 500$ points. For each column in $\mathbf{P}$, the SCC method of [3] will select 4 points to form the set $\mathcal{I}$ and estimate the similarity for all other points by carrying out the circle fitting for $n = 5$ points, a total of $N - (n - 1) = 496$ times. In contrast, for our approach in Eqn. 6, we fit a circle *only once* and then simply measure the distance of all the remaining 496 points from this circle. This results in substantial savings of computation especially in the case where the number of data points N is large or when the cost of fitting a model to data is high. It may also be noted here that although Eqn. 6 is an approximation to the true similarity measure, when we select a *pure* set of indices $\mathcal{I}$, the resulting model $\theta(\mathcal{I})$ will be close to the true model. As a result, the approximated similarity of $Prob(\mathbf{x}_i|\theta(\mathcal{I}))$ will be an accurate reflection of whether $\mathbf{x}_i$ belongs to the same cluster as $\mathcal{I}$. To summarise our argument here, since we will be using an iterative sampling technique to refine the selected columns to be drawn from *pure* cluster models, Eqn. 6 accurately represents the underlying similarity measure while greatly reducing the computational expense involved when compared to the computational load of estimating individual entries of a column in $\mathbf{P}$ as defined in both [8] and [3].

4. Low Rank Representations for Clustering

As mentioned in Sec. 1, much of the recent work on higher-order grouping presents different methods that build a similarity matrix between data points and then use spectral clustering. As a result, all of them suffer from an important limitation in that these methods do not scale well with increasing amounts of data. We recall that for a dataset of N points, spectral clustering requires an $N \times N$ similarity matrix $\mathbf{S}$ of which the K leading eigen-vectors are estimated. It is evident that as the data size N increases, so does the requirement for storage of $\mathbf{S}$ as well as the computational load for both estimating the entries of $\mathbf{S}$ and its eigen-decomposition.

Although $\mathbf{S}$ is of size $N \times N$, we note that it is a low-rank matrix with its effective rank equal to the number of data clusters, K. In the following we describe our second contribution that builds on this observation of the low-rank property of $\mathbf{S}$ to greatly reduce both memory and computational requirements, thereby making our method efficient and scalable with increasing amounts of data. If $\mathbf{S}$ is of rank K, from Eqn. 5 it is evident that $\mathbf{P}$ is itself a rank-K matrix. Moreover, if $U = \{\mathbf{u}_1, \cdots, \mathbf{u}_K\}$ are the eigen-vectors of $\mathbf{S}$, U is also a K-dimensional basis of the column space of matrix $\mathbf{P}$, implying that we could solve for clustering using $\mathbf{P}$ instead of $\mathbf{S}$. However, for large N, the eigen-decomposition of even the $N \times \mathcal{C}$ approximation of $\mathbf{P}$ used in Eqn. 5 is expensive. Nevertheless, we can exploit the geometry of low-rank matrices to derive an efficient method for estimating U. Our solution utilises the geometric structure of the Grassmann manifold which we briefly describe below. In Sec. 4.2, we describe the GROUSE algorithm that we use for the requisite computation on the Grassmann manifold.

4.1. Grassmann Manifolds

A natural representation for low-rank subspaces of a vector space is the Grassmann manifold. For a vector space $\mathbb{R}^N$, the space of all linear subspaces of K dimensions forms a compact Riemannian manifold known as the Grassmann manifold which is denoted as $\mathcal{G}(N, K)$. The Grassmann manifold is also identified as the quotient space of the Stiefel manifold $\mathcal{V}(N, K)$, i.e. $\mathcal{V}(N, K)\backslash\mathcal{O}(K)$. More informally, every point on the Grassmann manifold $\mathcal{G}(N, K)$ corresponds to a K-dimensional subspace of $\mathbb{R}^N$ and can be represented by an $N \times K$ matrix U where the columns of U span a K-dimensional subspace. Therefore if $U \in \mathcal{G}(N, K)$, so is UR where $R \in \mathcal{SO}(K)$ the space of all $K \times K$ orthonormal matrices.The reader may refer to [5, 4] for details on properties and computations on Grassmann manifolds. For our purposes, we note that since $\mathbf{P}$ is an $N \times \mathcal{C}$ matrix of effective rank equal to K, the eigen-vectors $U = \{\mathbf{u}_1, \cdots, \mathbf{u}_K\}$ that we seek span a K-dimensional subspace of $\mathbb{R}^N$, i.e. a point on the Grassmann manifold $\mathcal{G}(N, K)$. Therefore, our clustering problem is equivalent to identifying a point $U \in \mathcal{G}(N, K)$.

4.2. The GROUSE Algorithm

In recent years, geometric estimation on the Grassmann manifold has attracted a fair amount of attention [5, 4, 15, 2, 10]. Of particular interest to us is the GROUSE algorithm described in [2] that provides a highly efficient method of estimating a Grassmann representation for the column space of a given matrix, i.e. $\mathbf{P}$ in our case. GROUSE was specifically designed to track a K-dimensional subspace of $\mathbb{R}^N$ and uses matrix columns as input to incrementally update the subspace U on the Grassmann manifold $\mathcal{G}(N, K)$. Since GROUSE processes individual columns of $\mathbf{P}$ one at a time, it does not suffer from the large memory requirement of storing the $N \times N$ matrix $\mathbf{S}$ or $N \times \mathcal{C}$ matrix $\mathbf{P}$ as is required by the methods that use spectral clustering. We can compute individual columns of $\mathbf{P}$ on the fly and the only storage requirement is for U of size $N \times K$ where $K \ll N$. Moreover, the low-rank property of $\mathbf{P}$ implies that its entries are

redundant. In a manner similar to recent advances in matrix completion, GROUSE can efficiently learn the basis U even when only some of the entries of the columns of $\mathbf{P}$ are available. This is a great advantage for our problem since by using GROUSE, we need not compute all the entries of a column of $\mathbf{P}$. In the following we briefly describe the GROUSE algorithm, see [2] for details.

Given $U \in \mathcal{G}(N, K)$, the optimal fit for the $N \times \mathcal{C}$ observation matrix $\mathbf{P}$ can be defined as

$$F(U) = \min_{\mathbf{W} \in \mathbb{R}^{K \times \mathcal{C}}} ||U\mathbf{W} - \mathbf{P}||^2 \tag{7}$$

We can now adapt this cost function to incrementally update U as we observe the columns of $\mathbf{P}$ one at a time. Given a current U, let us observe a new column $\mathbf{P}(:, j)$ for a randomly selected index set $\mathcal{I}$. In the following, for notation convenience we shall denote $\mathbf{P}(:, j)$ as $\mathbf{p}_j$. For the new observed column $\mathbf{p}_j$, the optimal co-efficients $\mathbf{w}$ are given by the minimiser of $F(U, j) = ||U\mathbf{w} - \mathbf{p}_j||^2$. We can update U to account for the information about the subspace provided by $\mathbf{p}_j$ by using a gradient descent step on the Grassmann manifold $\mathcal{G}(N, K)$. The gradient on $\mathcal{G}(N, K)$ (see Eqn. 2.70 of [5]) can be defined as

$$F(U, j) = \min_{\mathbf{w}} ||U\mathbf{w} - \mathbf{p}_j||^2 \tag{8}$$

$$\Rightarrow \nabla F = (I - UU^T)\frac{dF}{dU} = -2\mathbf{r}\mathbf{w}^T \tag{9}$$

where $\mathbf{r} = (\mathbf{p}_j - U\mathbf{w})$ is the residual vector. We skip some details here for notational simplicity, but in the case where only some entries of $\mathbf{p}_j$ are available, the fitting error $F(U, j)$ and residual vector $\mathbf{r}$ can be suitably modified to be defined using only the available entries of $\mathbf{p}_j$. Using the fact that ∇F is of rank one, [2] shows that the update of the U can be carried out in a remarkably simple fashion. For a gradient descent in the direction ∇F with step size of $\eta > 0$, the updated basis matrix U is given as

$$U(\eta) = U + \left(sin(\sigma\eta)\frac{\mathbf{r}}{||\mathbf{r}||} + (cos(\sigma\eta) - 1)\frac{\mathbf{q}}{||\mathbf{q}||} \right) \frac{\mathbf{w}^T}{||\mathbf{w}||} \tag{10}$$

where σ is the singular value of ∇F and $\mathbf{q} = U\mathbf{w}$. Eqn. 10 is remarkable in its simplicity as this Grassmann Rank-One Update Subspace Estimation (GROUSE) ensures that the updated $U(\eta)$ lies on the Grassmann manifold $\mathcal{G}(N, K)$. For a suitably chosen sequence of step sizes η, the GROUSE algorithm is guaranteed to converge to the locally optimal estimate of U. Since this estimation process only needs to store the $N \times K$ matrix U and can work with a few of the entries in columns $\mathbf{p}_j$, GROUSE has a low memory overhead and is also computationally efficient.

Algorithm 1 Sparse Grassmann Clustering (SGC)

Input: $X = \{\mathbf{x}_1 \cdots \mathbf{x}_N\} \in \mathbb{R}^D$.
Output: Classification of X into K clusters.
Initialisation: Set column sampling to be uniform.

1: **while** Not Converged **do**
2: **for** T trials **do**
3: Randomly select $\mathcal{I}$ from within cluster
4: Construct sparse $\mathbf{p}_j$ by estimating random entries of $\mathcal{P}(i, \mathcal{I})$ using Eqn. 6
5: Update $U(\eta)$ using Eqn. 10
6: **end for**
7: Classify points in X using K-means on rows of U
8: Update sampling to current clusters
9: **end while**

5. Sparse Grassmann Clustering (SGC)

We can now state our algorithm that carries out higher-order clustering by combining an iterative refinement of column sampling with an incremental Grassmann update of U. As we do not need all the entries of a column of $\mathbf{P}$ to update U, we can sparsely estimate a few of the entries in a given column $\mathbf{p}_j$. Our method of Sparse Grassmann Clustering (SGC) can be summarised as given in Algorithm 1. We initially use uniform random sampling to draw columns from $\mathbf{P}$ to build an initial clustering estimate. Subsequently, during each iteration, we draw T columns according to the current cluster estimates. For each column we estimate a few of the entries using the approximate similarity of Eqn. 6. This makes for efficient estimation of individual columns that are used to drive the GROUSE algorithm. After each iteration, we carry out the K-means clustering of the rows of the Grassmann estimate $U \in \mathcal{G}(N, K)$ and update the sampling for the next iteration to be based on the current clusters. This process of interleaved iterative updating of sampling and the incremental Grassmann update of the low-rank representation is carried on till convergence. Convergence is achieved when either the fitting error of the data with respect to the estimated U falls below a threshold or when the change of U between iterations falls below another threshold. Due to the efficiency of the GROUSE algorithm, in all of our experiments we have found that our SGC algorithm rapidly converges to the final clustering estimate. We note that while the initial random sampling produces a rough clustering estimate, over subsequent iterations, this estimate is progressively refined by our algorithm thus yielding increasingly pure columns.

5.1. Properties of the SGC algorithm

Handling multiple subspaces : Most methods that carry out estimation on the Grassmann manifold fit a *single*

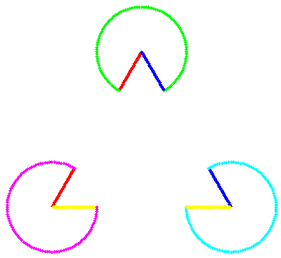
subspace to the observations. The GROUSE algorithm of [2] that we use is itself an example method of this type. Other such instances include matrix completion [10] and GRASTA which is a robust version of GROUSE [9]. A novel attribute of our SGC algorithm is that we are able to carry out classification of data observations into *multiple* subspaces using estimation on the Grassmann manifold. Unlike other methods that fit a single subspace to the observations, we map the observations into a similarity representation that measures the similarity or affinity relationships between data points. Thus, although the raw data points belong to a union of K subspaces, independent of K, our SGC method maps the data points into a representation that lies in a *single* subspace $U \in \mathcal{G}(N, K)$. Thus, we are able to use the power of Grassmann estimation to solve the multiway clustering problem.

Memory and computational requirements : From the arguments of [2], we can see that for an N point dataset to be clustered into K subspaces, each iteration of our SGC algorithm takes $O(TNK + \rho TK^2)$ flops where ρ is the number of entries estimated in each sampled column. In terms of storage requirements, our approach is extremely light-weight as it does not need to store the full $N \times N$ similarity matrix. Instead we only need to store the $N \times K$ matrix $U \in \mathcal{G}(N, K)$ and one sparse column at a time. Since $K \ll N$, as we shall demonstrate in Sec. 6 our SGC method can solve for the clustering of very large problems which cannot be handled by other approaches based on spectral clustering. It is germane to remark here that the notion of sparsity in our SGC algorithm applies to the fact that we only need a sparse number of entries used in individual columns and this should not be confused with the nature of sparsity in methods such as SSC [6].

Intrinsic robustness : Our SGC algorithm is also intrinsically robust to the presence of outliers in the data. From Eqn. 5 we see that the similarity between the i-th and k-th data point is given by $\mathbf{S}(i,k) = \sum_{j \in \mathcal{C}} \mathbf{P}(i,j)\mathbf{P}(k,j)$. As long as the sampled columns include a sufficient number of pure columns, the estimated similarity $\mathbf{S}(i,k)$ is not affected by the presence of outliers in X, i.e. SGC is robust to outliers. We should emphasise here that the discussion of $\mathbf{S}(i,k)$ is only to explain the behaviour of our method and the SGC algorithm never explicitly estimates the similarity matrix $\mathbf{S}$. We present results demonstrating robustness in Sec. 6.

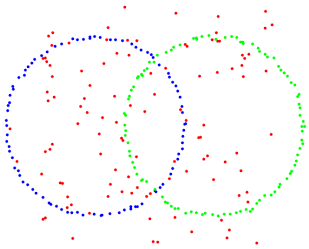
6. Results

In this Section we present some qualitative and quantitative results to demonstrate the efficacy of our Sparse Grassmann Clustering method. In Fig. 1 we show the result of clustering data points according to the model of a circle. As can be seen our method can correctly segment points into 5 circles which cannot be achieved by spectral clustering using Eqn. 1. In Fig. 2, we show the clustering results achieved on the well-known psychophysical model of the Kanizsa triangle. Since we sample points to build a model and score all other points with respect to this model, in our SGC method we can simultaneously cluster data into different types of geometric models. In this instance during one iteration of the for-loop in Algorithm 1 we sample columns according to both the circle and line model. All points in Fig. 2 with the same color belong to the same cluster and as can be seen our SGC method can correctly classify the data. It should be noted that the correct classification of points into circles or lines is automatically achieved using our approach where higher-order similarity information can be integrated across multiple types of models. In addition, our method does not need to know the specific number of lines and circles in the dataset. It is sufficient to know the total number of independent models (lines and circles in this example) and the types of models. The result in Fig. 2 also demonstrates the robustness of our method since, as far as points on circles are concerned, the



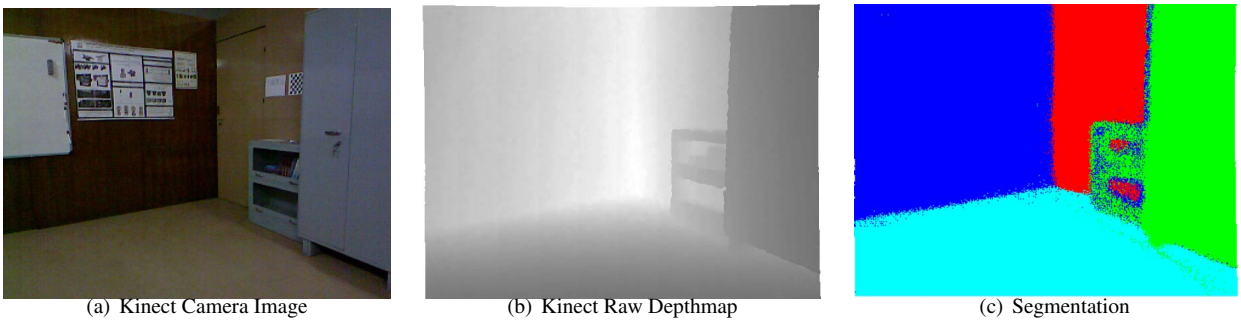
(a) Kanizsa Figure

Figure 2. Clustering results for the Kanizsa figure using our SGC method that can naturally find multiple geometric models (circles and lines in this case) in a single dataset. All points with the same color belong to the same cluster. Please view this figure in color.



(a) Circles with Outliers

Figure 3. Our SGC method is robust to the presence of outliers in the data. See text for details. All points with the same color belong to the same cluster. Please view this figure in color.



(a) Kinect Camera Image (b) Kinect Raw Depthmap (c) Segmentation

Figure 4. **Large scale clustering :** (a) shows the image of an indoor scene given by the Kinect's RGB camera (b) shows the corresponding raw depth map of size (480×640) pixels while (c) show the segmentation of the depth map into planes using our SGC algorithm. Please view this image in color.

points on the lines are outliers and vice-versa. Nevertheless our estimation of classification is unaffected. In Fig. 3 we additionally demonstrate the robustness of our method to the presence of outliers. The input data contains two intersecting circles of 100 points each, interspersed with 100 outlier points, i.e. this dataset has 33% outliers. As can be seen, we achieve good classification of points even in the presence of a large number of outliers. As we set $K = 3$ in this experiment, the data points are necessarily classified into 3 clusters. In our SGC method, the measured similarities between pairs of points that belong to the same circle is high, leading to the correct classification of the two circles as independent clusters. Consequently, all the randomly scattered outlier points are assigned to the third cluster as these points cannot be classified as belonging to either of the two circle clusters. In other words, our approach can correctly detect geometric models even in the presence of a large number of outlier points.

6.1. Plane Segmentation in Depth Maps

In Fig. 4 we show the results of applying our SGC algorithm to find planar regions in the depth map obtained from the popular Kinect sensor. The scene observed is of a room with the walls, floor and cupboard defining multiple 3D planes. We segment a point cloud representation of the depth map into multiple planar regions using our SGC method. In Fig. 4(a) we show a Kinect camera image of the scene. The raw depth map in Fig. 4(b) is of size 640×480, imply that we have $N = 640 \times 480 = 307200$ points to be classified which is very large. However, since we neither store the individual sampled columns of $\mathbf{P}$ nor actually estimate all entries of each column, our segmentation algorithm has a small memory footprint as we only need to estimate $U \in \mathcal{G}(307200, K)$ which is equivalent to storing K depth maps in memory. It will be noted that none of the methods that build the full similarity matrix for segmentation can handle such large problems. In our experiments, we choose $n = 5$, i.e. use 4 points to fit a plane and score all other points with respect to this plane. While $N = 307200$ is very large, since GROUSE can work with sparse columns, we estimate the values for only 20% of the entries in each column and use $T = 1000$ column samples per iteration in Algorithm 1. Our method rapidly converges and as can be seen from Fig. 4(c) we achieve excellent segmentation of the depth map into planar regions. We remark that some points at the left boundary of the cupboard are incorrectly classified since the Kinect estimates at a depth discontinuity are unreliable. Interestingly, points on the lower rack of the bookshelf are correctly classified along with the backwall since the shelf is empty. We also remark that the SSC method of [6] cannot handle such a large sized problem as it will need to solve $N = 307200$ individual optimizations. Similarly direct attempts at spectral clustering will not work due to the obvious memory bottleneck of representing an $N \times N$ similarity matrix for large values of N.

6.2. Motion Segmentation : Hopkins 155 Dataset

In this subsection we present our quantitative results for motion segmentation of the well-known Hopkins 155 dataset [14] that consists of 155 video sequences with 2 or 3 independent affine motions. For a video sequence of F frames, individual tracked two-dimensional feature points can be represented by a vector in $\mathbb{R}^{2F}$. Under an affine camera model, all feature tracks of a rigidly moving object lie in an affine subspace of dimension $d = 3$ or linear subspace of $d = 4$. Therefore, our observation matrix of the tracks X is of size $2F \times N$ and we wish to cluster individual columns of X into K subspaces each of dimension 4. For this dataset, we solve the segmentation problem using n-tuples where $n = 8$. For an individual point track $X(:, i)$ where $i \notin \mathcal{I}$ we estimate the projection error as $\mathbf{e}_i = (I - U_{\mathcal{I}} {U_{\mathcal{I}}}^T) X(:, i)$, where $U_{\mathcal{I}}$ is the fitted subspace for the n-tuple of selected points indexed by $\mathcal{I}$.

Method	LSA	SCC	LRR-H	LRSC	SSC	**SGC**
2 motions						
Mean	4.23	2.89	2.13	3.69	1.52	**1.03**
Median	0.56	0.00	0.00	0.29	0.00	**0.00**
3 motions						
Mean	7.02	8.25	4.03	7.69	4.40	**5.53**
Median	1.45	0.24	1.43	3.80	0.56	**0.35**
All						
Mean	4.86	4.10	2.56	4.59	2.18	**2.05**
Median	0.89	0.00	0.00	0.60	0.00	**0.00**

Table 1. Clustering error in percentage for the Hopkins 155 dataset. The last column **SGC** shows the error rates for our Sparse Grassmann Clustering method. All other results have been taken from Table 1 of [6].

For Eqn. 6, we use $\mathcal{P}(i, \mathcal{I}) = exp(-\frac{||\mathbf{e}||}{\sigma})$ which is less sensitive than the Gaussian form when $p = 2$. For each iteration of Algorithm 1, we use $T = 1000$ columns, but for each column we only estimate 10% of the entries of each column. In Table 1 we present the error rates for our method as well as that of other methods in the literature for comparison. Except for the last column presenting the results of our SGC method, all other columns corresponding to different methods are taken from Table 1 of [6]. As can be seen, our SGC method matches or outperforms the state-of-the-art results of the Sparse Subspace Clustering (SSC) method of [6]. While the mean error for 3 motions sequences for our method is higher than that of SSC, it will be noted that the median error is lower. This higher mean error is due to poor classification on 3 sequences where our classification error is somewhat higher. It will also be noted that we outperform the Spectral Curvature Clustering (SCC) method of [3] for the mean error in all cases.

7. Conclusion

In this paper we have presented our Sparse Grassmann Clustering (SGC) method that can solve the higher-order clustering problem by utilising the geometric structure of the Grassmann manifold. As it uses partial observations to incrementally update the clustering representation on the Grassmann manifold, our method is computationally efficient and has a very low memory requirement. Our method is efficient and scalable and can solve large-scale clustering problems such as segmenting Kinect depth maps. The accuracy of our method is also demonstrated on the motion segmentation problems of the Hopkins 155 dataset where we achieve results comparable to the state-of-the-art.

8. Acknowledgments

Preliminary research for this paper was carried out by Venu Madhav Govindu during a visit to Rama Chellappa's group at the University of Maryland, College Park, USA. This visit was partially supported by a MURI Grant N00014-10-1-0934 from the US Office of Naval Research. The authors thank the reviewers and Kaushik Mitra for suggesting the use of the GROUSE algorithm.

References

[1] S. Agarwal, J. Lim, L. Zelnik Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In CVPR, 2005.

[2] L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. In Proceedings of Allerton Conference on Communication, Control and Computing, 2010.

[3] G. Chen and G. Lerman. Spectral curvature clustering (scc). IJCV, 81(3), March 2009.

[4] Y. Chikuse. Statistics on Special Manifolds. Springer, 2003.

[5] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. SIAM Journal on Matrix Analysis and Applications, 20(2), 1998.

[6] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. PAMI, 35(11), 2013.

[7] P. Favaro, R. Vidal, and A. Ravichandran. A closed form solution to robust subspace estimation and clustering. In CVPR, 2011.

[8] V. M. Govindu. A tensor decomposition for geometric grouping and segmentation. In CVPR, 2005.

[9] J. He, L. Balzano, and A. Szlam. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In CVPR, 2012.

[10] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. J. Mach. Learn. Res., Aug 2010.

[11] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. PAMI, 35(1), 2013.

[12] U. Luxburg. A tutorial on spectral clustering. Statistics and Computing, 17(4), December 2007.

[13] A. Shashua, R. Zass, and T. Hazan. Multi-way clustering using super-symmetric non-negative tensor factorization. In ECCV, 2006.

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

[15] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical computations on grassmann and stiefel manifolds for image and video-based recognition. PAMI, 33(11), November 2011.

[16] R. Vidal. Subspace clustering. IEEE Signal Processing Magazine, May 2011.

[17] Y. Weiss. Segmentation using eigenvectors: A unifying view. In ICCV, 1999.

[18] J. Yan and M. Pollefeys. A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In ECCV, 2006.