Subset Selection for Subspace Clustering

M. Clara De Paolis Kaluza
Class Project: EECE 7370 Advanced Computer Vision
Northeastern University, Spring 2017
clara@ccs.neu.edu

Abstract

Many high-dimensional data, such as those in many computer vision applications, actually lie in a union of low-dimensional subspaces. In general, the dimensionality of these subspaces, the number of subspaces in the union, and the segmentation of the data is unknown. Solving these linked problems is addressed by the task of subspace clustering or segmentation. In this work, the subspace clustering problem is cast as a subset selection problem where a small set of representative models are chosen among a large set of candidates. A candidate set of subspaces is generated by fitting local subspaces to points in the data set. The distance between data points and the candidate subspaces is calculated, and a small subset of subspaces are selected by solving the dissimilarity-based sparse subset selection [4]. Experiments on motion segmentation in videos show the ability of the method to segment subspaces successfully in real-world applications.

1. Introduction

In computer vision and other machine learning fields, data are commonly high-dimensional, with a large number of features per data point. For example, an image may be represented by the intensity of its pixels in each color channel or by several low-level features extracted from the image. However, in many applications the information of interest has a much lower-dimensional structure. For instance, data may lie in a single lower-dimensional space and dimensionality reduction can be performed using PCA to discover an underlying low-dimensional structure of the data. In the general case however, data may exist in the union on several subspaces of the original higher-dimensional ambient space (see Figure 1 for an example). In such cases finding these subspaces is complicated by the fact that the subspace to which each data point belongs is unknown, the clustering of points is unknown, the subspaces themselves, including their dimensionality and bases are also unknown, and in general, the number of subspaces may also be unknown.

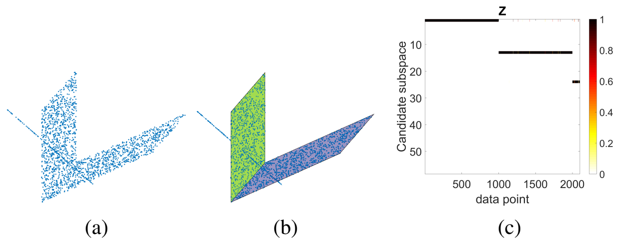


Figure 1: (a) Example of points in a 3 dimensional ambient space that lie in a union of three two- and one-dimensional subspaces of $\mathbb{R}^3$. (b) Segmentation of points according to their subspace. (c) Our method chooses 3 of the candidate subspaces that best represent the data, denoted by the non-zero row of the matrix Z.

Beyond the problem of both the subspace parameters and the data subspace membership being unknown, subspace clustering suffers from several challenges that models must address. First, data may contain noise, outliers, or missing data that make characterizing their subspaces more difficult. Secondly, subspaces may intersect and therefore points near the boundaries of the intersection may be very close in the original space and their subspace assignment may be difficult to distinguish. Furthermore, data may be arbitrarily distributed in the subspace on which they lie rather than being distributed closely around a cluster center or cluster centers may be close in the ambient space while lying on different subspaces. Finally, since the number of subspaces and their dimensionality is unknown, model selection must balance adding more subspaces of fewer dimensions or having fewer subspaces with greater dimensions to more adequately represent the data. Many approaches have been proposed to address some of these problems, and in Section 2 we review the methods most closely related to our work.

Many problems in computer vision may be framed as subspace clustering problem. Images of a three-dimensional object under varying lighting conditions have been shown in [9] to lie a low dimensional subspace, with different objects having images that lie in distinct subspaces. Thus, finding the segmentation of the subspaces provides a clustering of the objects. This property has been used to perform facial recognition since images of the same subject under varying lighting conditions lie in a low-dimensional subspace of the high dimensional ambient space of the image features. Motion trajectories of rigid objects have also been found to lie in lower-dimensional subspaces of the ambient space of image sequences. Objects with different motions, i.e. different objects, have motion trajectories that lie on different subspaces [3], thus segmenting the subspaces identifies the distinct objects in a video that move relative to each other.

Contribution In this paper, we consider the subspace clustering problem as a model selection problem that we solve as a dissimilarity-based sparse subset selection among a collection of candidate models (subspaces). We We apply our method to the task of motion segmentation in the experiments section and show that our method is competitive with previous work.

Organization The remainder of this paper is as follows. Section 2 provides a review of subspace clustering methods, with the works most closely related to the method proposed here highlighted. This section also provides a background on subset selection in the context of model selection. Section 3 provides the details of the proposed work. In Section 4, we provide an analysis of the method through experiments on synthetic data as well as experiments on the motion segmentation benchmark dataset Hopkins 155[15]. We provide conclusions and future directions in Section 5.

2. Background

2.1. Subspace Clustering

Many approaches to subspace clustering have been proposed Algebraic methods for subspace clustering focus on finding a low-rank factorizing of the data matrix and use its structure to segment the data, e.g.[3], or fitting the subspaces with a polynomial, i.e.[17]. Iterative methods solve the problem by alternatively assigning points to subspaces and updating the subspaces to fit the points assigned to them, analogous to k-means for clustering e.g.[1]. Statistical methods fit the data to subspaces by making assumptions on the distribution of the data in each subspace, e.g.[14], [13]. Finally, spectral clustering methods seek to form an affinity matrix on which to define a graph on the data, where more strongly connected components will belong to the same subspace and more weakly connected components will belong to different subspaces. For these methods, the clustering is found by spectral clustering methods but the subspaces clustering methods differ on how the affinity matrix is defined, e.g.[6], [11].

As a means of comparison to our work, we highlight a few methods in the above categories to explain in further detail. The Random Sample Consensus (RANSAC) algorithm [8] is a statistical method that finds one subspace at a time by fitting the data to a single subspace and identifying points as outliers if the residual (normal distance) to the subspace is above some threshold. The remaining points are considered inliers and removed from consideration in finding the remaining subspaces. The process continues until some minimum amount of inliers are identified or a certain number of subspaces are fit to the data. In this method, the subspaces are found by sampling d random points and fitting a $(d-1)$-dimensional subspace to those points. Once all the sets of inliers are found, a optimal basis for the subspace on which those points lie can be found using PCA. This step improves the subspaces since they are now required to fit only the inliers found in the first step and not the outliers.

The Local Subspace Affinity (LSA) [18] method is similar to RANSAC in that it also fits subspaces to a subset of points as a first step. However, in contrast to RANSAC, LSA is based on the assumption that data points that are close in the ambient space should lie on the same subspace. Thus, local subspaces are fit to each point and its k nearest neighbors as measured by their distance in the ambient space. This process produces one subspace $\hat{\mathcal{S}}_i$ associated with each point i. The goal of LSA is to cluster point which have lose-by subspaces. This aim is achieved by calculating the principal angles $\{\theta_{ij}^1, \theta_{ij}^2, \cdots \theta_{ij}^{\min(d_i,d_j)}\}$ between the bases of the subspaces $\hat{\mathcal{S}}_i$ and $\hat{\mathcal{S}}_j$ for two points i and j and constructing an affinity matrix A with (i,j)-th entry given by

$$A_{ij} = \exp\left(-\sum_{m=1}^{\min(d_i,d_j)} \sin^2(\theta_{ij}^m)\right) \qquad (1)$$

Here, d_i is the dimension of the i-th subspace $\hat{\mathcal{S}}_i$ and θ_{ij}^m is the m-th principal angle between $\hat{\mathcal{S}}_i$ and $\hat{\mathcal{S}}_j$. This affinity matrix is used for spectral clustering of the points. If the principal angles between the subspaces associated with points i and j are small, the subspaces are close to each other and A_{ij} is large (close to 1). Therefore, points i and j will be likely clustered together when spectral clustering is applied to the affinity matrix. In contrast, if the angles are large, and the subspaces are far apart, A_{ij} will be small and i and j are unlikely to appear in the same cluster.

The most closely related method to ours is Facility Location for Subspace Segmentation (FLoSS)[10] algorithm. FLoSS constructs a candidate set of subspaces by fitting subspaces of dimension $d-1$ to random sets of d points, as in RANSAC. As in RANSAC, the normal distances from each point and each subspace is calculated. The final step differs from RANSAC. Instead of using these candidate subspaces to identify the segmentation, a small subset of the candidates are chosen by solving the facility location problem approximately. Abstractly, the facility location is an optimization problem where there is a set of "customer" and a set of "facilities." There is a distance between customers and facilities and a cost associated with opening/operating a facility. The optimization problem seeks to find an assignment of customers to facilities such that the distance between them minimized, the cost of facilities is minimized, and each customer is assigned to exactly one facility. In the context of subspace clustering, the normal distance between points and subspaces serves as the distance measure and [10] assigns the cost of choosing a subspace to represent some data points as the sum of all the pairwise distances between points "assigned" to that subspaces. This cost captures a similar assumption as is made by LSA that points that are close in the ambient space should be represented by the same subspace. In [10] the NP-hard facility location problem is solved approximately using message passing.

2.2. Subset Selection

Similarly to [10], we consider the problem of subspace clustering one of choosing a small set of models (subspaces) to model the data. However, we solve the optimization in a different way than does [10]. Since our method is based on the Dissimilarity-based Sparse Subset Selection (DS3) [5], we review it here. DS3 is a subset selection method that seeks to find a small number of representative elements to encode all the data points in a dataset as defined by a pairwise dissimilarity measure between elements of each set. The set of candidate representatives is the *source set*, $\mathbb{X}$, and the set of data points to encode is the *target set*, $\mathbb{Y}$. In the case that the source set and the target set are identical, the representatives are elements of the dataset itself. However, the source and target sets are not restricted to being identical or even to be elements of the same type.

Let $\mathbb{X}$ be the source set of M elements $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\} \in \mathbb{X}$ and $\mathbb{Y}$ the target set of N elements $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$. Assume there is a measure of dissimilarity d_{ij} between each element $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$. A matrix $\boldsymbol{D} \in \mathbb{R}^{M\times N}$ can be constructed where the i, j-th element is given by the dissimilarity d_{ij}. Given this matrix, DS3 seeks to find a small number of elements of $\mathbb{X}$ to efficiently represent all the elements of $\mathbb{Y}$. Concretely, the method seeks to optimize a set of variables z_{ij} to indicate whether or not $\boldsymbol{x}_i$ is a representative for $\boldsymbol{y}_j$. If it is, $z_{ij} = 1$ and otherwise $z_{ij} = 0$. These variables can be arranged in a matrix $\boldsymbol{Z} \in \mathbb{R}^{M\times N}$ with i, j-th element z_{ij}. The optimization to find the values of $\boldsymbol{Z}$ is given by

$$\begin{aligned} \min_{\{z_{ij}\}} \ & \lambda \sum_{i=1}^{M} \mathbf{I}(\|\boldsymbol{z}_i\|_p) + \sum_{j=1}^{N}\sum_{i=1}^{M} d_{ij} z_{ij} \\ \text{s.t} \ & \sum_{i=1}^{M} z_{ij} = 1, \ \forall j; \quad z_{ij} \in \{0,1\}, \ \forall i, j. \end{aligned} \tag{2}$$

Here, $\|\boldsymbol{z}_i\|_p$ is the ℓ_p-norm of the i-th row of $\boldsymbol{Z}$, $\mathbf{I}(\cdot)$ is the indication function which equals one when its argument is nonzero and is zero otherwise. The second term in objective captures the sum of the distances between each point in $\mathbb{Y}$ and its chosen representative in $\mathbb{X}$. The ℓ_p-norm of $\boldsymbol{z}_i$ is nonzero only when there is at least one nonzero element in that row, indicating that the i-th element was chosen as a representative for at least one element of $\mathbb{Y}$. Therefore, the first term is a count of the nonzero rows of $\boldsymbol{Z}$, with a parameter λ serving as a trade-off between the two terms in the objective. The constraints encode the goal of each element of $\mathbb{Y}$ being represented by exactly one element of $\mathbb{X}$ and the assignment indicated by z_{ij} should be a hard assignment, either zero or one. Since this problem is NP-hard, DS3 proposes a convex relaxation where the indicator function is replaced by the sum of row norms and the variables z_{ij} are no longer binary

$$\begin{aligned} \min_{\{z_{ij}\}} \ & \lambda \sum_{i=1}^{M} \|\boldsymbol{z}_i\|_p + \sum_{j=1}^{N}\sum_{i=1}^{M} d_{ij} z_{ij} \\ \text{s.t} \ & \sum_{i=1}^{M} z_{ij} = 1, \ \forall j; \quad z_{ij} \geq 0, \ \forall i, j. \end{aligned} \tag{3}$$

This formulation is convex for $p \geq 1$ but the ℓ_1 norm promotes sparsity in the rows. This behavior is not desired since we want each chosen representative to represent as many points in $\mathbb{Y}$ as it can encode well, according to $\boldsymbol{D}$. Therefore, p is chosen as either 2 or ∞, with ∞ leading to solutions closer to hard assignment, where z_{ij} is close to binary. The optimization can be equivalently represented in matrix form as

$$\begin{aligned} \min_{\boldsymbol{Z}} \ & \lambda \|\boldsymbol{Z}\|_{1,p} + \operatorname{tr}(\boldsymbol{D}^T \boldsymbol{Z}) \\ \text{s.t} \ & \mathbf{1}^T \boldsymbol{Z} = \mathbf{1}, \quad \boldsymbol{Z} \geq \mathbf{0}, \end{aligned} \tag{4}$$

where $\|\boldsymbol{Z}\|_{1,p} = \sum_{i=1}^{M} \|\boldsymbol{z}_i\|_p$ and $\operatorname{tr}(\cdot)$ is the trace operator. Again, λ controls the relative importance of the two objectives: encoding well the elements of $\mathbb{Y}$ and having few representatives chosen from $\mathbb{X}$. When λ is large, the first term dominates the minimization and fewer representatives are chosen. When λ is small, the encoding cost in the second term dominated the objective function and more representatives are chosen so that points in $\mathbb{Y}$ are better represented. It is shown in [5] that when λ is set above some threshold

$\lambda_{\max}$, only one representative from $\mathbb{X}$ is chosen. This value depends only on the characteristics of the dissimilarity matrix.

The optimization in (4) is solved efficiently using the Alternating Direction Method of Multipliers (ADMM) framework [2]. Briefly, this method splits the optimization by introducing a new auxiliary variable $\boldsymbol{C}$ and the matrices $\boldsymbol{Z}$ and $\boldsymbol{C}$ are optimized in (5) iteratively until they converge.

$$\min_{\boldsymbol{Z},\boldsymbol{C}} \lambda\|\boldsymbol{Z}\|_{1,p} + \operatorname{tr}(\boldsymbol{D}^T\boldsymbol{C}) + \frac{\mu}{2}\|\boldsymbol{Z}-\boldsymbol{C}\|_F^2 \quad (5)$$
$$\text{s.t} \quad \boldsymbol{1}^T\boldsymbol{C} = \boldsymbol{1}, \quad \boldsymbol{C} \geq \boldsymbol{0},\ \boldsymbol{Z} = \boldsymbol{C}$$

3. Subset Selection for Subspace Clustering

To solve the subspace clustering problem using DS3, we define the problem as follows. Consider N data points $\{\boldsymbol{y}_i, \ldots, \boldsymbol{y}_N$ in a K-dimensional ambient space arranged in a matrix $Y \in \mathbb{R}^{N\times K}$. These data form the target set $\mathbb{Y}$. Since we are concerned with subspace clustering, we assume these data come from a union of n subpaces of $\mathbb{R}^K$ $\{\mathcal{S}_i\}_{i=1}^n$ each with dimension $\dim_i \ll K$. We form a set of m candidate subspaces $\{\mathcal{X}_i\}_{i=1}^m$ to define a source set $\mathbb{X}$. As in RANSAC and FLoSS, we define the distance d_{ij} from data point $\boldsymbol{y}_j \in \mathbb{Y}$ to a subspace $\mathcal{X}_i$ as the normal distance from the projection of $\boldsymbol{y}_j$ on $\mathcal{X}_i$. We arrange these distances to form the dissimilarity matrix $\boldsymbol{D} \in \mathbb{R}^{m\times N}$ with the i,j-th element of $\boldsymbol{D}$ equal to d_{ij}. By formulating the problem in this way, DS3 can be applied to find a small number of subspaces among the candidate set $\{\mathcal{X}_i\}_{i=1}^m$ that will represent the all the data points in $\mathbb{Y}$. Our proposed method is outlined in Algorithm 1 and detailed as follows.

Algorithm 1 Dissimilarity-based sparse subset selection for Subspace Clustering

Input: Matrix $Y \in \mathbb{R}^{N\times K}$ of N data points with ambient space dimension K, neighborhood size k.

```
 1: Initialize empty candidate set of subspaces $\mathcal{S} = \{\}$
 2: Initialize empty dissimilarity matrix $D$
 3: for $\boldsymbol{y}_i \in Y$ do
 4:     find $k$ nearest neighbors $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\}$ to $\boldsymbol{y}_i$
 5:     fit subspace $\mathcal{S}_i$ to $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\} \cup \boldsymbol{y}_i$
 6:     if $\mathcal{S}_i$ is sufficiently far from every subspace $\mathcal{S}_j \in \mathcal{S}$ then
 7:         $\mathcal{S} = \mathcal{S} \cup \mathcal{S}_i$
 8:     end if
 9: end for
10: for $\mathcal{S}_i \in \mathcal{S}$ do
11:     $D(i,j) \leftarrow$ distance from $\boldsymbol{y}_j$ to $\mathcal{S}_i$ $\forall \boldsymbol{y}_j \in Y$
12: end for
13: Solve the DS3[5] problem to find representatives indicated by $Z$
14: Assign each data point the subspace with the largest value in the corresponding row of $Z$
```

Output: clustering Z

3.1. Forming the Candidate Set

To form a suitable candidate set of subspaces, we fit local subspaces to the points in $\mathbb{Y}$. To find these local subspaces, we fit a subspace to each point $\boldsymbol{y}_i$ and its k nearest neighbors as defined by the Euclidean distance in the K-dimensional ambient space. This method reflects the assumption that points that are nearby in the ambient space may belong to the same subspace. However, unlike in the LSA method, we do not restrict a point to be modeled by this local subspace. This process only generates a large set of candidate subspaces from which a small subset will be chosen to represent the data. Note that this candidate set differs from FLoSS and RANSAC in that the subspaces are *local* to each point, not a random sample among all the points in the dataset.

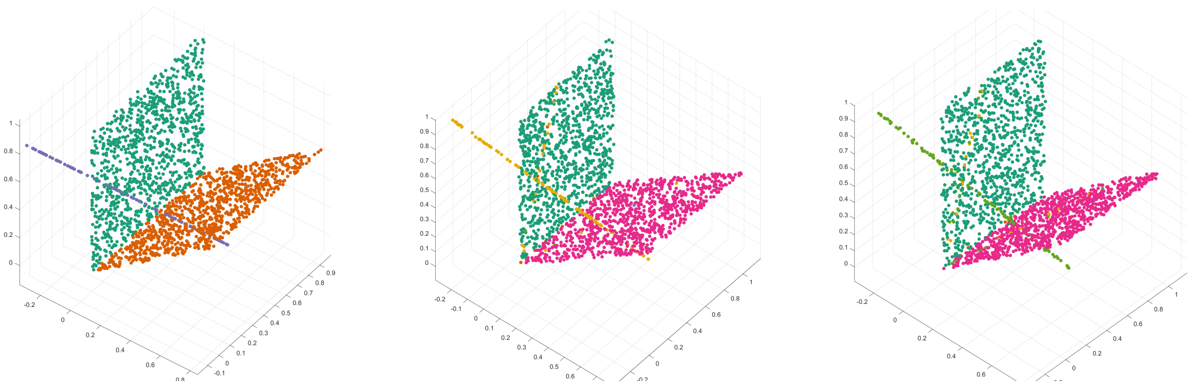
Since this proposed method of generating the subspace candidate set produces one subspace for each point in the dataset, the optimization required for the subset selection step would be computationally costly as the number of points increases. To promote better scaling as the data grows, we also propose an optional step of pruning the candidate set. When a local subspace is fit to a point, we add it to the candidate set only if it is sufficiently different than the subspaces already in the set. We measure the similarity of subspaces by the principal angles between their bases. If a subspace is found such that the sum of the principal angles are lower than some threshold, the new candidate subspace is not added to the set and we move on to fitting the next local subspace.

3.2. Segmentation

Once the candidate set is formed and the distance from each point to each candidate subspace is calculated, DS3 can be applied to the dissimilarity matrix $\boldsymbol{D}$ to find a matrix $\boldsymbol{Z}$. The nonzero rows of $\boldsymbol{Z}$ indicate the subspaces chosen as potential representatives. Because the optimization in (4) is a convex relaxation of (2), the variables z_{ij} are no necessarily binary. Thus, a column $\boldsymbol{Z}_{:,j}$, corresponding to the point $\boldsymbol{y}_j$, may have several nonzero entries. This means that several subspaces may be suitable to be representatives of $\boldsymbol{y}_j$. To solve this, we assign point $\boldsymbol{y}_j$ to subspace $\mathcal{X}_i$ if $z_{ij} > z_{kj} \forall k \neq i$. That is, subspace $\mathcal{X}_i$ is chosen to represent point $\boldsymbol{y}_j$ if the maximum element of the column $\boldsymbol{Z}_{:,j}$ is at position i.

4. Experiments

To motivate the strength of our method, we conduct experiments on some synthetic data to illustrate some compo-



(a) $\sigma = 0$ (b) $\sigma = 0.0025$ (c) $\sigma = 0.005$

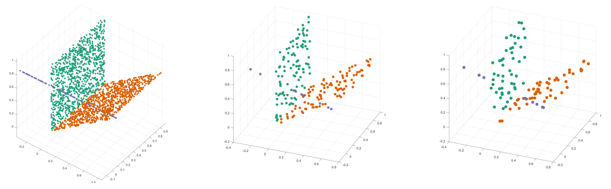
Figure 2: Synthetic data experiments with varying amounts of noise. The amount of Gaussian noise is listed under each image

nents of our method and we also apply out subspace clustering method to a real-world dataset for motion segmentation.

4.1. Synthetic Experiments

To gain some intuition about our method, we perform some experiments on a synthetic dataset constructed in three-dimensional ambient space as a union of two two-dimensional subspaces and a one-dimensional subspace, shown in Figure 2. As the one-dimensional subspace does not contain the origin, it is an affine subspace rather than linear. Figure 2 shows the segmentation results of our method on this dataset with varying amounts of Gaussian noise added to the data. The variance of this noise is listed under each figure. For this experiment we have fixed the regularization parameter λ to be $(0.01)\lambda_{\max}$. As the noise increases, the performance degrades slightly, and more representative subspaces are chosen for the same setting of λ. For all case, we set $k = 3$ for fitting the local subspaces to the k-nearest neighbors.

In a second experiment, we show the results for varying sizes of datasets. Figure 3a shows the segmentation results for a dataset composed of 1000 points in each two-dimensional subspace and 100 points in the one-dimensional subspace. Figure 3b shows a dataset with 100 points per two-dimensional subspace and 10 points in the one-dimensional subspace. Finally, Figure 3c shows a dataset of 50 points per two-dimensional subspace and 10 points in the one-dimensional subspace. In each of these case, we show the results for a noise-free dataset. Again, we fix $k = 3$ to fit the local subspace. The results show that even when the subspaces are very sparse and the points are very far in the ambient space, our method is able to successfully segment the subspaces. This is especially pronounced in the one-dimensional subspace which exhibits a large gap with no points in the middle. These results motivate the strength of our method to segment subspaces even when they are sparse and intersect each other. Namely, the one-dimensional subspace has points that are close to the other subspaces in the ambient space, but our method is able to accurately identify the underlying subspace of those points.



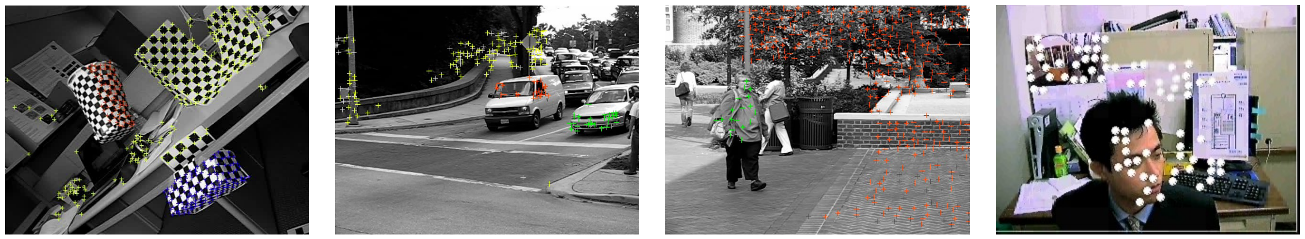
(a) 1000, 100 (b) 100, 10 (c) 50, 10

Figure 3: Synthetic data experiments with varying number of points per subspace. Under each figure, the number of points in the 2d subspaces is listed first, then the number of points in the 1d subspaces.

4.2. Motion Segmentation

The task of motion segmentation addresses separating moving objects in video sequences, identifying their location in space across the frames of the video. The first step of many motion segmentation methods is to extract motion trajectories of points in the video sequence. Under the affine projection model, these motion trajectories of n rigid bodies lie in union of affine subspaces of $\mathbb{R}^{2F}$, where F is the number of frames in the video sequence. Each subspace has a dimension of at most three. Subspace clustering of these trajectories provides a segmentation of the motions in the video.

The Hopkins 155[15] dataset provides 155 videos and extracted motion trajectories for videos of either two or



(a) Checkerboard (3): 1R2TCRT (b) Traffic (3): cars9 (c) Other (2): people2 (d) Other (2): kanatani3

Figure 4: The first frame from a few example sequences of the Hopkins 155 motion segmentation dataset. Tracked points are denoted by a cross or circle on the images. The category, number of motions, and sequence name are listed under each image.

three motions. A few different types of sequences are provides. The Checkerboard sequences are videos in an indoor setting with objects covered in a checkerboard patter move and rotate. The camera also undergoes rotation and translation. The Traffic sequences are videos of outdoor traffic scenes with trajectories tracked on vehicles with non-stationary camera. Lastly the Other sequences contain articulated motions just as those of people moving or walking and cranes working. The dataset contains noise, but outliers have been manually removed. Examples of these sequences for each type are shown in Figure 4.

As in previous work, we pre-process the data by projecting it into a lower-dimensional ambient space using PCA. As in [7], we chose to project the data into $\mathbb{R}^{2F}$. This step reduces the dimensionality of the ambient space to match the assumption of the affine projection camera model. As in the synthetic experiments, we fix k to be equal to the ambient space. This choice is motivated by the findings in [18] that the choice of k is not crucial as long as it is at least as large as the subspace dimension. Better results might have been found by setting k to 4 to match the assumption that the *affine* subspaces are at most three-dimensional, but we leave this to future work. We keep the value of λ constant for all sequences, setting it large enough to not merge clusters that belong to different subspaces. Thus do not explicitly restrict the number of subspaces we find, unlike in other work. An example of our segmentation and the associated $\boldsymbol{Z}$ matrix is shown in Figure 5.

As we do not restrict the number of subspaces chosen by our method, we do not report the classification error. Instead, we evaluate the cluster purity [12], defined as follows. Let N be the number of data points, in this case the number of motion trajectories. Let C_i represent the points assigned to the i-th chosen subspace, $N_i = |C_i|$ be the number of points in C_i, and N_{ij} the number of points in cluster C_i that belong to the class (motion) j. For example, if 100 tracked points were assigned to a subspace from the candididate space and 85 of those 100 points belonged to motion 1 and the remaining 15 points belonged to motion 2, then $N_i = 100$, $N_{i1} = 85$, and $N_{i2} = 15$. Define $p_{ij} := \frac{N_{ij}}{N_i}$. This quantity represents the empirical distribution of classes (motion labels) for cluster C_i. For the example above, $p_{i1} = 85/100 = 0.85$ and $p_{i1} = 15/100 = 0.15$. Define $p_i := \max_j p_{ij}$ as the fraction of points in cluster C_i that belong to the majority class in C_i. Returning to our example, this value would be $p_i = \max\{p_{i1}, p_{i2}\} = \max\{0.85, 0.15\} = 0.85$. Finally, the purity is a measure of the average of p_i for all clusters C_i:

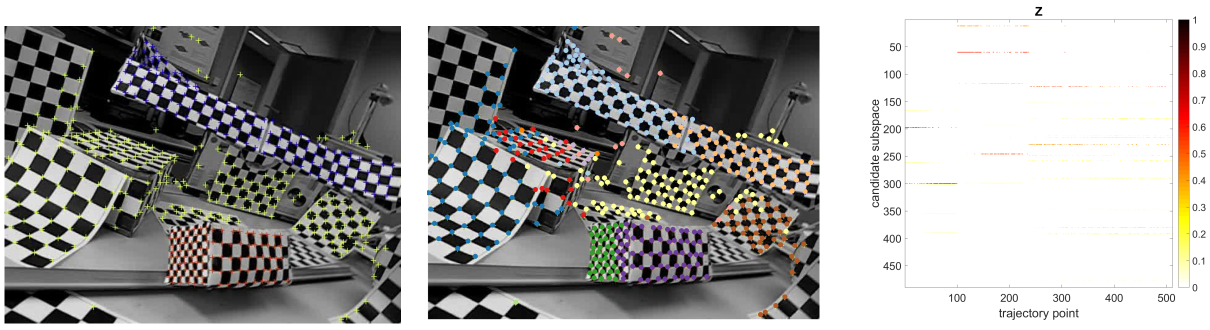
$$\text{purity} := \sum_i \frac{N_i}{N} p_i \tag{6}$$

This measure ranges from 0 (worst) to 1, indicating that all clusters contain only a single class. To form an error measure comparable to those reported in earlier works, we modify the purity measure as an error by subtracting the measure from 1 and multiplying by 100 to get a percent.

$$\text{purity error rate} := 100 \times \left(1 - \sum_i \frac{N_i}{N} p_i\right) \tag{7}$$

We report the purity error rate in Table 1b for two motion sequences and Table 2b for three motion sequences. The average number of clusters found for 2 motions is 5.30 clusters and for three motions 8.14 clusters. The low purity error rates we report indicate that each chosen subspace represents mostly points from a single class (motion). This result indicated that an additional step to merge clusters may be used to find good segmentations of the motion data.

We also list the results for the methods we described in Section 2 as well as top-performing method of Sparse Subspace Clustering (SSC) [7]. In the case of SSC, instead of using PCA to project the data to $\mathbb{R}^{2F}$, better results are achieved by random Normal projections. This pre-processing step may serve to improve the results of our method as well, but leave this to future work. The results of previous methods are listed in Table 1a for two-motion



(a) (b) (c)

Figure 5: The three-motion checkerboard sequence ‘2RT3RCR’ in the Hopkins 155 dataset. The ground truth segmentation for the first frame is shown in 5a. The segmentation from our method on the first frame is shown in 5b (best viewed in color) with the corresponding Z matrix shown in 5c. The columns of Z represent the individual motion trajectory points and the rows correspond to the candidate subspaces. The rows are shown ordered by their true clustering, with the first 100 columns corresponding to the first motion, columns 101 through 237 to the second motion, and the rest corresponding to the third motion. Notice the structure of the nonzero entries of Z which reflects the underlying segmentation of the motion trajectories. Our method segments this sequence into 12 subspaces and the purity error rate (7) for this sequence is 0.587%. This reflects the fact that although each motion is segmented into several clusters by our method, there is almost no overlap in the clusters assigned to different motions.

	Seq.	Error	RANSAC[8]	LSA[18]	FLoSS[10]	SSC-N[6]
Checker.	78	Mean	6.52	2.57	7.70	1.12
		Med.	1.75	0.27	1.23	0.00
Traffic	31	Mean	2.55	5.43	0.14	0.02
		Med.	0.21	1.48	0.00	0.00
Other	11	Mean	7.25	4.10	4.69	0.62
		Med.	2.64	1.22	1.30	0.00

(a) Classification error (%) of previous work

	Seq.	Purity Error	Ours
Checker.	78	Mean	2.48
		Med.	1.40
Traffic	31	Mean	2.15
		Med.	1.39
Other	11	Mean	2.17
		Med.	1.07

(b) The purity error rate (7) for this work’s clustering

Table 1: Experimental results on the Hopkins 155 dataset for two-motion sequences

	Seq.	Error	RANSAC[8]	LSA[18]	FLoSS[10]	SSC-N[6]
Checker.	26	Mean	25.7	5.80	16.45	2.97
		Med.	26.01	1.77	16.79	0.27
Traffic	7	Mean	12.83	25.07	0.29	0.58
		Med.	11.45	23.79	0.00	0.00
Other	2	Mean	21.38	7.25	8.51	1.42
		Med.	21.38	7.25	8.51	0.00

(a) Classification error (%) of previous work

	Seq.	Purity Error	Ours
Checker.	26	Mean	2.37
		Med.	1.94
Traffic	7	Mean	4.28
		Med.	4.25
Other	2	Mean	4.79
		Med.	4.79

(b) The purity error rate (7) for this work’s clustering

Table 2: Experimental results on the Hopkins 155 dataset for three-motion sequences

sequences and Table 2a for three-motion sequences. Compared to other work, our method offers fairly consistent results for all features and performs consistently for two- and three-motion sequences. In contrast, RANSAC, LSA, and FLoSS exhibit dramatic drops in performance between the two- and three-motion sequences. Likewise, they perform much better on some types of sequences than others. This suggests that our method is more adaptable to different number of subspaces and to the different configurations of the subspaces realized by different types of sequences in this dataset.

5. Conclusion

We present a method that leverages the dissimilarity-based sparse subset selection framework to efficiently perform subspace clustering. To do so, we propose generating a large set of local subspaces to the the data and use DS3 to select a small number of these candidates to best represent all the data points. We demonstrate the usefulness of this method on synthetic and real-world datasets. Further optimizations are possible in the choice of candidate subspaces and the method for pruning this set. We leave this to future work. Furthermore, the DS3 construction offers a variation to handle outliers, which we do not address here but leave this extension of our method to future work.

References

[1] P. K. Agarwal and N. H. Mustafa. k-Means Projective Clustering. *ACM SIGMOD-SIGACT-SIGART Symp. Princ. database Syst.*, pages 155–165, 2004.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*, volume 25. 2010.

[3] J. P. Costeira and T. Kanade. A Multibody Factorization Method for Independently Moving Objects. *Int. J. Comput. Vis.*, 29(3):159–179, 1998.

[4] E. Elhamifar, G. Sapiro, and S. S. Sastry. Dissimilarity-based Sparse Subset Selection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8828(c):1–19, 2016.

[5] E. Elhamifar, G. Sapiro, and S. S. Sastry. Dissimilarity-based Sparse Subset Selection. *PAMI*, pages 1–18, 2016.

[6] E. Elhamifar and R. Vidal. Sparse Subspace Clustering :. *CVPR*, pages 130–138, 2009.

[7] E. Elhamifar and R. Vidal. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2765–2781, 2013.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 26, 1981.

[9] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. *2003 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, 2003. Proceedings.*, 1, 2003.

[10] N. Lazic, I. Givoni, B. Frey, and P. Aarabi. FLoSS: Facility location for subspace segmentation. *Proc. IEEE Int. Conf. Comput. Vis.*, pages 825–832, 2009.

[11] G. Liu and Z. Lin. Robust Subspace Segmentation by Low-Rank Representation. *Matrix*, 2010.

[12] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, Cambridge, MA, 2012.

[13] M. Tipping and C. Bishop. Probabilistic Principal Component Analysis. *J. R. Stat. Soc.*, 61(3):611–622, 1999.

[14] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Comput.*, 11(2):443–482, 1999.

[15] R. Tron and R. Vidal. A Benchmark for the comparison of 3D motion segmentation algorithms. *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–8, 2007.

[16] R. Vidal. Subspace Clustering. *IEEE Signal Process. Mag.*, 28(2):52–68, 2011.

[17] R. Vidal, Y. Ma, and S. Sastry. Generalized Principal Component Analysis. *IEEE Trans. Pattern Analyisis Mach. Intell.*, 27(12):1945–1959, 2006.

[18] J. Yan and M. Pollefeys. A General Framework for Motion Segmentation : Degenerate and Non-degenerate. *Comput. Vis. ECCV 2006*, 3954(Chapter 8):94–106, 2006.