Reduction and Null Space Algorithms for the Subspace Clustering Problem

Akram Aldroubi and Ali Sekmen

Abstract

This paper presents two algorithms for clustering high-dimensional data points that are drawn from a union of lower dimensional subspaces. The first algorithm is based on binary reduced row echelon form of a data matrix. It can solve the subspace segmentation problem perfectly for noise free data, however, it is not reliable for noisy cases. The second algorithm is based on Null Space representation of data. It is devised for the cases when the subspace dimensions are equal. Such cases occur in applications such as motion segmentation and face recognition. This algorithm is reliable in the presence of noise, and applied to the Hopkins 155 Dataset it generates the best results to date for motion segmentation. The recognition rates for two and three motion video sequences are 99.15% and 98.85%, respectively.

Index Terms

Subspace segmentation, motion segmentation, data clustering.

I. INTRODUCTION

Subspace data clustering deals with finding a nonlinear model of the form $\mathcal{U} = \bigcup_{i \in I} S_i$ where $\{S_i\}_{i \in I}$ is a set of subspaces that fits a set of data $W = \{w_1, \ldots, w_N\} \in \mathbb{R}^d$. The model can then be used to classify the data $W$ into classes called clusters.

In many engineering and mathematics applications, data lives in a union of low dimensional subspaces [1], [2], [3], [4]. For instance, consider a moving affine camera that captures $F$ frames of a scene that contains multiple moving objects. If $N$ feature points are detected in each frame and tracked across the frames, the vectors corresponding to paths of the feature points (i.e., these vectors are sometimes called data points) coming from the same object lie in an at most 4 dimensional subspace of $\mathbb{R}^{2F}$. Therefore, the number of subspaces corresponds to the number of moving objects (the stationary background is also considered as a moving object due to the camera motion). Another clustering problem that can be modeled as union of subspaces is recognition of faces. The set of all two dimensional images of a given face $i$, obtained under different illuminations and facial positions, can be modeled as vector belonging to a low dimensional subspace $S_i$ living in a higher dimensional space $\mathbb{R}^d$ [5], [4]. A set of such images from different faces is then a union $\mathcal{U} = \bigcup_{i \in I} S_i$. Similar nonlinear models arise in sampling theory where $\mathbb{R}^d$ is replaced by an infinite dimensional Hilbert space $\mathcal{H}$, e.g., $L^2(\mathbb{R}^d)$ [6], [1], [7].

In many data clustering problems, although the ambient space is high dimensional, the data typically comes from low dimensional subspaces whose sum has a dimension which is often still much smaller than the ambient space. Therefore, high dimensional data is first projected into a low dimensional space and then clusters are generated by identifying further subspaces within the low dimensional space. Measurements are prone to noise and therefore some (if not all) of the data points may be corrupted by noise. In addition, some of the data dimensions may not be available. For instance, the feature points on an object in a video sequence may not be detected across all frames as the object may be occluded from time to time, giving rise to vectors that have some components whose values are unknown. Also, some data may include outliers. For instance, some feature points on the object might be incorrectly located across some frames.

The research of A. Aldroubi is supported in part by NSF Grant DMS-0807464.
A. Subspace Segmentation Problem

The goal of subspace clustering is to identify all of the subspaces that a set of data \( W = \{w_1, ..., w_N\} \in \mathbb{R}^d \) is drawn from and assigning each data point \( w_i \) to the subspace it belongs to. The number of subspaces, their dimensions, and a basis for each subspace are to be determined even in the presence of noise, missing data, and outliers. The subspace clustering or segmentation problem can be stated as follows:

Let \( U = \bigcup_{i=1}^{M} S_i \) where \( \{S_i \subset \mathcal{H}\}_{i=1}^{M} \) is a set of subspaces of a Hilbert space \( \mathcal{H} \). Let \( W = \{w_j \in \mathcal{H}\}_{j=1}^{N} \) be a set of data points drawn from \( U \). Then,

1) determine the number of subspaces \( M \),
2) determine the set of dimensions \( \{d_i\}_{i=1}^{M} \),
3) find an orthonormal basis for each subspace \( S_i \),
4) collect the data points belonging to the same subspace into the same cluster.

In some subspace clustering problems, the number \( M \) of subspaces or the dimensions of the subspaces \( d_i \) are known. A number of approaches have been devised to solve this problem. They can be summarized as follows:

1) Sparsity Methods: In general compressive sampling framework [8], [9], a \( k \)-sparse signal \( x \in \mathbb{R}^n \) (at most \( k \) nonzero entries) can be reconstructed by convex minimizing

\[
\min ||\tilde{x}||_1 \quad \text{subject to} \quad y = A\tilde{x}
\]  

(1.1)

where \( y \in \mathbb{R}^m \) (with \( m < n \)), and \( A \) a measurement matrix satisfying the so called restricted isometry property (RIP) [10].

Using ideas similar to the ones in compressed sensing, Eldar [11] recently considered the recovery of signals that lie in a structured union of subspaces instead of a single subspace. Specifically, a signal \( x \) is assumed to lie in a union of \( k \) disjoint subspaces with known bases. Although \( x \) is in one of the subspaces, it is not known which one priori. She then shows that the problem of reconstructing \( x \) can be cast as a sparse recovery problem, in which a sparse vector with particular sparsity pattern is recovered based on minimizing an \( l_2/l_1 \) norm from given measurements.

Elhamifar et al extended Eldar’s work and developed an algorithm for linear and affine subspace clustering using sparse representation of vectors [12], [13]. They assume that the data points are drawn from a union of independent [12] or disjoint [13] linear or affine subspaces. They further assume that the collection of data points are self-expressive, i.e., any data point \( y \in V \) in the collection, where \( V \) is a \( d \) dimensional subspace, can be expressed as a linear combination of any other \( d - 1 \) points that are in the collection and in \( V \). However, we did not see an argument that guarantees that the measurement matrix used in [12] satisfies RIP.

2) Algebraic Methods: Generalized Principle Component Analysis (GPCA) is the main algebraic approach for subspace clustering [4], [14], [15]. It models \( U = \bigcup_{i \in I} S_i \) with a set of polynomials whose derivatives at a point are used to determine a set of basis vectors for the subspace passing through that point. The basis vectors are then used for segmenting the data. GPCA can distinguish subspaces of different dimensions. Since it is algebraic, it is computationally inexpensive, however, its complexity increases exponentially as the number of subspaces and their dimensions increase. It is also very sensitive to noise and outliers.

3) Iterative and Statistical Methods: Iterative methods such as nonlinear least squares [6], [3] and K-subspaces [5] start with an initial estimation of subspaces (or estimation of the bases of the subspaces). Then, a cost function reflecting the “distance” of a point to a each subspace is computed and the point is assigned to its closest subspace. After that, each cluster of data is used to reestimate each subspace. The procedure is repeated until the segmentation of data points does not change.

The statistical methods such as Multi Stage Learning (MLS) [2], [16] are typically based on Expectation Maximization (EM) [17]. The union of subspaces is modeled by a mixture of probability distributions. For example, each subspace is modeled by a Gaussian distribution. The model parameters are then estimated
using Maximum Likelihood Estimation. This is done by using a two-step process that optimizes the log-likelihood of the model which depends on some hidden (latent) variables. In E-Step (Expectation), the expectation of the log-likelihood is computed using the current estimate of the latent variables. In M-Step (Maximization), the values of the latent variables are updated by maximizing the expectation of the log-likelihood.

The success of the iterative and statistical methods highly depends on initialization of model parameters or segmentation. They generally assume that the number of subspaces as well as their dimensions are known. Random Sample Consensus (RANSAC) [18], which has been applied to numerous computer vision problems, is successful in dealing with noise and outliers, however, it assumes that the dimension of each subspace is known and each subspace has the same dimension.

4) Other Methods: Some of the latest subspace clustering algorithms (such as [12], [13], [19]) aim at defining an appropriate similarity matrix between data points which then can be used for further processing using the spectral clustering method (see Algorithm 1 below). An application of spectral clustering to motion segmentation can be found in [20]. Spectral curvature clustering [21], [22] is a variant of spectral clustering.

Algorithm 1 Spectral Clustering Algorithm

Require: Assume \( \{x_i\}_{i=1}^N \) are data points in \( \mathbb{R}^D \). Subspace clustering algorithms based on spectral clustering generally gets the following as input:

- A similarity (affinity) matrix \( S = (s_{ij}) \), where \( s_{ij} \) determines how “close” \( x_i \) is to \( x_j \). For example, \( s_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \) if \( i \neq j \) and \( s_{ii} = 1 \).
- The number of subspaces, \( m \).

1: Compute the diagonal degree matrix \( D = \text{Diag}(d_1, \cdots, d_N) \) where \( d_i = \sum_{j=1}^N s_{ij} \).
2: Compute the normalized graph Laplacian matrix \( L = D^{-1/2}SD^{-1/2} \). \( L \) is positive semi-definite with the smallest eigenvalue 0.
3: Compute the \( m \) eigenvectors \( u_1, \cdots, u_m \) corresponding to \( m \) highest eigenvalues.
4: Build the matrix \( W = [u_1 \ u_2 \ \cdots \ u_m] \in \mathbb{R}^{N \times m} \).
5: Apply a traditional clustering technique (such as k-means) in \( \mathbb{R}^m \) to the rows of \( W \).

The reduction methods initially build a data matrix that contains the data points as columns. They perform dimensionality reduction and noise elimination. This is typically done by factoring the data matrix by Singular Value Decomposition (SVD). These methods then generate an interaction (similarity) matrix to be used to cluster the data points in the original data matrix. Some related work can be found in [23], [24], [25], [26], [27]. The main drawback of these methods is their sensitivity to noise and outliers. Although some of the noise is reduced by SVD, it becomes difficult to eliminate the remaining noise. Yan [27] developed an algorithm that uses local affinity of the data points to cluster them. Each data point is first projected on a unit sphere and some neighboring points are found by calculating the angle between the points. Then a local subspace is fitted to the neighboring points. In other words, each point is represented by a local subspace. The distance between the local subspaces are calculated and a similarity matrix is computed. Finally, spectral clustering is applied to the similarity matrix. This method fails to cluster the data points around the intersection of the subspaces and it is not guaranteed that the algorithm will work even with perfect data matrix (noise and outliers free with no missing data points).

B. Paper Organization and Contributions

This paper presents new methods and algorithms for subspace segmentation and data clustering. The first algorithm is a general algorithm for solving the subspace segmentation problem described in Section I-A. It is based on the binary reduced row echelon form of a data matrix. Although the algorithm works perfectly well for noise free data, it does not perform well in noisy data such as motion segmentation.
A similar approach was proposed in [24] for motion segmentation without mathematical justification, but our paper provides a more general algorithm as well as proofs of the concepts used in the algorithm. Since our tests on real data (synthetic as well as the Hopkins 155 Dataset) did not produce good results, we devised a an algorithm for a special case when the subspace dimensions are equal. Such cases occur in applications such as motion segmentation and face recognition. The second algorithm is reliable in the presence of noise, and applied to the Hopkins 155 Dataset it generates the best results to date for motion segmentation.

The organization of the paper is as follows: Section II-B gives theoretical underpinning and algorithms for subspace segmentation. In Section II-C we give an algorithm based on row echelon form reduction of the data matrix. In Section II-D, we devise an algorithm for subspace segmentation problem in the special case where the number of subspaces is known and have equal dimensions. We apply and test our algorithm to the motion segmentation. Section III explains the experimental procedure and presents the experimental results.

II. SUBSPACE SEGMENTATION

This section will first describe the motion segmentation problem which is a special case of subspace segmentation problem. Then, we will present our methods for solving this type and more general problems.

A. Motion Segmentation Problem - Special Case

The appendix gives a detailed treatment of motion segmentation as a special case of the subspace segmentation problem. First, a data matrix $W_{2M \times N}$ is constructed using $N$ feature points that are tracked across $M$ frames. Then, each column of $W$ (i.e. the path of a feature point) is treated as a data point and it is shown that all of the data points that correspond to the same moving object lie in at most 4-dimensional subspace of $\mathbb{R}^{2M}$. The number of subspaces corresponds to the number of moving objects.

GPCA as an algebraic approach has been applied to motion segmentation problem [14], [15], [4]. The complexity of GPCA based algorithms grows exponentially as the number of moving objects increases. They are also sensitive to outliers. The statistical methods [24], [25], [16] seem to be sensitive to noise and outliers and they have too many parameters to tune. Factorization methods [26], [24], [28], [29] have difficulty in dealing with dependent motion and they are sensitive to noise, missing data components, and outliers. Spectral (curvature) clustering [20], [21], [22] and local subspace approximations approach [27] have also been applied to motion segmentation.

B. Preliminaries

This section presents some definitions and propositions that will be utilized in the theoretical development and algorithms of the following sections. Propositions [1] will be used later to justify that a data matrix $W$ whose columns represent data points can be replaced with a lower rank matrix after computing its SVD (i.e. $W = U\Sigma V^t$). It can be paraphrased by saying that for any matrices $A, B, C$, a cluster of the columns of $B$ is also a cluster of the columns of $C = AB$. A cluster of $C$ however is not necessarily a cluster $B$, unless $A$ has full rank:

**Proposition 1.** Let $A$ and $B$ be $m \times n$ and $n \times k$ matrices. Let $C = AB$. Assume $J \subset \{1,2, \cdots , k\}$.

1) If $b_i \in \text{span} \{b_j : j \in J\}$ then $c_i \in \text{span} \{c_j : j \in J\}$.

2) If rank($A$) is full and $m \geq n$ then $b_i \in \text{span} \{b_j : j \in J\} \iff c_i \in \text{span} \{c_j : j \in J\}$
Proof. The first part can be proved by the simple matrix manipulation
\[
AB = A \begin{bmatrix} b_1 & \cdots & b_i & \cdots & b_k \end{bmatrix} \\
= \begin{bmatrix} Ab_1 & \cdots & Ab_i & \cdots & Ab_k \end{bmatrix} \\
= \begin{bmatrix} Ab_1 & \cdots & A \sum_{j \in J} k_j b_j & \cdots & Ab_k \end{bmatrix} \\
= \begin{bmatrix} Ab_1 & \cdots & \sum_{j \in J} k_j Ab_j & \cdots & Ab_k \end{bmatrix} \\
= \begin{bmatrix} c_1 & \cdots & \sum_{j \in J} k_j c_j & \cdots & c_k \end{bmatrix}
\] (II.1)

For the second part, we note that $A^t A$ is invertible and $(A^t A)^{-1} A^t C = B$. We then apply part 1 of the proposition. Note that the same result clearly holds if $A$ is invertible. \hfill \Box

The proposition above suggest that—for the purpose of column clustering—we can replace a matrix $B$ by matrix $C$ as long as $A$ has the stated properties. Thus by choosing $A$ appropriately the matrix $B$ can be replaced by a more suitable matrix $C$, e.g. $C$ has fewer rows, is better conditioned or is in a format where columns can be easily clustered. One such useful format is if $C$ is a row echelon form matrix as will be demonstrated in reduction method of Section II-C. In fact, the first $r$ rows of the reduced echelon form of $C = AB$ and $B$ are the same if $B$ has rank $r$:

**Proposition 2.** Let $A$ be an $m \times n$ full rank matrix, $B$ be an $n \times k$ matrix with $m \geq n$ and rank $r$. Then the first $r$ rows of the reduced row echelon forms of $AB$ and $B$ are the same.

In particular in the absence of noise, a data matrix $W$ with the SVD $W = U \Sigma V^t$ has the same reduced row echelon form as that of $V^t$ up to its rank $r$. This fact together with Proposition [1] will help us devise a reduction algorithm for subspace clustering.

Before proving Proposition [2] recall that there are three elementary row operations that can be used to transform a matrix to its unique reduced row echelon form. The three elementary row operations can be performed by the elementary row operation matrices:

**Definition 1.** Let $A^{(i)}$ be the $i^{\text{th}}$ row of $A$. The three types of elementary row operation matrices are:

\[
E_{i,j} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 0 & 1 & \\ & & 1 & \ddots & \\ & & & & 1 \end{bmatrix}, \quad E_i(c) = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & e_i \\ & & e_i & \ddots \\ & & & 1 \end{bmatrix}, \quad E_{i,j}(c) = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & c \\ & & c & \ddots \\ & & & 1 \end{bmatrix}
\]

Then $E_{i,j} A^{(i)} = A^{(j)}$ corresponds to row-switching, $E_i(c) A^{(i)} = c A^{(i)}$ corresponds to row-multiplication ($c \neq 0$), and $E_{i,j}(c) A^{(i)} = A^{(i)} + c A^{(j)}$ corresponds to row-addition.

**Proof of Proposition 2.** The reduced row echelon form of $A$ can be obtained by $\{E_1, \cdots, E_k\}$ row-switching, row-multiplication, and row-addition operations:

\[
\text{rref}(A) = E_k \cdots E_1 A = \begin{bmatrix} 1 & \cdots & 1 \\ 0 & \ddots & \vdots \\ \vdots & \cdots & 1 \\ 0 & 0 & \cdots \\ 0 & 0 & \cdots \\ 0 & 0 & \cdots \end{bmatrix} = [I_{n \times n}]_{m \times n}
\] (II.2)

Apply the same elementary row operations to $AB$:

\[
D := (E_k \cdots E_1) AB = (E_k \cdots E_1 A) B = \begin{bmatrix} I_{n \times n} & \\ \end{bmatrix} B_{n \times k} = \begin{bmatrix} B & \\ \end{bmatrix}_{m \times k}
\] (II.3)
\[
\text{rref}(D) = \text{rref}\left(\begin{bmatrix} B \\ 0 \end{bmatrix}\right) 
\] (II.4)

Since \(\text{rref}(D) = \text{rref}(AB)\), we get \(\text{rref}(D) = \text{rref}\left(\begin{bmatrix} B \\ 0 \end{bmatrix}\right)\). Also, since the rank of \(AB\) and \(B\) are \(r\), the first \(r\) rows of the reduced row echelon forms of \(AB\) and \(B\) are the same. \(\square\)

Corollary 1 will be utilized in the development of our subspace segmentation algorithm based on the reduced row echelon form.

**Corollary 1.** Let \(W\) be represented as \(USV^t\) by SVD. The reduced row echelon form of \(W\) up to \(r\) (rank of \(W\)) rows is equal to the reduced row echelon form of the first \(r\) rows of \(V^t\).

**Proof.** Since the rank of \(W\) is \(r\), we get \(W = U_r \Sigma_r (V_r)^t\), where \((V_r)^t\) is the first \(r\) rows of \(V^t\). Also, \(U_r \Sigma_r\) is a full rank matrix. We conclude by applying Proposition 2. \(\square\)

Next we state two definitions that are useful for the remainder of the paper. The first definition is equivalent to the definition of self-expressive data from [12].

**Definition 2.** Let \(S\) be a linear subspace of \(\mathbb{R}^D\) with dimension \(d\). The data drawn from \(S\) is said to be generic if every \(d\) subset of data forms a basis for \(S\) and there are more data points than \(d\).

**Definition 3.** Matrix \(R\) is said to be the binary reduced row echelon form of matrix \(A\) if all non-pivot column vectors converted to binary vectors, i.e., non-zero entries are set to one.

### C. Row Echelon Form Approach

Assume that the data is drawn from a union of linearly independent subspaces, i.e., from \(U = \bigcup_{i=1}^{3} S_i\) where \(\{S_i \cap S_j = 0\}_{i \neq j}\). We treat noise-free and noisy cases separately.

1) **Noiseless Case:** We first present a theorem (whose proof is postponed after an example) for determining the number of subspaces, a base for each subspace, and the data clusters.

**Theorem 1.** Let \(\{S_i\}_{i=1}^{k}\) be a set of linearly independent subspaces of \(\mathbb{R}^D\) with corresponding dimensions \(\{d_i\}_{i=1}^{k}\). Let \(W = [W_1 \cdots W_N] \in \mathbb{R}^{D \times N}\) be a matrix whose columns are drawn from \(\bigcup_{i=1}^{k} S_i\). Assume the data drawn from each subspace is generic. Let \(R = [R_1 \cdots R_N]\) be the binary reduced row echelon form of \(W\).

1) Let \(J \subset \{1, \cdots, N\}\) be such that \(R_j = R_k \forall j, k \in J\). Then \(\{W_j\}_{j \in J}\) are drawn from the same subspace.

2) The dimension of each subspace is equal to the \(l_1\)-norm of a corresponding non-pivot column vector in \(R\).

3) The inner product of a pivot column and a non-pivot column in \(R\) is one, if and only if the corresponding column vectors in \(W\) belong to the same subspace.

This theorem suggests a very simple yet effective approach to cluster the data points. We simply find the reduced row echelon form of the data matrix \(W\) and cluster the data points according to the corresponding columns in the reduced row echelon form. For example, consider the following data matrix that contains data points drawn from the union of three (3) linearly independent subspaces (i.e. from \(U = \bigcup \{S_i\}_{i=1}^{3}\) such that \(S_i \cap S_j = 0\) for \(i, j = 1, 2, 3\) as column vectors.

\[
W = \begin{bmatrix}
2006 & 4292 & 352 & 7240 & 2861 & 3072 & 1847 & 665 & 6968 & 646 & 1709 & 2794 & 2080 & 2366 & 1012 \\
5803 & 1405 & 657 & 2498 & 549 & 864 & 3854 & 687 & 2158 & 390 & 629 & 628 & 4711 & 4654 & 545 \\
5124 & 744 & 2092 & 1335 & 602 & 1056 & 2835 & 1110 & 1131 & 484 & 774 & 762 & 867 & 4546 & 309 \\
6701 & 3192 & 757 & 5420 & 775 & 1248 & 4502 & 578 & 5148 & 578 & 919 & 896 & 5638 & 5354 & 812 \\
7102 & 1625 & 802 & 2862 & 888 & 1440 & 4793 & 522 & 2522 & 672 & 1064 & 1030 & 6059 & 5866 & 585 \\
495 & 223 & 117 & 577 & 322 & 960 & 266 & 247 & 169 & 668 & 866 & 520 & 275 & 430 & 388 \\
2065 & 1117 & 287 & 3027 & 4040 & 4800 & 1376 & 159 & 715 & 1360 & 2920 & 4100 & 1797 & 1662 & 2172
\end{bmatrix}
\] (II.5)
The computed (binary) reduced row echelon form of \( W \) is \( W_b \):

\[
W_b = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Then, our algorithm (Algorithm 3, below) correctly clusters columns of \( W \) as \{1, 3, 7, 8, 13, 14\}, \{2, 4, 9, 15\}, and \{5, 6, 10, 11, 12\}.

**Proof of Theorem 1.** Apply a series of row-switching, row-multiplying, and row-addition operations \( E_1, \ldots, E_k \) to \( W \) to obtain the reduced row echelon form of \( W \).

\[
(E_k \cdots E_1)W_{D \times N} = \text{rref}(W)_{D \times N} = \begin{bmatrix} F_{r \times N} \\ 0 \end{bmatrix}
\]

where \( r \) is the rank of \( W \). Let \( E := E_k \cdots E_1 \). Then \( W = E^{-1} \begin{bmatrix} F \\ 0 \end{bmatrix} \). Since \( E^{-1} \) is invertible, its columns are linearly independent. Let \( C_r \) be a matrix consisting of the first \( r \) columns of \( E^{-1} \). Then we get

\[
W = C_rF
\]

Note that a column vector in the reduced echelon form \( F \) corresponding to a pivot has exactly one entry whose value is 1 and all other entries are zeros. Thus, it is clear that \( C_r \) consists of the \( r \) linearly independent columns of \( W \) in the same order as the columns of \( F \) that corresponds to pivots (see example below). Then, \( C_r \) multiplied by a non-pivot column of \( F \) will be a combination of some linearly independent columns of \( W \). Therefore, the non-pivot columns of \( F \) that have nonzero entries in the same indices will be in the span of the same subspace. The same is valid for the binary reduced row echelon form \( R \). This proves part 1 of the theorem. Clearly, if the inner product of a pivot column and a non-pivot column of \( F \) is nonzero, then they belong to the same subspace, from which part 3 follows. Part 2 is then a consequence of the assumption that the data \( W \) is generic.

**Example 1.** Let \( W = \begin{bmatrix} 2 & 4 & 1 & 2 \\ 2 & 4 & 2 & 5 \\ 2 & 4 & 3 & 8 \\ 2 & 4 & 4 & 10 \\ 2 & 4 & 5 & 20 \end{bmatrix} \), where 1\(^{st}\), 3\(^{rd}\), and 4\(^{th}\) columns are linearly independent.

Then,

\[
\text{rref}(W) = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad F = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad C_r = \begin{bmatrix} 2 & 1 & 2 \\ 2 & 2 & 5 \\ 2 & 3 & 8 \\ 2 & 4 & 10 \\ 2 & 5 & 20 \end{bmatrix}
\]

Clearly, \( C_r \) consists of the linearly independent columns of \( W \) and \( W = C_rF \).
2) **Noisy Case:** Since the data is high dimensional and corrupted by noise, the first step is to project \( W \) into a lower dimensional subspace. This can be achieved by SVD. Let \( W_{m \times n} \) be the data matrix with \( m > n \). Then,

\[
W = U \Sigma V^t = [u_1 \quad u_2 \quad \ldots \quad u_m]
\]

The rank \( r \) of \( W \) can be estimated by using the following model selection algorithm [27]:

\[
r = \arg \min_r \frac{\sigma_{r+1}^2}{\sum_{i=1}^r \sigma_i^2} + \kappa r
\]

where \( \sigma_j \) is the \( j^{th} \) singular value and \( \kappa \) is a suitable constant. \( U_r \Sigma_r (V_r)^t \) is the best rank-\( r \) approximation of \( W = U \Sigma V^t \), where \( U_r \) refers to a matrix that has the first \( r \) columns of \( U \) as its columns. After the rank is estimated using Equation (II.10), the first \( r \) rows of \( V^t \) (:= \( (V_r)^t \)) replaces the data matrix \( W \). This is a consequence of Proposition 1. Note that the data points in \( W \) are \( m \) dimensional columns, whereas they are \( r \) dimensional in \( (V_r)^t \). For the case of motion detection, if there are \( k \) independent motions across the frames captured by a moving camera, the rank of \( W \) is between \( 2(k+1) \) and \( 4(k+1) \). SVD reduces noise by discarding the small singular values in the factorization.

**Corollary 2.** Let \( W \) be the data matrix of independent motions with rank \( r \). Let \( R \) be the reduced row echelon form of \( (V_r)^t \) with all non-pivot column vectors converted to binary vectors. Then, the indices of the column vectors that are equal in \( R \) correspond to the indices of the data vectors (in \( W \)) drawn from the same subspace.

**Proof.** Apply Corollary 1 and Theorem 1 to \( W \).

3) **Algorithm - Reduced Row Echelon Based:** Algorithm 2 summarizes the algorithm for subspace clustering when the data points are not corrupted by noise. Note that if we know a single data point from each subspace (i.e. the path of a feature point from each object or a column in the data matrix \( W \)), we can find all the subspaces (with their dimensions and bases) using only the reduced row echelon form \( R \) of \( W \). For example, if we know that Columns 8, 9, and 10 of \( W \) in Equation (II.5) belong to the subspaces \( S_1 \), \( S_2 \), and \( S_3 \), respectively, then we can just use Columns 8, 9, and 10 of Equation (II.6) to determine that \( \{e_1, e_3, e_7\}, \{e_2, e_4\}, \{e_5, e_6\} \) are the corresponding basis vectors, where \( e_i \) is \( i^{th} \) unit vector in \( \mathbb{R}^{10} \). Since the noise will make \( W \) a full rank matrix, we modify the algorithm as in Algorithm 3. The data matrix \( W \) is replaced by \( (V_r)^t \) by Proposition 1. The effective rank estimation using Equation (II.10) typically works well for independent motions. This step does not only reduce the dimensionality but also it reduces the noise. We must further reduce the noise by thresholding the entries of the \( (V_r)^t \) at each step of the reduced row echelon form calculations. However, although the echelon form algorithms work well for synthetic data and relatively low level of noise, it did not produce good results for moderate noise level or when we tested it on the object motion data. We believe that the reason for this non-reliability is that we do not have a good thresholding mechanism of the data matrix entries during the echelon form calculations. Gear [24] proposed a statistical algorithm to eliminate the noise in a similar case, however, his algorithm was not successful in general settings either.

**D. Null Space Approach**

In this section, we develop a specialized algorithm for the case when the dimensions of the subspaces are equal and known.
Algorithm 2 Subspace Segmentation - Row Echelon Form Approach - No Noise

1: Find the reduced row echelon form $R$ of $W$.
2: for all non-pivot columns of $R$ do
3: Set 1 for each nonzero entry.
4: end for
5: Call this matrix $S$
6: Set $i = 1$
7: while the non-pivot column vector $c_i$ of $S$ exists do
8: Separate $c_i$ into unit vectors $u_1^i, \ldots, u_d^i$. {These vectors form a basis for one of the subspaces, e.g., $S_i$.}
9: Set $i = i + 1$
10: Pick $c_i$ of $S$.
11: if $c_i$ does not exist then
12: break the loop
13: end if
14: if $c_i \in \text{span}(u_j^1, \ldots, u_j^{d_j})$ for some $j$ then
15: $c_i \in S_j$ and go to (11),
16: end if
17: end while

Algorithm 3 Subspace Segmentation - Row Echelon Form Approach - Noisy Case

1: Generate the data matrix $W$ {Note that $W_{2F \times N}$ can be obtained by placing the paths as column vectors as in Equation (2) for motion segmentation}
2: Compute the SVD of $W$ and find $(V_r)^t$ as in Equation (II.9).
3: Estimate the rank of $W$ using Equation (II.10).
4: Compute $(V_r)^t$ consisting of the first $r$ rows of $V^t$.
5: Replace the data matrix $W$ with $(V_r)^t$.
6: Follow Algorithm 2 {While finding $R$, threshold all of the entries of $W$ before each column operation.}

Let $W$ (e.g., $W = W_{2F \times N}$ for motion segmentation) be the data matrix with $W = U_r \Sigma_r (V_r)^t$, where $r$ is the effective rank of $W$. We replace $W$ by $(V_r)^t$ by Proposition 1 as before. The null space of every subspace of dimension 1 is a hyperplane in $\mathbb{R}^r$. This subspace corresponds to a point when projected onto the unit sphere $S^{r-1}$. If we can detect all such points, then we can peel the corresponding data points from the data matrix and repeat the process for subspaces of dimension 2. The process can be repeated until all the data points are consumed. An illustration is given in Figure 1 for $\mathbb{R}^3$. All the data points in a line (1-dimensional subspace of $\mathbb{R}^3$) are projected to a point in for $S^2$ and all the data points on a plane (2-dimensional subspace of $\mathbb{R}^3$) are projected on to a circle.

In general, the data points that come from a $k$-dimensional subspace lie in a $k$-flat of $\mathbb{R}^r$. Our approach finds (or approximates) a perpendicular subspace (of dimension $n - k$) for each data point of this subspace. We assume that we know the dimension of the subspaces. In motion segmentation problem (see ), each data point (each path) will lie in a (at most) 4 dimensional subspace. If there are $n$ independently moving objects, then the rank of the data matrix will be at most $4n$. In this specific problem, we can consider an algorithm for detecting the subspaces of dimension (at most) 4 in $\mathbb{R}^{4n}$.

1) Algorithm for Motion Segmentation: An algorithm for motion segmentation using Null Space approach is given in Algorithm 4. The details of various steps are as follows:

- **Step-1:** The SVD of $W$ as in Equation (II.9) is computed and the rank of $W$ is estimated using Equation (II.10). Then, $(V_r)^t$ consists of the first $r$ rows of $V^t$. 
• **Step-2:** The data matrix $W$ can be replaced by $(V_r)^t$ (by Proposition I).

• **Step-3:** The columns of $(V_r)^t$ are normalized so that all the data points on a flat in $\mathbb{R}^r$ will be reflected onto $S^{r-1}$ as illustrated in Figure 1. Note that the normalization can be done by using $l_p$ norms of the columns of $(V_r)^t$. The difference between them is illustrated in Figure 2.

• **Step-4:** The neighbors of each point (i.e., each column of $(V_r)^t$) are found. This can be done in different ways. For example, if $l_2$-norm is used for normalization in Step-3, we can find the angles between the points, i.e., we can compute the matrix $\arccos(V_r \times (V_r)^t)$. Then we can sort the angles and find the closest neighbors of each point. If we use $l_p$-norm for normalization in Step-3, we can generate a distance matrix $(a_{i,j}) = (||x_i - x_j||_p)$ and then sort each column to find the neighbors of each $x_i$, which is the $i^{th}$ column of $(V_r)^t$.

• **Step-5:** For each point $x_i$, (i.e., the $i^{th}$ column of $(V_r)^t$), we generate a set of $k+1$ points $\{x_i, x_{i_1}, ..., x_{i_k}\}$ consisting of $x_i$ and its $k$ closest neighbors found in Step-4. Then we generate a 4-dimensional subspace that best approximates (in the least square sense) the data $\{x_i, x_{i_1}, ..., x_{i_k}\}$. This is accomplished by using SVD

\[
X = [x_i \ x_{i_1} \ ... \ x_{i_k}] = A_4 \Sigma_4 (B_4)^t
\]  

The row space $R(A_4)$ of $A_4$ is the best approximation (in the least square sense) of the 4-dimensional


Algorithm 4 Motion Segmentation - Hypercircle Detection Approach

Require: The data matrix $W_{2F \times N}$
Ensure: Clustering of the feature points.

1: Compute the SVD of $W$ as in Equation (II.9).
2: Estimate the rank of $W$ using Equation (II.10).
3: Compute $(V_r)^t$ consisting of the first $r$ rows of $V^t$.
4: Normalize the columns of $(V_r)^t$.
5: Replace the data matrix $W$ with $(V_r)^t$.
6: Find the angle between the column vectors (paths) of $W$ and represent it as a matrix. \{ i.e., \arccos(W^t \times W) \}.
7: Sort the angles and find the closest neighbors of column vector.
8: for all Column vector $x_i$ of $W$ do
9: Find a basis for the null space of the set consisting of $x_i$ and $k$ neighbors (see Equation (II.11)). \{Theoretically, $k$ is at least 3. We can use the least square approximation for the null space as in Step-5 of the detailed explanation. Let $\{ B_{i1}, B_{i2}, ..., B_{i4(n-1)} \}$ be a basis for such null space association for $x_i$ \}
10: end for
11: for $i = 1$ to $N$ do
12: for $j = 1$ to $N$ do
13: define $Dis = (d_{ij}) = (||B_i \times x_j||_p + ||B_j \times x_i||_p) / 2$
14: end for \{Version-1 for distance matrix\}
15: for $j = i$ to $N$ do
16: define $Dis = (d_{ij}) = (||B_i \times x_j||_p) / 2$
17: end for \{Version-2 for distance matrix\}
18: end for \{Build the distance matrix\}
19: Sort the entries of $Dis_{N \times N}$ from smallest to highest values and find the index $(k, l)$ that corresponds to round$(N \times N/(2n))^{th}$ highest entry value. \{Compute the threshold\}
20: All entries of $Dis$ less than threshold are set to 1 and set to 0 otherwise and $S$ is constructed. \{Build the binary similarity matrix\}
21: Let $S = (S + S^t)/2$ to ensure that the similarity matrix is symmetric.
22: Normalize the rows of $S$ using $l_1$-norm.
23: Perform SVD $S^t = U_n^{(s)} \Sigma_n^{(s)} (V_n^{(s)})^t$.
24: Cluster the columns of $(V_n^{(s)})^t$ using k-means.

---

flat. Then we find the null space $N(A_4)$ of $A_4$, which is orthogonal to the approximating flat. Notice that $N(A_4)$ is $4(n-1)$ dimensional, where $n$ is the number of motions. Then, each point $x_j$ is associated with a $4(n-1)$-dimensional subspace. Let $\{ B_{i1}, B_{i2}, ..., B_{i4(n-1)} \}$ be a basis for such null space association to $x_i$.

- **Step-6:** If a point $x_j$ lies on the same flat as $x_i$, then both are expected to be orthogonal to the associated null space for $x_i$. Construct

$$B_i = \begin{bmatrix} B_{i1}^t \\ B_{i2}^t \\ \vdots \\ B_{i4(n-1)}^t \end{bmatrix} \quad \text{(II.12)}$$

- **Step-7:** A distance matrix $Dis$ is generated as $Dis = (d_{ij}) = (||B_i \times x_j||_p + ||B_j \times x_i||_p) / 2$. 

A convenient choice of $p$ is 2. Note that as $d_{ij}$ decreases, the probability of having $x_j$ on the same flat increases (We can use $Dis = (d_{ij}) = (||B_i \times x_j||_p)$ for $i = 1, \ldots, N$ and $j = i, \ldots, N$ but this matrix is not symmetric).

- **Step-8:** Convert the distance matrix $Dis = (d_{ij})_{N \times N}$ into a similarity matrix $S = (s_{ij})$. This can be done by applying an adaptive thresholding as follows:
  1. Sort the entries of $Dis_{N \times N}$ from smallest to highest values.
  2. Find the index $(k, l)$ that corresponds to $\text{round}(N \times N/(2n))^{th}$ highest entry value.
  3. Let $S = Dis < \text{threshold}$, i.e., all entries of $Dis$ less than threshold are set to 1 and set to 0 otherwise.
  4. Let $S = (S + S^t)/2$ to ensure that the similarity matrix is symmetric.

- **Step-9:** It is important to see that the initial data segmentation problem has now been converted to segmentation of $n$ 1-dimensional subspaces from the rows of $S$. This is because

$$S = \begin{bmatrix}
s_1^t \\
\vdots \\
s_N^t
\end{bmatrix}$$

and if $x_i$ and $x_j$ are similar (i.e., in the same subspace), the rows $s_i^t$ and $s_j^t$ are equal. Since there are $n$ motions, then there will be $n$ 1-dimensional subspaces.

- **Step-10:** The rows of $S$ are normalized using $l_1$-norm, i.e., $S_N = D \times S$, where $D$ is a diagonal matrix $(d_{ij}) = \sum_{j=1}^N s_{ij}$. Note that although $S$ is symmetric, $S_N$ does not need to be symmetric (if we use $S_N = D^{-1/2}SD^{-1/2}$, then $S_N$ becomes symmetric). We can replace $S$ with $S_N$ after normalization.

- **Step-11:** Now, the problem is again a subspace segmentation problem, but this time the data matrix is $S$ with each row as a data point. Also, each subspace is 1-dimensional and there are $n$ subspaces. Therefore, we can apply the same approach. In other words, we can use SVD as

$$S^t = U^{(s)}n\Sigma^{(s)}n(V^{(s)}n)^t$$

and cluster the columns of $(V^{(s)}n)^t$, that is, the rows of $V^{(s)}n$. Note that $V^{(s)}n$ is $N \times n$. By Proposition 1, $(V^{(s)}n)^t$ replaces $S^t$. now, since the problem is only segmentation of subspaces of dimension 1, we can use any traditional segmentation algorithm such as k-means to cluster the data points.

### III. Experimental Results

#### A. The Hopkins 155 Dataset

The Hopkins 155 Dataset [15] was created as a benchmark database to evaluate motion segmentation algorithms. It contains two (2) and three (3) motion sequences. There are three (3) groups of video sequences in the dataset: (1) 38 sequences of outdoor traffic scenes captured by a moving camera, (2) 104 indoor checker board sequences captured by a handheld camera, and (3) 13 sequences of articulated motions such as head and face motions. Cornerness features that are extracted and tracked across the frames are provided along with the dataset. The ground truth segmentations are also provided for comparison. Figure 3 shows two (2) samples from the dataset with the extracted features.

#### B. Results

Tables I, II, and III display some of the experimental results for the Hopkins 155 Dataset. Our Null Space (NS) detection approach have been compared with five (5) motion detection algorithms: (1) GPCA [14], (2) RANSAC [18], (3) Linear Subspace Affinity (LSA) [27], (4) MLS [2], [16], and (5) Sparse Subspace Clustering (SSC) [12]. An evaluation of those algorithms is presented in [12] with a small error for articulated three motion analysis of SSC-N. SSC-B and SSC-N correspond to Bernoulli and Normal
random projections, respectively [12]. Table I displays the misclassification rates for the two motion video sequences. NS outperforms all of the algorithms for the checkerboard sequences, which are linearly independent motions. The overall misclassification rate is 0.79% when 3 neighboring points are used for null space calculations with distance matrix version-1 (NS-k3). It is 0.69% when 4 neighboring points are used for least square approximation of the null space with distance matrix version-2 (NS-k4). Table II shows the misclassification rates for the three motion sequences. NS-k3 has 1.15% misclassification rate and performs 53% better than the next best algorithm (i.e. SSC-N). Table III presents the misclassification rates for all of the video sequences. Our algorithm NS-k3 (with 0.89% misclassification rate) performs 28% better than the next best algorithm (i.e. SSC-N). In general, our algorithms outperforms SSC-N, which is given as the best algorithm for the two and three motion sequences together.

<table>
<thead>
<tr>
<th>Checker (78)</th>
<th>GPCA</th>
<th>LSA</th>
<th>RANSAC</th>
<th>MSL</th>
<th>ALC</th>
<th>SSC-B</th>
<th>SSC-N</th>
<th>NS-k4</th>
<th>NS-k3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>6.09%</td>
<td>2.57%</td>
<td>6.52%</td>
<td>4.46%</td>
<td>1.35%</td>
<td>0.83%</td>
<td>1.12%</td>
<td>0.22%</td>
<td>0.35%</td>
</tr>
<tr>
<td>Median</td>
<td>1.03%</td>
<td>0.27%</td>
<td>1.75%</td>
<td>0.00%</td>
<td>0.29%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Traffic (31)</td>
<td>GPCA</td>
<td>LSA</td>
<td>RANSAC</td>
<td>MSL</td>
<td>ALC</td>
<td>SSC-B</td>
<td>SSC-N</td>
<td>NS-k4</td>
<td>NS-k3</td>
</tr>
<tr>
<td>Average</td>
<td>1.41%</td>
<td>5.43%</td>
<td>2.53%</td>
<td>2.23%</td>
<td>1.59%</td>
<td>0.23%</td>
<td>0.02%</td>
<td>0.92%</td>
<td>2.24%</td>
</tr>
<tr>
<td>Median</td>
<td>0.00%</td>
<td>1.48%</td>
<td>0.21%</td>
<td>0.00%</td>
<td>1.17%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.42%</td>
<td>0.41%</td>
</tr>
<tr>
<td>Articulated (11)</td>
<td>GPCA</td>
<td>LSA</td>
<td>RANSAC</td>
<td>MSL</td>
<td>ALC</td>
<td>SSC-B</td>
<td>SSC-N</td>
<td>NS-k4</td>
<td>NS-k3</td>
</tr>
<tr>
<td>Average</td>
<td>2.88%</td>
<td>4.10%</td>
<td>7.25%</td>
<td>7.23%</td>
<td>10.70%</td>
<td>1.63%</td>
<td>0.62%</td>
<td>2.33%</td>
<td>2.36%</td>
</tr>
<tr>
<td>Median</td>
<td>0.00%</td>
<td>1.22%</td>
<td>2.64%</td>
<td>0.00%</td>
<td>0.95%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.88%</td>
<td>0.88%</td>
</tr>
<tr>
<td>All (120 seq)</td>
<td>GPCA</td>
<td>LSA</td>
<td>RANSAC</td>
<td>MSL</td>
<td>ALC</td>
<td>SSC-B</td>
<td>SSC-N</td>
<td>NS-k4</td>
<td>NS-k3</td>
</tr>
<tr>
<td>Average</td>
<td>4.59%</td>
<td>3.45%</td>
<td>5.56%</td>
<td>4.14%</td>
<td>2.40%</td>
<td>0.75%</td>
<td>0.82%</td>
<td>0.69%</td>
<td>0.79%</td>
</tr>
<tr>
<td>Median</td>
<td>0.38%</td>
<td>0.59%</td>
<td>1.18%</td>
<td>0.00%</td>
<td>0.43%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

TABLE I

% CLASSIFICATION ERRORS FOR SEQUENCES WITH TWO MOTIONS.

IV. CONCLUSIONS AND OPEN ISSUES

It should be noted that, in the noise free cases, the row echelon based approach works even when subspaces have non-trivial intersections. However, as we have discussed we still do not know how to make this method reliable in the presence of moderate noise. Reliability of this method seems to be a difficult problem and maybe a topic for future exploration.

On the other hand, the Null Space approach can handle noise effectively, but it works only in special cases of subspaces segmentation problems as explained earlier. It may be possible to extend Algorithm...
for general subspace segmentation problem. One possible approach is through a peeling process: the data drawn from the lowest dimensional subspace are peeled out. Then, the process is repeated in a lower dimension until all of the data points are consumed as explained in Section II-D.

In this paper, we considered the ambient space $\mathcal{H}$ to be finite dimensional. There may be situations in which the ambient space is better modeled an infinite dimensional Hilbert space. Such cases can be found in analog signal processing and modeling. This topic has theoretical as well as practical appeal and maybe another subject of future exploration.

### APPENDIX

Assume that there is a rigid body that rotates around a vector in a given coordinate frame as shown in Figure 4. We define two coordinate frames: (1) World Frame $(X, Y, Z)$ and (2) Object Frame $(x, y, z)$. Initially World Frame and Object Frame coincide. As the object rotates, Object Frame deviates from World Frame. Let $r_1$, $r_2$, and $r_3$ be the orthogonal unit vectors of Object Frame (which forms an orthonormal basis for $\mathbb{R}^3$). Let $p \in \mathbb{R}^3$ be a feature point on the object. Let $p_w = [X_p \ Y_p \ Z_p]^T$ and $p_o = [x_p \ y_p \ z_p]^T$ be coordinates of $p$ with respect to World Frame and Object Frame, respectively. Since $\{r_1, r_2, r_3\}$ forms an orthonormal basis for $\mathbb{R}^3$, we have $p_w = x_p r_1 + y_p r_2 + z_p r_3$. That is, $p_w = [r_1 \ r_2 \ r_3]^T p_o = Rp_o$, where $R$ is a rotation matrix. If the object both rotates and translates (Figure 5), then $p_w = Rp_o + t_w$, where $t_w$ is the world coordinates of the center of the object. If the object is sufficiently away from the camera, then camera projection can be modeled as affine which generalizes orthographic, weak perspective, and paraperspective projection [30]. In Figure 6, $Z$-axis is assumed to be the optical axis of the camera. Therefore, the projection is parallel to $Z$-axis. We then can write

$$p_w(k) = R(k)p_o + t_w(k)$$

$$\left[ \begin{array}{l} X_p(k) \\ Y_p(k) \\ Z_p(k) \end{array} \right] = \left[ \begin{array}{ccc} r_1(k) & r_2(k) & r_3(k) \end{array} \right] \left[ \begin{array}{l} a_p \\ b_p \\ c_p \end{array} \right] + \left[ \begin{array}{l} X_t(k) \\ Y_t(k) \\ Z_t(k) \end{array} \right]$$

$$\left[ \begin{array}{l} X_p(k) \\ Y_p(k) \end{array} \right] = a_p \tilde{r}_1(k) + b_p \tilde{r}_2(k) + c_p \tilde{r}_3(k) + \tilde{t}_w(k)$$

(1)

where $\tilde{r}_1(k), \tilde{r}_2(k), \tilde{r}_3(k)$, and $\tilde{t}_w(k)$ correspond to $r_1(k), r_2(k), r_3(k)$, and $t_w(k)$ with $3^{rd}$ rows truncated.

### TABLE II

% CLASSIFICATION ERRORS FOR SEQUENCES WITH THREE MOTIONS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Checker (26)</th>
<th>Traffic (7)</th>
<th>Articulated (2)</th>
<th>All (35 seq)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Median</td>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td>GPCA</td>
<td>19.55%</td>
<td>16.85%</td>
<td>16.85%</td>
<td>28.66%</td>
</tr>
<tr>
<td>LSA</td>
<td>23.79%</td>
<td>7.25%</td>
<td>21.38%</td>
<td>9.73%</td>
</tr>
<tr>
<td>RANSAC</td>
<td>11.45%</td>
<td>21.08%</td>
<td>12.83%</td>
<td>22.03%</td>
</tr>
<tr>
<td>MSL</td>
<td>0.00%</td>
<td>2.71%</td>
<td>1.80%</td>
<td>5.76%</td>
</tr>
<tr>
<td>ALC</td>
<td>2.97%</td>
<td>1.60%</td>
<td>0.61%</td>
<td>0.67%</td>
</tr>
<tr>
<td>SSC-B</td>
<td>0.49%</td>
<td>21.08%</td>
<td>7.75%</td>
<td>6.69%</td>
</tr>
<tr>
<td>SSC-N</td>
<td>0.00%</td>
<td>1.60%</td>
<td>0.49%</td>
<td>3.56%</td>
</tr>
<tr>
<td>NS-k4</td>
<td>0.72%</td>
<td>20.55%</td>
<td>0.58%</td>
<td>1.24%</td>
</tr>
<tr>
<td>NS-k3</td>
<td>0.86%</td>
<td>4.25%</td>
<td>1.37%</td>
<td>1.57%</td>
</tr>
</tbody>
</table>

### TABLE III

% CLASSIFICATION ERRORS FOR ALL SEQUENCES.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPCA</td>
<td>10.34%</td>
<td>2.54%</td>
</tr>
<tr>
<td>LSA</td>
<td>4.94%</td>
<td>0.90%</td>
</tr>
<tr>
<td>RANSAC</td>
<td>9.76%</td>
<td>3.21%</td>
</tr>
<tr>
<td>MSL</td>
<td>5.03%</td>
<td>0.00%</td>
</tr>
<tr>
<td>ALC</td>
<td>3.56%</td>
<td>0.50%</td>
</tr>
<tr>
<td>SSC-B</td>
<td>1.45%</td>
<td>0.00%</td>
</tr>
<tr>
<td>SSC-N</td>
<td>1.24%</td>
<td>0.00%</td>
</tr>
<tr>
<td>NS-k4</td>
<td>1.57%</td>
<td>0.22%</td>
</tr>
<tr>
<td>NS-k3</td>
<td>0.89%</td>
<td>0.18%</td>
</tr>
</tbody>
</table>
Figure 7 illustrates the projection of a feature point on the camera frame as the object moves. Since the moving body is considered to be rigid, all of the feature points move together with the same translations and rotations. Let $X_{r(k)}$ and $Y_{r(k)}$ be coordinates (in World Frame) of the $r^{th}$ feature point in frame $k$.

We can then define the following data matrix for $N$ feature points collected across $M$ frames:

$$
W = \begin{bmatrix}
X_{1(1)} & X_{2(1)} & \cdots & X_{k(1)} & \cdots & X_{N(1)} \\
Y_{1(1)} & Y_{2(1)} & \cdots & Y_{k(1)} & \cdots & Y_{N(1)} \\
X_{1(2)} & X_{2(2)} & \cdots & X_{k(2)} & \cdots & X_{N(2)} \\
Y_{1(2)} & Y_{2(2)} & \cdots & Y_{k(2)} & \cdots & Y_{N(2)} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
X_{1(M)} & X_{2(M)} & \cdots & X_{k(M)} & \cdots & X_{N(M)} \\
Y_{1(M)} & Y_{2(M)} & \cdots & Y_{k(M)} & \cdots & Y_{N(M)}
\end{bmatrix}_{2M \times N}
$$

where $k^{th}$ column of $W$ corresponds to the coordinates of the $k^{th}$ feature point across $M$ frames and $(2i - 1)^{th}$ and $(2i)^{th}$ rows correspond to the coordinates of $N$ feature points in $i^{th}$ frame. We get the following by using Equation (1):
Affine Camera Projection Model.

\[
Path_k = \begin{bmatrix}
  X_{k(1)} \\
  Y_{k(1)} \\
  X_{k(2)} \\
  Y_{k(2)} \\
  \vdots \\
  X_{k(M)} \\
  Y_{k(M)}
\end{bmatrix}_{2M \times 1} = q_1 + a_k q_2 + b_k q_3 + c_k q_4
\] (3)

where

\[
q_1 = \begin{bmatrix}
  \tilde{X}_{t(1)} \\
  \tilde{Y}_{t(1)} \\
  \vdots \\
  \tilde{Y}_{t(M)}
\end{bmatrix}_{2M \times 1}
q_2 = \begin{bmatrix}
  \tilde{X}_{r_1(1)} \\
  \tilde{Y}_{r_1(2)} \\
  \vdots \\
  \tilde{Y}_{r_1(M)}
\end{bmatrix}_{2M \times 1}
q_3 = \begin{bmatrix}
  \tilde{X}_{r_2(1)} \\
  \tilde{Y}_{r_2(2)} \\
  \vdots \\
  \tilde{Y}_{r_2(M)}
\end{bmatrix}_{2M \times 1}
q_4 = \begin{bmatrix}
  \tilde{X}_{r_3(1)} \\
  \tilde{Y}_{r_3(2)} \\
  \vdots \\
  \tilde{Y}_{r_3(M)}
\end{bmatrix}_{2M \times 1}
\] (4)

Therefore, the points \{Path_1, Path_2, \ldots, Path_N\} in \( \mathbb{R}^{2M} \) belongs to the 4-dimensional subspace of \( \mathbb{R}^{2M} \) spanned by the vectors \{q_1, q_2, q_3, q_4\}. In fact, if the motion is translational in \( XY \) plane and rotational in \( Z \) axis, then \( r_3(k) = [0 \ 0 \ 1]^T \) and \( \tilde{r}_3(k) = [0 \ 0]^T \) and therefore \( q_4 = 0 \). Hence, \{Path_1, Path_2, \ldots, Path_N\} lies in a 3-dimensional subspace spanned by \{q_1, q_2, q_3\}. A similar concept is described in [29]. Another proof of this concept from “motion and shape matrix factorization” perspective is given in [26].

REFERENCES

Fig. 7
AFFINE PROJECTION OF PATH.