Int J Comput Vis (2008) 79: 85–105
DOI 10.1007/s11263-007-0099-z

Multiframe Motion Segmentation with Missing Data Using PowerFactorization and GPCA

René Vidal · Roberto Tron · Richard Hartley

Received: 21 November 2006 / Accepted: 26 September 2007 / Published online: 27 November 2007

Abstract We consider the problem of segmenting multiple rigid-body motions from point correspondences in multiple affine views. We cast this problem as a subspace clustering problem in which point trajectories associated with each motion live in a linear subspace of dimension two, three or four. Our algorithm involves projecting all point trajectories onto a 5-dimensional subspace using the SVD, the PowerFactorization method, or RANSAC, and fitting multiple linear subspaces representing different rigid-body motions to the points in $\mathbb{R}^5$ using GPCA. Unlike previous work, our approach does not restrict the motion subspaces to be four-dimensional and independent. Instead, it deals gracefully with all the spectrum of possible affine motions: from two-dimensional and partially dependent to four-dimensional and fully independent. Our algorithm can handle the case of missing data, meaning that point tracks do not have to be visible in all images, by using the PowerFactorization method to project the data. In addition, our method can handle outlying trajectories by using RANSAC to perform the projection. We compare our approach to other methods on a database of 167 motion sequences with full motions, independent motions, degenerate motions, partially dependent motions, missing data, outliers, etc. On motion sequences with complete data our method achieves a misclassification error of less that 5% for two motions and 29% for three motions.

Keywords Multibody factorization · Multibody grouping · 3-D motion segmentation · Structure from motion · Subspace clustering · PowerFactorization and Generalized Principal Component Analysis (GPCA) · Missing data

R. Vidal (✉) · R. Tron
Center for Imaging Science, Department of Biomedical Engineering, Johns Hopkins University, 302B Clark Hall, 3400 N. Charles St., Baltimore, MD 21218, USA
e-mail: rvidal@cis.jhu.edu

R. Tron
e-mail: tron@cis.jhu.edu

R. Hartley
Vision Science Technology and Applications Program, National ICT Australia, Department of Information Engineering, Australian National University, Canberra ACT 0200, Australia
e-mail: Richard.Hartley@anu.edu.au

1 Introduction

The past few decades have witnessed significant advances on the understanding of the geometry and reconstruction of *static scenes* observed by a moving camera. More recently, there has been an increasing interest on developing geometrical and statistical models for the understanding of *dynamic scenes*, i.e. scenes in which both the camera and multiple objects move. This is a challenging problem in visual motion analysis, because it requires the simultaneous estimation of an unknown number of motion models, without knowing which measurements correspond to which model.

Motion segmentation from multiple views has been studied mostly in the case of affine cameras, because in this case the trajectories associated with each motion live in a linear subspace of dimension four or less (Boult and Brown 1991; Tomasi and Kanade 1992) (see Sect. 2.1). This subspace constraint was used by Boult and Brown (1991) to propose a multiframe 3-D motion segmentation algorithm based on thresholding the leading singular vector of the matrix of point trajectories $\mathtt{W}$. Costeira and Kanade (CK) (Costeira and Kanade 1998) extended this approach by thresholding the entries of the so-called *shape interaction* matrix $\mathtt{Q}$. This matrix is built from the singular value decomposition (SVD)

of $\mathtt{W}$ and has the property that $\mathtt{Q}_{ij} = 0$ when points i and j correspond to independent motions, as we will see in Sect. 2.3.

With noisy data the equation $\mathtt{Q}_{ij} = 0$ holds only approximately. In this case, CK's algorithm obtains the segmentation by maximizing the sum of squared entries of $\mathtt{Q}$ in different groups. However, this thresholding process is very sensitive to noise (Gear 1998; Kanatani 2001; Wu et al. 2001). Wu et al. (2001) reduce the effect of noise by building a similarity matrix from the distances among the subspaces obtained by CK's algorithm. Kanatani scales the entries of $\mathtt{Q}$ using the geometric Akaike's information criterion for linear (Kanatani 2001) and affine (Kanatani and Matsunaga 2002) subspaces. Ichimura (1999) finds the groups by thresholding the most discriminant rows of $\mathtt{Q}$. Gear (1998) uses bipartite graph matching to threshold the entries of the row echelon canonical form of $\mathtt{W}$. Gruber and Weiss (2004) formulate the problem as a mixture of factor analysis problem and derive an expectation-maximization (EM) based maximum-likelihood algorithm that also deals with missing data.

Another disadvantage of CK's method and its extensions is that the equation $\mathtt{Q}_{ij} = 0$ holds only when the motion subspaces are linearly independent (Kanatani 2001). That is, the algorithm is not provably correct for most practical motion sequences which usually exhibit partially dependent motions, such as when two objects have the same rotational but different translational motion relative to the camera (Sugaya and Kanatani 2004), or for articulated motions (Yan and Pollefeys 2005), as we will see in Sect. 2.2. This problem motivated the work of Zelnik-Manor and Irani (2003) who use the singular vectors of a normalized shape interaction matrix to build a similarity matrix from which the clustering of the features is obtained using spectral clustering techniques (see Weiss 1999 and references therein). This solution is based on the expectation that on the average the angular displacement of trajectories corresponding to the same motion is smaller than the one of trajectories corresponding to different motions. This assumption is, however, not provably correct. Sugaya and Kanatani (2004) has also studied the case of partially dependent (degenerate) motions under the assumption that the type of degeneracy is known, e.g., 2-D similarity motion or pure translation. Once an initial clustering of the correspondences is obtained, the motion models are estimated using an iterative process that alternates between feature clustering and motion estimation, similarly to the EM algorithm. Yan and Pollefeys (2006) deal with dependent motions by locally fitting a subspace around each point trajectory. The data are segmented by applying spectral clustering to a similarity matrix built from the subspace angles among pairs of subspaces. Since the subspaces are estimated locally, the method cannot deal with transparent motions. To the best of our knowledge, the work of Fan et al. (2006) is the only one that can deal with partially dependent motions without making strong assumptions. The method associates a subspace descriptor with each point trajectory, and clusters these descriptors using a robust voting scheme. The method, however, cannot deal with missing data.

This paper proposes an approach that works for all the spectrum of affine motions: from two-dimensional and partially dependent to four-dimensional and fully independent. This is achieved by a combination of SVD or PowerFactorization or RANSAC, and Generalized Principal Component Analysis (GPCA) that leads to the following purely geometric solution to the multiframe 3-D motion segmentation problem:

1. Project the trajectories onto a five-dimensional subspace using the SVD (complete data), the PowerFactorization method (missing data), or RANSAC (data with outliers).
2. Fit a collection of subspaces to the projected trajectories using Spectral GPCA. Specifically:
   (a) Fit a homogeneous polynomial representing all motion subspaces to the projected data.
   (b) Obtain a basis for each motion subspace from the derivatives of this polynomial.
   (c) Apply spectral clustering to a similarity built from the subspace angles to cluster the data.

We test our approach on a database of 167 motion sequences with full motions, independent motions, degenerate motions, dependent motions, missing data, outliers, etc. On motion sequences with complete data our algorithm achieves a misclassification error of less than 5% for two motions and 29% for three motions.

2 Multiframe Motion Segmentation Problem

In this section, we review the geometry of the 3-D motion segmentation problem from multiple affine views. We first show that the problem is equivalent to clustering multiple low-dimensional linear subspaces of a high-dimensional space. We then review Costeira and Kanade's multibody factorization algorithm (Costeira and Kanade 1998) for independent motion subspaces, and show that it fails when the motion subspaces are not independent.

2.1 Motion Subspace of a Single Rigid-Body Motion

Let $\{\boldsymbol{x}_{fp} \in \mathbb{R}^2\}_{p=1,\ldots,P}^{f=1,\ldots,F}$ be the projections in F frames of P 3-D points $\{\boldsymbol{X}_p \in \mathbb{P}^3\}_{p=1}^{P}$ lying on an object moving rigidly with respect to a rigidly moving camera. Here and henceforth, $\boldsymbol{X}_p$ denotes the homogeneous coordinate representation of the point as a 4-vector with final coordinate equal to 1. On the other hand, $\boldsymbol{x}_{fp}$ is a 2-vector representing the point in non-homogeneous coordinates.

Since only the motion of the object relative to the camera is important, we may assume for simplicity that the camera is stationary and develop our theory based on this assumption. From the evidence of image measurements alone, it is impossible to distinguish this situation from one in which the camera itself is moving.

Under the affine projection model, which generalizes orthographic, weak perspective, and paraperspective projection (Poelman and Kanade 1997), the images satisfy the equation

$$\boldsymbol{x}_{fp} = \mathtt{A}_f \boldsymbol{X}_p, \tag{1}$$

where

$$\mathtt{A}_f = \mathtt{K}_f \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R}_f & \boldsymbol{t}_f \\ \boldsymbol{0}^\top & 1 \end{bmatrix} \tag{2}$$

is the *affine camera matrix* at frame f, which depends on the camera calibration parameters $\mathtt{K}_f \in \mathbb{R}^{2\times 3}$ and the object pose relative to the camera $(\mathtt{R}_f, \boldsymbol{t}_f) \in SE(3)$. Note also that the rows of each $\mathtt{A}_f$ involve linear combinations of the first two rows of the rotation matrix $\mathtt{R}_f$, hence $\mathtt{A}_f$ is a 2×4 matrix of rank 2.

Let $\mathtt{W}_1 \in \mathbb{R}^{2F\times P}$ be the matrix whose P columns are the image point trajectories $\{\boldsymbol{x}_{fp}\}_{p=1}^P$. We call the span of the columns of $\mathtt{W}_1$ the *motion subspace* for the rigid-body motion. It is a fundamental observation that this motion subspace is of dimension no greater than 4. To see this, notice from (1) that $\mathtt{W}_1$ can be decomposed into a *motion matrix* $\mathtt{M}_1 \in \mathbb{R}^{2F\times 4}$ and a *structure matrix* $\mathtt{S}_1 \in \mathbb{R}^{P\times 4}$ as

$$\mathtt{W}_1 = \mathtt{M}_1 \mathtt{S}_1^\top, \tag{3}$$

$$\begin{bmatrix} \boldsymbol{x}_{11} \cdots \boldsymbol{x}_{1P} \\ \vdots \quad\quad \vdots \\ \boldsymbol{x}_{F1} \cdots \boldsymbol{x}_{FP} \end{bmatrix}_{2F\times P} = \begin{bmatrix} \mathtt{A}_1 \\ \vdots \\ \mathtt{A}_F \end{bmatrix}_{2F\times 4} \begin{bmatrix} \boldsymbol{X}_1 \cdots \boldsymbol{X}_P \end{bmatrix}_{4\times P}.$$

Therefore, $\operatorname{rank}(\mathtt{W}_1) \le 4$ and $\operatorname{rank}(\mathtt{W}_1) \ge \operatorname{rank}(\mathtt{A}_f) = 2$, and so there are three possible values for $\operatorname{rank}(\mathtt{W}_1)$.

1. The case $\operatorname{rank}(\mathtt{W}_1) = 2$ occurs when the 3-D points lie in a line, because $\operatorname{rank}(\mathtt{S}_1) = 2$.
2. The case $\operatorname{rank}(\mathtt{W}_1) = 3$ occurs when the 3-D points lie in a plane, because $\operatorname{rank}(\mathtt{S}_1) = 3$, or with a purely rotating camera, with $\mathtt{K}_f = [\mathtt{I}\ \boldsymbol{0}]$ and $\boldsymbol{t}_f = \boldsymbol{0}$, because the last column of $\mathtt{M}_1$ is zero. It also occurs when all the rotation matrices have the same third row, namely $[0\ 0\ 1]\mathtt{R}_f = \boldsymbol{r}$, because then $[\boldsymbol{r}\ 0]^\top$ is in the (right) null space of each $\mathtt{A}_f$ and hence of $\mathtt{M}_1$. This occurs, for example with a purely translating object or camera, for which $\mathtt{R}_f = \mathtt{I}$; more generally it occurs for an object translating and rotating with respect to a fixed camera, where the rotational part of the motion is about the projection direction of the camera.
3. The case $\operatorname{rank}(\mathtt{W}_1) = 4$ is the generic case, and requires the 3-D points to be in general position in $\mathbb{R}^3$ and the motion of the object relative to the camera to be arbitrary.

In summary, under the affine projection model, the 2-D trajectories of 3-D points on a rigidly moving object (the columns of $\mathtt{W}_1$) live in a subspace of $\mathbb{R}^{2F}$ of dimension $d_1 = \operatorname{rank}(\mathtt{W}_1) = 2$, 3 or 4. When $d_1 = 4$, one can use the SVD to factor $\mathtt{W}_1$ as $\mathtt{W}_1 = \hat{\mathtt{M}}_1 \hat{\mathtt{S}}_1^\top$, where both $\hat{\mathtt{M}}_1$ and $\hat{\mathtt{S}}_1$ have 4 columns. For orthographic cameras, one can compute $\mathtt{M}_1$ and $\mathtt{S}_1$ linearly from $\hat{\mathtt{M}}_1$ and $\hat{\mathtt{S}}_1$ (Tomasi and Kanade 1992). Extensions to weak perspective and paraperspective cameras can be found in (Poelman and Kanade 1997). When $d_1 < 4$, $\mathtt{W}_1$ still factors as $\mathtt{W}_1 = \hat{\mathtt{M}}_1 \hat{\mathtt{S}}_1^\top$, but $\hat{\mathtt{M}}_1$ and $\hat{\mathtt{S}}_1$ have less than 4 columns, hence $\mathtt{M}_1$ and $\mathtt{S}_1$ cannot be obtained directly from $\hat{\mathtt{M}}_1$ and $\hat{\mathtt{S}}_1$. Specific methods for $d_1 = 3$ and $d_1 = 2$ can be found in (Vidal and Oliensis 2002) and (Oliensis and Genc 2001), respectively.

As we will see in short, a precise knowledge of the motion and structure matrices $\mathtt{M}_1$ and $\mathtt{S}_1$ in (3) is not necessary for motion segmentation purposes. All what matters is that the measurement matrix $\mathtt{W}_1$ is of rank $d_1 = 2$, 3, or 4, thus it factors as $\mathtt{W}_1 = \hat{\mathtt{M}}_1 \hat{\mathtt{S}}_1^\top$, where $\hat{\mathtt{M}}_1$ and $\hat{\mathtt{S}}_1$ have d_1 columns.

2.2 Segmentation of Multiple Rigid-Body Motions

Assume now that the P trajectories $\{\boldsymbol{x}_{fp}\}_{p=1}^P$ correspond to n objects undergoing n different rigid-body motions relative to the camera. The *3-D motion segmentation problem* is the task of clustering these P trajectories according to the n moving objects. Since the trajectories associated with each object span a linear subspace of $\mathbb{R}^{2F}$ of dimension d_i, $i = 1, \ldots, n$, the 3-D motion segmentation problem is equivalent to clustering a set of points into n subspaces of $\mathbb{R}^{2F}$ of unknown dimensions $d_i \in \{2, 3, 4\}$, where $i = 1, \ldots, n$.

We quickly give the notation relevant to this n-body segmentation problem. As discussed earlier, it is most convenient (though not essential) to think of the camera as defining the world coordinate system and express the motion of the rigid bodies relative to the position of the camera. The camera can then be represented by a projection matrix of the form

$$\mathtt{K}_f \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

The ith rigid body motion is represented by transformation

$$\begin{bmatrix} \mathtt{R}_{if} & \boldsymbol{t}_{if} \\ \boldsymbol{0}^\top & 1 \end{bmatrix} \tag{5}$$

which defines the position of the ith body during frame f. The 2×4 projection matrix is then given by

$$\mathtt{A}_{if} = \mathtt{K}_f \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R}_{if} & \boldsymbol{t}_{if} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{6}$$

A point $\boldsymbol{X}_p$ belonging to the ith rigid motion is projected to the point $\boldsymbol{x}_{fp} = \mathtt{A}_{if} \boldsymbol{X}_p$.

Now, the data matrix consisting of the trajectories of all points can be written as

$$\mathtt{W} = \left[\mathtt{W}_1, \mathtt{W}_2, \ldots, \mathtt{W}_n\right] \Gamma \in \mathbb{R}^{2F \times P}, \tag{7}$$

where the columns of $\mathtt{W}_i \in \mathbb{R}^{2F \times P_i}$ are the trajectories of the P_i points belonging to the ith moving object, $P = \sum_{i=1}^n P_i$, and $\Gamma \in \mathbb{R}^{P \times P}$ is an unknown permutation matrix that specifies the segmentation of the points into separately moving objects. As in the single motion case, the ith data matrix $\mathtt{W}_i$ can be factorized as

$$\mathtt{W}_i = \hat{\mathtt{M}}_i \hat{\mathtt{S}}_i^\top, \quad i = 1, \ldots, n, \tag{8}$$

where $\hat{\mathtt{M}}_i \in \mathbb{R}^{2F \times d_i}$ and $\hat{\mathtt{S}}_i \in \mathbb{R}^{P_i \times d_i}$. Consequently, the data matrix associated with all the objects can be factorized as

$$\begin{aligned} \mathtt{W} &= [\mathtt{W}_1\ \mathtt{W}_2\ \ldots\ \mathtt{W}_n]\ \Gamma \\ &= \left[\hat{\mathtt{M}}_1, \hat{\mathtt{M}}_2, \ldots, \hat{\mathtt{M}}_n\right] \begin{bmatrix} \hat{\mathtt{S}}_1^\top & & & \\ & \hat{\mathtt{S}}_2^\top & & \\ & & \ddots & \\ & & & \hat{\mathtt{S}}_n^\top \end{bmatrix} \Gamma \\ &= \mathtt{M}\mathtt{S}^\top \Gamma, \end{aligned} \tag{9}$$

where $\mathtt{M} \in \mathbb{R}^{2F \times \sum_{i=1}^n d_i}$ and $\mathtt{S} \in \mathbb{R}^{P \times \sum_{i=1}^n d_i}$.

It follows that one possible way of solving the motion segmentation problem is to find a permutation matrix Γ such that $\mathtt{W}\Gamma^\top = \mathtt{M}\mathtt{S}^\top$ can be decomposed into a matrix $\mathtt{M}$ and a *block diagonal* matrix $\mathtt{S}$. This idea is the basis for Gear's algorithm (Gear 1998) and also for CK's algorithm (Costeira and Kanade 1998), which find the permutation matrix from the row echelon canonical form of $\mathtt{W}$ and from the SVD of $\mathtt{W}$, respectively.

However, the motion subspaces need not satisfy the constraint $\operatorname{rank}(\mathtt{W}) = \sum_{i=1}^n d_i$, thus we may not be able to factorize $\mathtt{W}$ as in (9). Trivially, if one subspace is contained in another, then the two subspaces cannot be separated. In general the factorizability of $\mathtt{W}$ according to (9) depends on how the individual motion subspaces are arranged relative to each other. The following subsections study the motion segmentation problem under the assumption that the subspaces are *independent* (see Definition 1) or *partially dependent* (see Definition 2).

Definition 1 (Independent Subspaces) *A set of n linear subspaces $\{\mathcal{W}_i \subset \mathbb{R}^{2F}\}_{i=1}^n$ are said to be independent if for all $i \neq j = 1, \ldots, n$*

$$\dim(\mathcal{W}_i \cap \mathcal{W}_j) = 0. \tag{10}$$

As a consequence, $\dim(\bigoplus_{i=1}^n \mathcal{W}_i) = \sum_{i=1}^i d_i$, where $d_i = \dim(\mathcal{W}_i)$ and $\oplus$ is the direct sum operator.

Definition 2 (Partially Dependent Subspaces) *A set of n linear subspaces $\{\mathcal{W}_i \subset \mathbb{R}^{2F}\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$ are said to be partially dependent if there exist $i, j \in \{1, \ldots, n\}$ such that*

$$0 < \dim(\mathcal{W}_i \cap \mathcal{W}_j) < \min\{d_i, d_j\}. \tag{11}$$

2.3 Segmentation of Independent Motions

Let D be the rank of the data matrix $\mathtt{W}$. We denote the motion subspace corresponding to the ith motion by $\mathcal{W}_i$. Recall that this is the vector space spanned by the columns of the matrix $\mathtt{W}_i$. If the motion subspaces $\{\mathcal{W}_i\}_{i=1}^n$ are independent, then $D = \sum_{i=1}^n d_i$, where d_i is the rank of $\mathtt{W}_i$. In this case, the matrix $\mathtt{W}$ can be factorized as in (9) and the motion segmentation problem can be solved by looking at the entries of the so-called *shape interaction* matrix (Costeira and Kanade 1998), which we will now define.

Let $\mathtt{W} = \mathtt{U}\Sigma\mathtt{V}^\top$ be the singular value decomposition of the data matrix. Since $\mathtt{W}$ has rank D, the matrix Σ has D non-zero diagonal entries. We assume that these diagonal entries are ordered with the non-zero entries coming first. The matrix $\mathtt{V}$ may then be subdivided as $\mathtt{V} = [\mathtt{V}_1\ \mathtt{V}_2]$ where $\mathtt{V}_1$ consists of the first D columns of $\mathtt{V}$ and $\mathtt{V}_2$ consists of the remaining columns. Thus, the rows of the matrix $\mathtt{V}_1^\top$ form an orthonormal basis for the row space of $\mathtt{W}$ and the columns of $\mathtt{V}_2$ form an orthonormal basis for the right null space of $\mathtt{W}$. The shape interaction matrix is now defined to be

$$\mathtt{Q} = \mathtt{V}_1\mathtt{V}_1^\top. \tag{12}$$

Note that $\mathtt{Q}$ has dimension $P \times P$, where P is the total number of point trajectories. As proved in (Kanatani 2001) this matrix has the following property

$$\mathtt{Q}_{ij} = 0 \quad \text{if points } i \text{ and } j \text{ correspond to different motions.} \tag{13}$$

To see this, let $\mathtt{N}_i \in \mathbb{R}^{P_i \times (P_i - d_i)}$ be a matrix whose columns form an orthonormal basis for the null space of $\mathtt{W}_i$, that is

$\mathtt{W}_i\mathtt{N}_i = 0$ and $\mathtt{N}_i^\top \mathtt{N}_i = \mathtt{I}$. Consider now the matrix

$$\mathtt{N} = \Gamma^\top \begin{bmatrix} \mathtt{N}_1 & 0 & \cdots & 0 \\ 0 & \mathtt{N}_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathtt{N}_n \end{bmatrix} \in \mathbb{R}^{P\times(P-D)}. \tag{14}$$

It is clear that the $P - D$ columns of this matrix are orthonormal and lie in the null space of $\mathtt{W}$. On the other hand, the columns of the matrix $\mathtt{V}_2$ defined before form a basis for the right null space of $\mathtt{W}$. Since $\mathtt{V}_2$ and $\mathtt{N}$ have the same dimension, it follows that the columns of $\mathtt{N}$ also form a basis for the right null space of $\mathtt{W}$. Thus, there exists an orthogonal matrix $\mathtt{O}$ such that $\mathtt{V}_2 = \mathtt{N}\mathtt{O}$. Combining this with the fact that $\mathtt{V}\mathtt{V}^\top = \mathtt{I}$ we have

$$\mathtt{V}\mathtt{V}^\top = \mathtt{V}_1\mathtt{V}_1^\top + \mathtt{V}_2\mathtt{V}_2^\top = \mathtt{Q} + \mathtt{N}\mathtt{N}^\top = \mathtt{I}. \tag{15}$$

From this it follows that

$$\Gamma\mathtt{Q}\Gamma^\top = \mathtt{I} - (\Gamma\mathtt{N})(\Gamma\mathtt{N})^\top, \tag{16}$$

which is block-diagonal. This means that $\mathtt{Q}_{ij} = 0$ if i and j correspond to different motions, as required.

Equation (13) has been used by CK's algorithm and its variations to compute the unknown permutation $\Gamma^\top$ that permutes the columns of $\mathtt{W}$ according to the n motions. However,

1. The construction of $\mathtt{Q}$ requires knowledge of the rank of $\mathtt{W}$. As reported in (Gear 1998), using the wrong rank in CK's algorithm leads to very poor results.
2. Equation (13) is valid only when $\mathtt{W}$ is noise free. While several approaches to thresholding noisy entries of $\mathtt{Q}$ have been proposed (see Sect. 1), none of them seems to be effective in practice.
3. Equation (13) does not hold when the motion subspaces are not independent of each other, as we will show in the next subsection.

2.4 Segmentation of Partially Dependent Motions

In practice, not all video sequences are such that the motion subspaces associated with the different moving objects are independent. We consider cases in which this may happen. In the following examples, we suppose that each individual motion has full rank $d_i = 4$. Therefore, for each $i, j = 1, \ldots, n$, the data matrices $\mathtt{W}_i$ and $\mathtt{W}_j$ factor into motion and structure matrices of the form given in (3). Examples of situations where the independence assumption is violated are now given.

1. *Common translation*: when two objects i and j move with the same translation but different rotation relative to the camera, the fourth columns of their motion matrices $\mathtt{M}_i$ and $\mathtt{M}_j$ are equal. Therefore, $\dim(\mathcal{W}_i \cap \mathcal{W}_j) \geq 1$ and $\operatorname{rank}([\mathtt{W}_i\mathtt{W}_j]) \leq d_i + d_j - 1$.
2. *Common rotation*: when two objects i and j move with the same rotation but different translation relative to the camera, the first three columns of $\mathtt{M}_i$ and $\mathtt{M}_j$ are common. Therefore, $\operatorname{rank}([\mathtt{W}_i\ \mathtt{W}_j]) \leq 5$.
3. *Pure translation*: when objects i and j do not rotate relative to the camera, i.e. $\mathtt{R}_f = \mathtt{I}$, the first two columns of $\mathtt{M}_i$ and $\mathtt{M}_j$ are common and the third columns are zero. Therefore, $\operatorname{rank}(\mathtt{M}_i) \leq 3$, $\operatorname{rank}(\mathtt{M}_j) \leq 3$ and $\operatorname{rank}([\mathtt{W}_i\ \mathtt{W}_j]) = 4$. This particular degeneracy was studied in (Sugaya and Kanatani 2004) under the name of *parallel 2-D plane degeneracy*.
4. *Articulated motions*: the work of (Yan and Pollefeys 2005) studied the motion subspaces of articulated objects consisting of two rigid objects connected by a link. When the link is a joint, the objects have a common translation, and so $\operatorname{rank}([\mathtt{W}_i\mathtt{W}_j]) = d_i + d_j - 1$. When the joint is an axis, then the objects not only have common translation, but also their rotation matrices are related by a rotation about the joint axis. As shown in (Yan and Pollefeys 2005), this causes an additional rank drop, so that $\operatorname{rank}([\mathtt{W}_i\mathtt{W}_j]) = d_i + d_j - 2$.
5. *Small number of frames*: it could be the case that the two motions are fully independent, yet their motion subspaces are partially dependent. For instance, with $n = 2$ four-dimensional motions and $F = 3$ frames we have $\operatorname{rank}(\mathtt{W}) \leq 6 < 4 + 4 = 8$.

Unfortunately, when the motions subspaces are not independent, the result in (13) no longer holds. The main reason for this is that the null space of $\mathtt{W}$ can no longer be decomposed as the direct sum of the null spaces of the individual data matrices $\mathtt{W}_i$ as in (14). This is a consequence of the following facts:

1. When the subspaces are partially dependent, the equation $\mathtt{W} = \mathtt{M}\mathtt{S}^\top\Gamma$ in (9) with $\mathtt{M} \in \mathbb{R}^{2F\times\sum d_i}$ and $\mathtt{S} \in \mathbb{R}^{P\times\sum d_i}$ is still valid. However, given $\mathtt{W}$ we cannot recover the matrices $\mathtt{M}$ and $\mathtt{S}$ in (9) using the SVD or any other matrix factorization technique, because $\operatorname{rank}(\mathtt{W}) = D < \sum d_i$. Instead, we recover factors $\hat{\mathtt{M}} \in \mathbb{R}^{2F\times D}$ and $\hat{\mathtt{S}} \in \mathbb{R}^{P\times D}$, where $\hat{\mathtt{M}}$ and $\hat{\mathtt{S}}$ have fewer columns than $\mathtt{M}$ and $\mathtt{S}$ respectively.
2. We may as usual build the shape interaction matrix $\mathtt{Q}$ from the first D columns of $\mathtt{V}$ in the SVD of $\mathtt{W} = \mathtt{U}\Sigma\mathtt{V}^\top$. However, (13) no longer holds, as we shall see. Let $\mathtt{N}_i \in \mathbb{R}^{P_i\times(P_i-d_i)}$ be a matrix whose columns form an orthonormal basis for the null space of $\mathtt{W}_i$. For simplicity, we consider the case where the first two motion subspaces $\mathcal{W}_1$ and $\mathcal{W}_2$ are partially dependent, but the others are independent. Let $\dim(\mathcal{W}_1 \cap \mathcal{W}_2) = d_{12}$. Then $0 < d_{12} < \min(d_1, d_2)$, and so there exist matrices

$\mathtt{N}_{12}^1 \in \mathbb{R}^{P_1 \times d_{12}}$ and $\mathtt{N}_{12}^2 \in \mathbb{R}^{P_2 \times d_{12}}$ of rank d_{12} such that the columns of

$$\mathtt{N}_{12} = \begin{bmatrix} \mathtt{N}_{12}^1 & \mathtt{N}_1 & 0 \\ \mathtt{N}_{12}^2 & 0 & \mathtt{N}_2 \end{bmatrix}, \tag{17}$$

where $\mathtt{N}_{12} \in \mathbb{R}^{(P_1+P_2)\times(P_1+P_2-d_1-d_2+d_{12})}$, form an orthonormal basis for the null space of $[\mathtt{W}_1\mathtt{W}_2]$. The columns of

$$\mathtt{N} = \Gamma^\top \mathrm{diag}(\mathtt{N}_{12}, \mathtt{N}_3, \ldots, \mathtt{N}_n) \tag{18}$$

therefore form an orthonormal basis for the null space of $\mathtt{W}$. However,

$$\mathtt{N}_{12}\mathtt{N}_{12}^\top = \begin{bmatrix} \mathtt{N}_{12}^1\mathtt{N}_{12}^{1\top} + \mathtt{N}_1\mathtt{N}_1^\top & \mathtt{N}_{12}^1\mathtt{N}_{12}^{2\top} \\ \mathtt{N}_{12}^2\mathtt{N}_{12}^{1\top} & \mathtt{N}_{12}^2\mathtt{N}_{12}^{2\top} + \mathtt{N}_2\mathtt{N}_2^\top \end{bmatrix} \tag{19}$$

is not block diagonal, so neither is $\Gamma\mathtt{N}(\Gamma\mathtt{N})^\top$. In general, whenever motion subspaces $\mathcal{W}_i$ and $\mathcal{W}_j$ are partially dependent, the matrix $\Gamma\mathtt{N}(\Gamma\mathtt{N})^\top$ will contain non-zero fill-in in the off-diagonal blocks belonging to the pair of motions i and j. Referring back to (16) we see that

$$\Gamma\mathtt{Q}\Gamma^\top = \mathtt{I} - (\Gamma\mathtt{N})(\Gamma\mathtt{N})^\top \tag{20}$$

is not block-diagonal and so entries in $\mathtt{Q}$ corresponding to points from different motions will not be zero.

As discussed in Sect. 1, two main approaches have been proposed for dealing with partially dependent motions. The work of (Sugaya and Kanatani 2004) assumes the parallel 2-D plane degeneracy discussed in the previous subsection, and solves the motion segmentation problem for this particular case. The work of (Zelnik-Manor and Irani 2003) applies spectral clustering using the angles between the rows of $\mathtt{V}_1$ as a similarity measure. However, this method is not provably correct for partially dependent motions.

The goal of this paper is to find an algorithm that is provably correct both for independent and partially dependent motions. Our algorithm consists of two steps:

1. Projecting the 2-D point trajectories onto a 5-dimensional subspace of $\mathbb{R}^{2F}$.
2. Fitting a polynomial to the projected trajectories and obtaining a basis for the projected subspaces by applying spectral clustering to a similarity matrix built from the derivatives of the polynomial.

The next two sections describe each one of these two steps in detail.

3 Projection onto a 5-Dimensional Subspace

The first step of our algorithm is to project the point trajectories (columns of $\mathtt{W}$) from $\mathbb{R}^{2F}$ to $\mathbb{R}^5$. At a first sight, it may seem counter-intuitive to perform this projection. For instance, if we have $F = 4$ frames of $n = 2$ independent four-dimensional motions, then we can readily apply CK's algorithm, because we are in a non-degenerate situation. However, if we first project onto $\mathbb{R}^5$, the motions subspaces become partially dependent, because the rank of the projected data matrix is at most $5 < 4 + 4 = 8$.

What is the reason for projecting then? The reason is that the segmentation of data lying in multiple subspaces is preserved by a generic linear projection. For instance, if one is given data lying in two lines in $\mathbb{R}^3$ through the origin, then one can project the lines onto a plane in general position[1] and then cluster the data inside that plane. More generally the principle is (Vidal et al. 2005):

Theorem 1 (Segmentation-Preserving Projections) *If the vectors $\{\boldsymbol{w}_j\}$ lie in n subspaces of dimensions $\{d_i\}_{i=1}^n$ in $\mathbb{R}^D$, and if $\pi_{\mathcal{P}}$ is a linear projection from $\mathbb{R}^D$ onto a subspace $\mathcal{P}$ of dimension D', then the points $\{\pi_{\mathcal{P}}(\boldsymbol{w}_j)\}$ lie in $n' \leq n$ linear subspaces of $\mathcal{P}$ of dimensions $\{d_i' \leq d_i\}_{i=1}^n$. Furthermore, if $D > D' > \max_{i=1,\ldots,n}\{d_i\}$, then there is an open and dense set of projections that preserve the number and dimensions of the subspaces, i.e. $n' = n$ and $d_i' = d_i$ for $i = 1, \ldots, n$.*

The same principle applies to the motion segmentation problem. Since we know that the maximum dimension of each motion subspace is four, then projecting onto a generic five-dimensional subspace preserves the clustering of the motion subspaces. Loosely speaking, in order for two motion subspaces to be distinguishable from each other, it is enough for them to be different along one dimension, i.e. we do not really need them to be different in all four dimensions. It is this key observation the one that enables us to treat *all* partially dependent motions as well as *all* independent motions in the same framework: clustering subspaces of dimension two, three or four living in $\mathbb{R}^5$.

Another advantage of projecting the data onto a 5-dimensional space is that, except for the projection itself, the complexity of the motion segmentation algorithm we are about to present becomes independent of the number of frames. Indeed, our algorithm requires a minimum of only *three* frames for *any* number of independent motions.[2] In addition, the projection step enables us to handle complete data (Sect. 3.1), missing data (Sect. 3.2), and data corrupted with outliers (Sect. 3.3), as we will see in the following three subsections.

[1] Notice that the plane must be in a generic position. For instance, a plane perpendicular to any of the lines or perpendicular to the plane containing the lines would fail.

[2] Previous work required the image points to be visible in $2n$ views for n independent motions, as we will see in the next section.

3.1 Projection Using the SVD for Complete Data

Let us first consider the case in which all the feature points are visible in all frames, so that all the entries of the data matrix $\mathtt{W} \in \mathbb{R}^{2F\times P}$ are known. The goal is to find a matrix $\mathcal{P} \in \mathbb{R}^{5\times 2F}$ that projects the columns of $\mathtt{W}$ onto a 5-dimensional subspace of $\mathbb{R}^{2F}$ and preserves the segmentation of the data.

From Theorem 1, we know that there is an open and dense set of linear projections that preserves the segmentation. Therefore, one possible approach is to chose one or more projections at random. This approach will work well with perfect point correspondences. With noisy data, however, each random projection could result in a different segmentation. In fact, it is clear that some projections are better than others for the purpose of segmentation. For instance, if we are given data lying in two lines in $\mathbb{R}^3$, projecting onto the plane containing the lines would maintain the lines as separated as possible. Unfortunately, for data lying on subspaces in general configuration, there is no general methodology for finding a linear projection that keeps the individual subspaces as separated as possible.

An alternative approach is to choose a projection that minimizes the error between the original and the projected data. The optimal solution is to project onto a dominant eigensubspace, which we can do simply by computing the SVD of $\mathtt{W} = \mathtt{U}\Sigma\mathtt{V}^\top$, and then defining a new data matrix $\hat{\mathtt{W}}$ to consist of the first 5 rows of $\mathtt{V}^\top$ (assuming that the singular values are in descending order on the diagonal of Σ). Unfortunately, the projection given by the SVD may not be segmentation preserving. For instance, if we are given points in $\mathbb{R}^3$ distributed symmetrically along the x, y, and z axes, then the principal directions are the main axes. Thus, if we choose the two principal directions to be the x and y axis, then all points in the z axis are projected to the origin, and the segmentation of the data is not preserved.

In practice, however, degenerate cases in which the SVD-based projection fails to preserve the segmentation are rare. In fact, the SVD-based projection works quite well, as we will see in Sect. 6.

3.2 Projection Using the PowerFactorization Method for Incomplete Data

As an alternative to using the SVD to do the projection, we can use the technique of PowerFactorization (Hartley 2003), which in some cases may be more rapid. In addition, it allows us deal with the case in which some entries of the data matrix $\mathtt{W} \in \mathbb{R}^{2F\times P}$ are missing, a fairly common occurrence in feature tracking due to occlusions or points disappearing from the field of view. In this case, we need a way to project such point trajectories onto $\mathbb{R}^5$ in the same way as with complete trajectories. Clearly this cannot be done using SVD, as detailed in the previous section.

We use a method adapted for incomplete data, based on an analysis of the "power method" for computation of eigenvalues of a matrix. The method known as PowerFactorization gives a rapid method for approximating low rank matrices. The PowerFactorization algorithm is discussed in some detail in (Hartley 2003).

Complete Data Case We begin by describing PowerFactorization in the complete data case. Let $\mathtt{W}$ be a matrix of dimension $N \times P$ that we want to approximate by some matrix of rank r. We start with a random matrix $\mathtt{A}_0$ of dimension $N \times r$, and then carry out the following steps for $k = 1, 2, \ldots$ until convergence of the product $\mathtt{A}_k\mathtt{B}_k^\top$.

1. Let $\mathtt{B}_k = \mathtt{W}^\top\mathtt{A}_{k-1}$.
2. Orthonormalize the columns of $\mathtt{B}_k$ by (for instance) the Gram-Schmidt algorithm. This is sometimes called QR algorithm, since it is equivalent to replacing $\mathtt{B}_k$ by a matrix $\mathtt{B}'_k$ such that $\mathtt{B}_k = \mathtt{B}'_k\mathtt{N}_k$, where $\mathtt{B}'_k$ has orthonormal columns, and $\mathtt{N}_k$ is upper-triangular.
3. Let $\mathtt{A}_k = \mathtt{W}\mathtt{B}'_k$.

It was indicated in (Hartley 2003) that the sequence $\mathtt{A}_k\mathtt{B}_k^\top$ converges rapidly to the rank-r matrix closest to $\mathtt{W}$ in Frobenius norm, provided that $\mathtt{W}$ is close to having rank r. Specifically, the rate of convergence is proportional to $(s_{r+1}/s_r)^k$, where s_i is the ith largest singular value of $\mathtt{W}$. Observe that this algorithm is very simple, requiring only matrix multiplications and Gram-Schmidt normalization.

In the case of motion segmentation, the subject of this paper, we wish to replace $\mathtt{W}$ by a matrix obtained by projecting its columns onto a 5-dimensional subspace. If $\mathtt{A}\mathtt{B}^\top$ is the nearest rank-5 factorization to $\mathtt{W}$, then $\hat{\mathtt{W}} = \mathtt{B}^\top$ is the matrix that we require.

Missing Data Case In the case where some of the entries of $\mathtt{W}$ are not known, we cannot carry out SVD or matrix multiplication either. However, the goal remains the same—to find matrices $\mathtt{A}$ and $\mathtt{B}$ such that $\mathtt{A}\mathtt{B}^\top$ is as close to $\mathtt{W}$ as possible. The measure of closeness is

$$\sum_{(i,j)\in\mathcal{I}} (\mathtt{W}_{ij} - (\mathtt{A}\mathtt{B}^\top)_{ij})^2, \tag{21}$$

where $\mathcal{I}$ is the set of pairs (i, j) for which $\mathtt{W}_{ij}$ is known.

In the case of missing data, PowerFactorization takes a slightly different form. Starting as before with a random matrix $\mathtt{A}_0$, we alternate the following steps until convergence of $\mathtt{A}_k\mathtt{B}_k^\top$.

1. Given $\mathtt{A}_{k-1}$, find the $P \times r$ matrix $\mathtt{B}_k$ that minimizes $\sum_{(i,j)\in\mathcal{I}} |\mathtt{W}_{ij} - (\mathtt{A}_{k-1}\mathtt{B}_k^\top)_{ij}|^2$.
2. Orthonormalize the columns of $\mathtt{B}_k$ by replacing it by a matrix $\mathtt{B}'_k$ such that $\mathtt{B}_k = \mathtt{B}'_k\mathtt{N}_k$, where $\mathtt{B}'_k$ has orthonormal columns, and $\mathtt{N}_k$ is upper-triangular.

3. Given $\mathtt{B}_k$, find the matrix $\mathtt{A}_k$ that minimizes $\sum_{(i,j)\in\mathcal{I}} |\mathtt{W}_{ij} - (\mathtt{A}_k\mathtt{B}_k^\top)_{ij}|^2$.

In this algorithm, the computation of each $\mathtt{B}_k$ and $\mathtt{A}_k$ proceeds just one column at a time, and consists of finding the least-squares solution to a set of linear equations.

It was pointed out in (Hartley 2003) that if $\mathtt{W}$ has no missing entries, then this algorithm gives precisely the same sequence of products as the version of the algorithm involving matrix multiplications. Consequently, it is provably rapidly convergent to the optimal solution. In the case of moderate amounts of missing data, we cannot strongly assert this, though the theoretical result for complete data gives us a strong expectation that this will be so; this expectation is borne out by practical experience. PowerFactorization fails to converge to the global minimum only in rare cases, such as with strongly banded data (that is a long image sequence with only short point-tracks).

Essentially this algorithm alternates between computing $\mathtt{A}_k$ and $\mathtt{B}_k$ using least-squares. Similar (not identical) alternation algorithms have been proposed in (Shum et al. 1995; De la Torre and Black 2001; Buchanan and Fitzgibbon 2000). As mentioned, (Hartley 2003) gives theoretical justification for this algorithm.

3.3 Projection Using RANSAC for Data with Outliers

Feature point trajectories from real sequences are affected not only by noise and missing data, but also by outliers. Outliers occur when feature points in two image frames are incorrectly matched, thus the trajectory passing through these feature points does not belong to any of the motion subspaces. It is well known that the existence of outliers can severely affect the performance of motion segmentation algorithms, particularly when linear least-squares estimation methods are used (Sugaya and Kanatani 2002). Therefore, it is important that outliers be detected before segmentation.

Since the first step of our algorithm is to project the $2F$-dimensional trajectories onto a 5-dimensional subspace, one way to deal with outliers is to perform this projection in a robust fashion. A method that is particularly suited for this task is RANdom SAmple Consensus (RANSAC) (Fischler and Bolles 1981). RANSAC is a statistical method for fitting a model to a cloud of points corrupted with outliers in a statistically robust way. More specifically, if d is the minimum number of points required to fit a model to the data, RANSAC proceeds by

1. Randomly sampling d points from the data.
2. Fitting a model to these d points.
3. Computing the residual of each data point to the current model.
4. Choosing the points whose residual is below a threshold as the inliers.

These four steps are then repeated for another d sample points, until the number of inliers is above a threshold, or enough samples have been drawn. The outputs of the algorithm are the parameters of the model (subspace basis) and the labeling of inliers and outliers.

In our case, we wish to fit a 5-dimensional subspace to the $2F$-dimensional trajectories. Therefore, in each iteration of RANSAC we use the SVD with complete data or PowerFactorization with missing data to find a basis for the subspace spanned by the $d = 5$ randomly chosen points. Then, the residuals are given by the distance of each point to the current subspace. Once the trajectories have been projected onto $\mathbb{R}^5$, we can simply apply GPCA to segment the projected trajectories, as we describe in the next section.

4 Fitting Motion Subspaces Using GPCA

We have reduced the motion segmentation problem to finding a set of linear subspaces in $\mathbb{R}^5$, each of dimension at most 4, which contain the data points (or come close to them). The points in question, $\{\boldsymbol{w}_p\}_{p=1}^P$, are the columns of the projected data matrix $\hat{\mathtt{W}} = \left[\boldsymbol{w}_1 \cdots \boldsymbol{w}_P\right] \in \mathbb{R}^{5\times P}$. We solve this problem by fitting and differentiating polynomials using GPCA (Vidal et al. 2005).

4.1 Multiple Subspaces as Algebraic Varieties

It serves our purposes to fit a slightly more general model to the points, fitting them by an *algebraic variety*. In essence, notice that the n motion subspaces can be represented as the zero-set of m polynomials of degree n in 5 variables, $\{q_{n\ell}(\boldsymbol{w})\}_{\ell=1}^m$, where $\boldsymbol{w} \in \mathbb{R}^5$. That is, a union of linear subspaces forms an algebraic variety in $\mathbb{R}^5$. Our task is to find the algebraic variety that best fits the set of points in $\mathbb{R}^5$. The fact that a union of linear subspaces forms an algebraic variety (in simple terms, can be expressed as the zero set of a collection of polynomials) is simple enough. If all the subspaces are hyperplanes (having dimension 4) in $\mathbb{R}^5$, then a single polynomial of degree n suffices. This is because a single plane is represented by a single linear polynomial, just as a plane in $\mathbb{R}^3$ is represented by a linear polynomial equation $ax + by + cz = 0$. Similarly, a set of n hyperplanes is represented by the product of n linear polynomials, one for each plane, e.g., $(ax + by + cz)(dx + ey + fz) = 0$ for two planes.

A linear subspace of codimension greater than one (that is, dimension less than 4 in $\mathbb{R}^5$) is not represented by a single equation. Instead, it may be expressed as the intersection of a set of hyperplanes. Equivalently, the points on the subspace simultaneously satisfy the equations of each of the intersecting hyperplanes—thus, a subspace of codimension c is the common zero-set of c distinct linear polynomials.

Finally, points on a collection of n linear subspaces will satisfy any polynomial formed as the product of linear polynomials, where each linear polynomial represents a hyperplane plane through one of the subspaces. Thus, the collection of n subspaces forms the zero set of a collection of degree n polynomials, as claimed.

Although by the above argument each of the polynomials factors into linear factors, we will not use this condition, since it is not preserved in the presence of slight perturbations (noise) applied to the points. The polynomials that we find will not exactly factor, hence their zero set (the algebraic variety) will not precisely consist of a set of linear subspaces. We will deal with this difficulty in Sect. 4.3. Before that, we will turn to the question of estimating these polynomials from data.

4.2 Fitting Polynomials to Projected Trajectories

Although in the case of degenerate motions, the data points (namely the columns of $\widehat{\mathbb{W}}$) could lie in subspaces of $\mathbb{R}^5$ of dimension less than 4, for the present we will assume that the subspaces are 4-dimensional. Consequently, we limit the discussion to fitting the points to a codimension-1 variety. This means that we need only find a single polynomial that fits the points. The case of degenerate subspaces will be discussed in Sect. 4.4.

In an abstract context, we are given a set of points $\{\boldsymbol{w}_i\}$ in $\mathbb{R}^5$ and we wish to find a homogeneous polynomial of degree n (in 5 variables) that fits the points. Recall that n is the expected number of independent motions. To consider a more familiar problem, suppose we want to fit a homogeneous polynomial of degree n in three variables to a set of points. For simplicity, let $n = 2$. The general homogeneous polynomial of degree 2 in three variables is $ax^2+by^2+cz^2+dxy+exz+fyz$. This is in fact the equation of a conic in the projective plane $\mathcal{P}^2$. Now, suppose that a point (x_i, y_i, z_i) lies on this curve. We substitute into this polynomial and equate to zero to obtain a linear equation in the six unknowns $a, \ldots, f$. It does not matter that the polynomial is non-linear in the variables x, y and z; it is linear in the coefficients that we need to determine. Given sufficiently many points (in this case 5) on the curve we may solve a set of linear equations to determine the coefficients $a, \ldots, f$. With more than 5 points we find a least-squares fit. This is a common procedure of algebraic fitting of a conic to data points.

The process extends naturally to fitting any number of points in any space $\mathbb{R}^K$ with a curve of degree n. The process consists of generating all the possible $M_n = \binom{n+K-1}{n}$ monomials of degree n in the K entries of $\boldsymbol{w}$, i.e. $w_1^{n_1} w_2^{n_2} \cdots w_K^{n_K}$, which we can stack into a vector $\widetilde{\boldsymbol{w}} \in \mathbb{R}^{M_n}$. A general homogeneous polynomial of degree n is a combination of these monomials with certain coefficients $\boldsymbol{c} \in \mathbb{R}^{M_n}$, i.e. $\sum c_{n_1,\ldots,n_K} w_1^{n_1} w_2^{n_2} \cdots w_K^{n_K} = \boldsymbol{c}^\top \widetilde{\boldsymbol{w}}$. Each point required to satisfy the polynomial will lead to a linear equation in $\boldsymbol{c}$, and sufficiently many points will allow us to determine $\boldsymbol{c}$ as the least-squares solution of

$$\widetilde{\mathbb{W}}^\top \boldsymbol{c} = 0, \tag{22}$$

where the columns of $\widetilde{\mathbb{W}} = [\widetilde{\boldsymbol{w}}_1 \cdots \widetilde{\boldsymbol{w}}_P] \in \mathbb{R}^{M_n \times P}$ contain all the monomials of degree n generated by the columns of $\hat{\mathbb{W}} = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_P] \in \mathbb{R}^{K \times P}$.

As an alternative to least-squares, one may use robust techniques for estimating the null space of $\tilde{\mathbb{W}}$ in the presence of outlying feature points. In particular, we can compute $\boldsymbol{c}$ by minimizing the cost function

$$\sum_{p=1}^{P} \rho(\widetilde{\boldsymbol{w}}_p{}^\top \boldsymbol{c}, \sigma) \tag{23}$$

subject to $\|\boldsymbol{c}\| = 1$, where $\rho(x) = \frac{x^2}{x^2+\sigma^2}$ is a robust error function and σ is scaling parameter. This minimization problem can be solved using iterative re-weighted least squares, or gradient descent.

4.3 Segmenting Motion Subspaces by Polynomial Differentiation

At this stage, we have a homogeneous degree-n polynomial q in 5 variables fitting a set of points $\{\boldsymbol{w}_i\}$. Ideally, the polynomial q factors into n linear factors, each one corresponding to a hyperplane in $\mathbb{R}^5$. The present task is to partition the points $\{\boldsymbol{w}_i\}$ according to which of these hyperplanes they lie in (or near). With inexact measurements, the polynomial q will not exactly factor into linear factors. However, the variety it defines (its zero-set) will *approximate* a set of codimension-one subspaces, or hyperplanes.

We consider the derivative of q. This is a 5-vector

$$\nabla q(\boldsymbol{w}) = (\partial q/\partial w_1, \ldots, \partial q/\partial w_5), \tag{24}$$

where q is a polynomial in the variables w_k. The key observation is that this derivative, when evaluated at a point $\boldsymbol{w}$ lying in the variety defined by q, yields the normal vector to the variety at that point.

Considering still the ideal case, if two points $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ lie in the same hyperplane, then $\nabla q(\boldsymbol{w}_i)$ and $\nabla q(\boldsymbol{w}_j)$ will be vectors in the same direction, whereas if they lie in different hyperplanes, these direction vectors will be different. In fact, since q is a homogeneous polynomial, the hyperplanes in question will be linear codimension-1 subspaces passing through the origin. Therefore, two such hyperplanes are identical if and only if they have the same unit normal vector. Therefore two data points $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ will lie in the same hyperplane if and only if the angle between $\nabla q(\boldsymbol{w}_i)$ and $\nabla q(\boldsymbol{w}_j)$ is zero (or 180°).

This suggests a procedure for partitioning the points $\boldsymbol{w}_i$. When q does not factor into linear factors, the normal vectors $\nabla q(\boldsymbol{w}_i)$ and $\nabla q(\boldsymbol{w}_j)$ will not be exactly the same. Nevertheless, we may define a similarity measure for two such points, as follows

$$\mathtt{S}_{ij} = \cos^2(\theta_{ij}), \tag{25}$$

where θ_{ij} is the angle between the two vectors $\nabla q(\boldsymbol{w}_i)$ and $\nabla q(\boldsymbol{w}_j)$. Then $\mathtt{S}_{ij} \approx 1$ if points $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ are from the same motion (and so belong to the same hyperplane), whereas $\mathtt{S}_{ij} < 1$ if they belong to different motions. Given the so-defined similarity matrix $\mathtt{S} \in \mathbb{R}^{P\times P}$, one can apply any spectral clustering technique to obtain the segmentation of the feature points, e.g., (Weiss 1999; Ng et al. 2001). Once the features have been clustered, one can estimate the motion and structure of each moving object using the standard factorization approach for affine cameras, e.g., (Tomasi and Kanade 1992). We therefore have the following algorithm (Algorithm 1) for motion estimation and segmentation from multiple affine views.

Algorithm 1 (Multiframe motion segmentation by PowerFactorization and Spectral GPCA)

Given a matrix $\mathtt{W} \in \mathbb{R}^{2F\times P}$ containing P feature points in F frames (possibly with missing data)

1: *Projection*: Project the columns of $\mathtt{W}$ onto a five-dimensional subspace using the SVD (complete data) or PowerFactorization (incomplete data) or RANSAC (data with outliers) to obtain a (complete) data matrix $\hat{\mathtt{W}} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_P] \in \mathbb{R}^{5\times P}$.
2: *Multibody motion estimation via polynomial fitting*: Obtain a polynomial q representing the n motion subspaces by computing its vector of coefficients $\boldsymbol{c} \in \mathbb{R}^{M_n}$ as the singular vector of the embedded data matrix $\widetilde{\mathtt{W}} = [\widetilde{\boldsymbol{w}_1}, \ldots, \widetilde{\boldsymbol{w}_P}] \in \mathbb{R}^{M_n\times P}$ corresponding to its smallest singular value.
3: *Feature clustering via polynomial differentiation*: Cluster the feature points by applying spectral clustering to the similarity matrix $\mathtt{S}_{ij} = \cos^2(\theta_{ij})$, where θ_{ij} is the angle between the vectors $\nabla q(\boldsymbol{w}_i)$ and $\nabla q(\boldsymbol{w}_j)$ for $i, j = 1, \ldots, P$.
4: *Motion Estimation*: apply the standard factorization approach for affine cameras to each one of the n group of features to obtain motion and structure parameters.

4.4 Dealing with Degenerate and Partially Dependent Motions

Since our method is particularly intended to handle degenerate and partially dependent motions, in which the different linear subspaces have smaller dimension than 4, or intersect non-trivially, we need to understand why the proposed method based on fitting 4-dimensional subspaces works in these cases.

In the case of partially dependent (but fully-dimensional) motions, the hyperplane subspaces will have dimension 4 in $\mathbb{R}^5$, as usual. Two such subspaces will have distinct normals, even if they intersect in some non-zero subspace. Since the normals are different, our method will work effectively.

In the case of degenerate motions, some of the subspaces may have smaller dimension than the expected dimension 4. Such a subspace may be defined as the intersection of more than one hyperplane. By choosing a single polynomial q to fit all subspaces, we effectively choose a single one of these hyperplanes containing the low-dimensional subspace.[3] As long as this hyperplane does not correspond with the hyperplane defining one of the other motions, there will be no problem with using such a hyperplane to segment the degenerate motion. Generally this favorable condition will apply.

Modeling Low-Dimension Subspaces Explicitly We refer the reader to (Vidal et al. 2005) for an extension of the aforementioned techniques to modeling low-dimensional subspaces explicitly. To do this, instead of fitting the data to a single polynomial, we need to compute a set of independent polynomials $\{q_k\}$ that fit the points. Derivatives of these polynomials give a set of vectors generating a normal subspace, rather than a single normal. The similarity measure defined in (25) may be generalized to measure the largest principal angle between the normal subspaces corresponding to different points.

Using multiple polynomials to model degenerate subspaces could in principle give better segmentation results than using a single polynomial, because one can avoid choosing a hyperplane containing two motion subspaces. However, an important problem is that we do not know a priori how many polynomials to use, because we do not know a priori if a sequence contains degenerate motions or not. Under ideal noise free conditions the number of polynomials is simply the dimension of the left null space of the matrix $\widetilde{\mathtt{W}}$. With noise and outliers it is easy to overestimate the number of polynomials, which will lead to an erroneous segmentation. Even if we could determine the rank of $\widetilde{\mathtt{W}}$ effectively, recall that the rank also depends on the number of motions. Determining both the number of polynomials and the number of motions from a noisy matrix $\widetilde{\mathtt{W}}$ is a difficult model selection problem.

[3] It was shown in (Vidal et al. 2005) that even though for degenerate motions the polynomial q may not be factorizable, its derivative at a point in one of the motion subspaces still gives a normal vector to that subspace. It is the hyperplane associated with this normal vector the one we are referring to here, which exists for any q, factorizable or not.

5 Benchmark

In order to evaluate the performance of our algorithm on real data, we collected a database consisting of *53 video sequences* of indoor and outdoors scenes containing two or three motions. The trajectories of each video sequence *X* containing three motions were split into three motion sequences *X_g12*, *X_g13* and *X_g23* containing the points from groups one and two, one and three, and two and three, respectively. This gave a total of *167 motion sequences*: 129 with two motions and 38 with three motions. Based on the video content, the sequences can be categorized into four groups:

Checkerboard Sequences This group consists of 104 sequences of indoor scenes taken with a handheld camera under controlled conditions. The checkerboard pattern on the objects is used to assure a large number of tracked points. Most sequences contain full and independent motions. Sequences *1R2RC–2T3RTCR* contain three motions: two objects (identified by the numbers *1* and *2*, or *2* and *3*) and the camera itself (identified by the letter *C*). The type of motion of each object is indicated by a letter: *R* for rotation, *T* for translation and *RT* for both rotation and translation. If there is no letter after the *C*, then the camera is fixed. For example, if a sequence is called *1R2TC* it means that the first object rotates, the second translates and the camera is fixed. Sequence *three-cars* is taken from (Vidal et al. 2006) and contains three motions of two toy cars and a box moving on a plane (the table) taken by a fixed camera.

Traffic Sequences This group consists of 38 sequences of outdoor traffic scenes taken by a moving handheld camera. Sequences *carsX–truckX* have vehicles moving on a street. Sequences *kanatani1* and *kanatani2* are taken from (Sugaya and Kanatani 2004) and display a car moving in a parking lot. Most scenes contain degenerate motions, particularly linear and planar motions.

Articulated/Non-Rigid Sequences This group contains 13 sequences displaying motions constrained by joints, head and face motions, people walking, etc. Sequences *arm* and *articulated* contain checkerboard objects connected by arm articulations and by strings, respectively. Sequences *people1* and *people2* display people walking, thus one of the two motions (the person walking) is partially non-rigid. Sequence *kanatani3* is taken from (Sugaya and Kanatani 2004) and contains a moving camera tracking a person moving his head. Sequences *head* and *two_cranes* are taken from (Yan and Pollefeys 2006) and contain two and three articulated objects, respectively.

Table 1 Database average statistics

	2 Groups			3 Groups		
	# Seq.	Points	Frames	# Seq.	Points	Frames
Checkerboard	78	291	28	26	437	28
Traffic	31	241	30	7	332	31
Articulated	11	155	40	2	122	31
Missing data	9	396	35	3	594	35
All	129	275	30	38	414	29
Point Distrib.		35%–65%			20%–24%–56%	

Missing Data Sequences This group consists of 12 checkerboard sequences, $oc1R2RCT\_X$, $oc1R2RC\_X$ and $oc2R3RCRT\_X$, which contain 4.48% to 35.57% of missing data.

Table 1 reports the number of sequences in each category. It can be seen that most sequences are checkerboard sequences. Table 1 also reports the average number of tracked points and frames for each category. For the sequences taken from (Sugaya and Kanatani 2004; Vidal et al. 2006; Yan and Pollefeys 2006), the point trajectories were provided in the respective datasets. For the remaining sequences, the trajectories were obtained in a semi-automatic manner. First, a tool based on a tracking algorithm implemented in OpenCV (2000) was used to extract feature points in the first frame and track them in the following frames. Then an operator removed obviously wrong trajectories, e.g., points disappearing in the middle of the sequence due to an occlusion by another object. Afterwards, the ground-truth segmentation was obtained by manually assigning each point trajectory to its corresponding cluster. The number of points per sequence ranges from 39 to 556, and the number of frames ranges from 15 to 100. Table 1 also contains the average distribution of points per moving object, with the last group corresponding to the camera motion (motion of the background). This statistic was computed on the original 53 videos only. Notice that typically the number of points tracked in the background is about twice as many as the number of points tracked in a moving object.

Table 2 categorizes the sequences with complete data according to the types of motions present in the scene: full and independent motions, full and partially dependent motions, degenerate and independent motions, or degenerate and partially dependent motions. The types of motions were obtained from the dimensions of the individual motion subspaces as well as the dimensions of their unions. However, determining the dimension of a motion subspace is not straightforward, because it involves computing the rank of a noisy matrix. For this purpose, we adapted the method of Kanatani and Matsunaga (2002), which is based on the minimum description length (MDL). More specifically, for each motion group within each sequence, we computed the rank

Table 2 Number of motion sequences of different types of motion

	2 Groups	3 Groups	Total
Full & Independent	83	17	100
Full & Dependent	31	16	47
Degenerate & Independent	3	0	3
Degenerate & Dependent	3	2	5
Total	120	35	155

r_i of the (noisy) data matrix $\mathtt{W}_i$ in (3) using the method described in Kanatani and Matsunaga (2002) with a minimum rank of $r_{\min} = 2$ and a maximum rank $r_{\max} = 4$. A similar method was then used to compute the ranks $r_{i\cup j}$ of the data matrices associated with the union of two motion groups i and j using the individual ranks r_i and r_j computed in the previous step. In this case the minimum and maximum ranks were chosen as $r_{\min} = \max\{r_i, r_j\} + 1$ and $r_{\max} = r_i + r_j$. It is then possible to compute the dimension of the intersection between the two subspaces as $r_{i\cap j} = r_i + r_j - r_{i\cup j}$. For sequences with three motions, we computed the rank $r_{i\cup j\cup k}$ of the union of three motion subspaces using of the union of three motion subspaces using again the method described in Kanatani and Matsunaga (2002). The minimum and maximum ranks were obtained using the formula

$$r_{i\cup j\cup k} = r_i + r_j + r_k - r_{i\cap j} - r_{i\cap k} - r_{j\cap k} + r_{i\cap j\cap k} \tag{26}$$

where $r_{i\cap j\cap k}$ is constrained to be

$$\begin{aligned} &\max\{(r_{i\cap j} + r_{i\cap k} - r_i), (r_{i\cap j} + r_{j\cap k} - r_j), \\ &\quad (r_{i\cap k} + r_{j\cap k} - r_k), 0\} \\ &\leq r_{i\cap j\cap k} \leq \min\{r_{i\cap j}, r_{i\cap k}, r_{j\cap k}\}. \end{aligned} \tag{27}$$

By looking at the numbers in Table 2, it can be seen that most sequences with complete data contain full and independent motions. This is expected, because most of the sequences are checkerboard sequences with generic motions.

To assess if the motion sequences in the database are well approximated by the affine projection model, a rank-4 approximation $\hat{\mathtt{W}}_i$ of the data matrix $\mathtt{W}_i$ associated with the ith motion group of each video sequence was computed. The squared reprojection error for a video sequence is then given by $\frac{1}{2FPn}\sum_{i=1}^{n} \|\hat{\mathtt{W}}_i - \mathtt{W}_i\|_F^2$. Figure 1 shows the number of sequences achieving a certain reprojection error. It can be seen that more than 95% of the sequences have a reprojection error of less than 1 pixel, showing that the affine model is appropriate for the sequences in the database.

To summarize the amount of motion present in all the sequences, we computed the rotation and translation between all pairs of consecutive frames for each motion in each sequence using the standard factorization method (Tomasi and Kanade 1992). This information was used to produce the

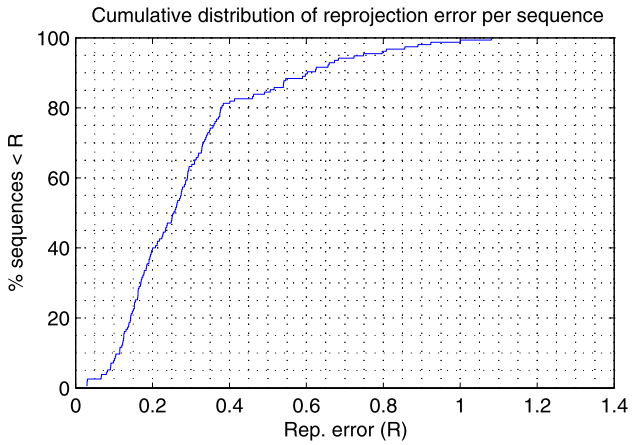
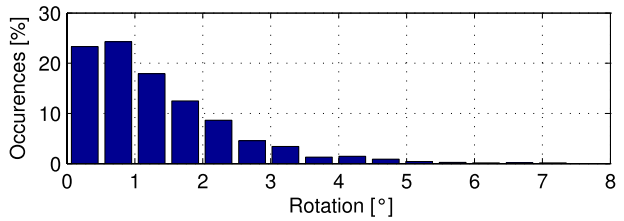
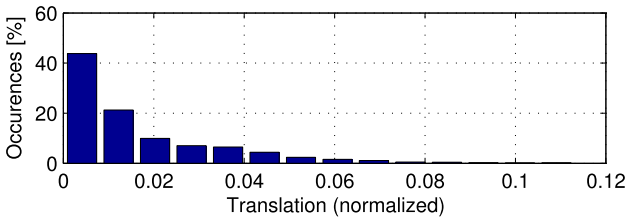


Fig. 1 Histogram of reprojection error (per sequence) for the sequences with complete data



(a) Amount of rotation $\theta = \text{acos}\left((\text{trace}(\mathtt{R}_{f+1}^{\top}\mathtt{R}_f)) - 1)/2\right)$ in degrees.



(b) Amount of translation $\tau = \|\boldsymbol{t}_f\| / \max\{\text{depth}\}$.

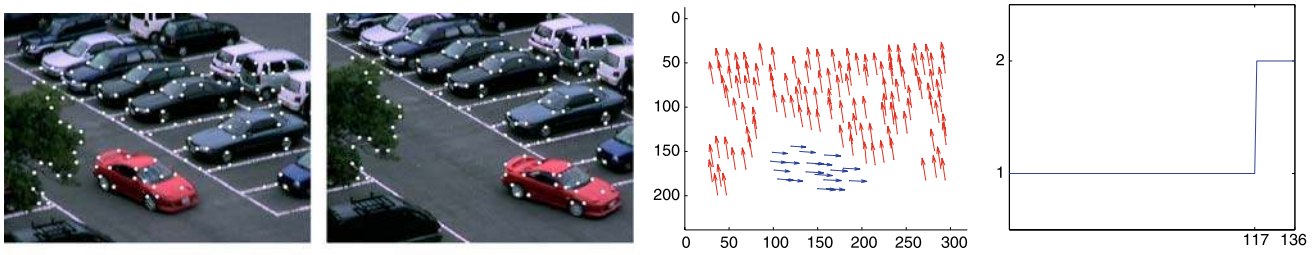
Fig. 2 Histograms with the amount of rotation and translation between two consecutive frames for the sequences with complete data

histograms shown in Fig. 2. Notice that the inter-frame rotation is between 0 and 5 degrees, and the inter-frame translation is between 0 and 0.1.

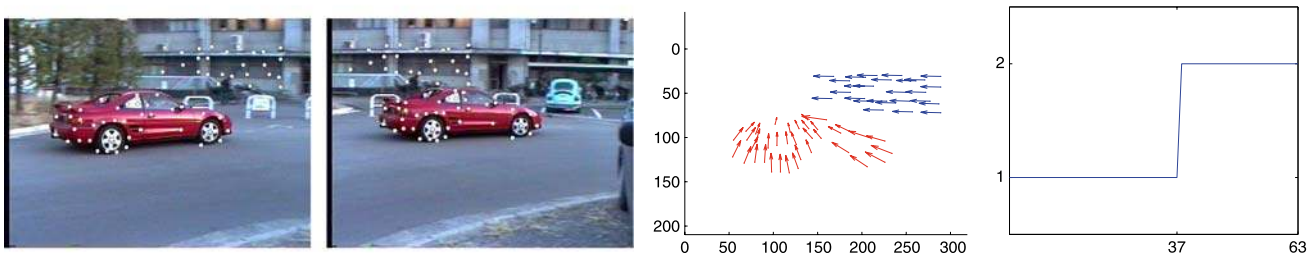
6 Experiments on Real Images

In this section, we compare three variations of our algorithm against several multi-view affine algorithms on our benchmark of 167 motion sequences. More specifically, we compare the following methods:

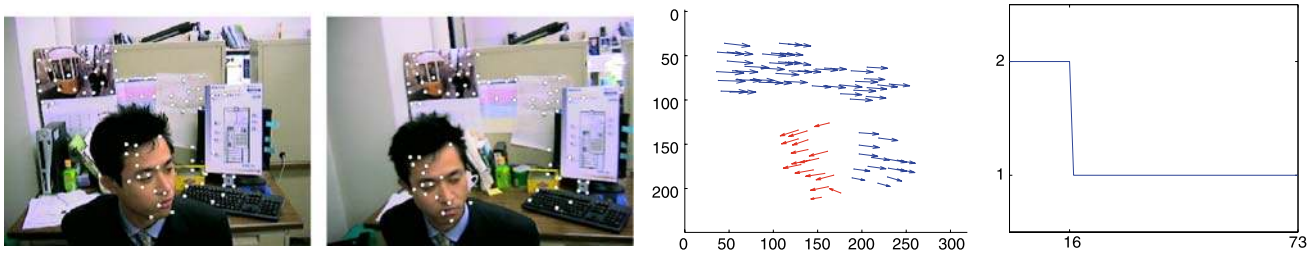
1. **Costeira & Kanade (CK)**: this method (Costeira and Kanade 1998) builds a shape interaction matrix $\mathtt{Q}$ from the SVD of the data matrix. The data is segmented by



(a) *kanatani1* sequence



(b) *kanatani2* sequence



(c) *kanatani3* sequence

Fig. 3 Motion segmentation results for the Kanatani sequences. The *first and second columns* show respectively the first and last frames of the sequence with point correspondences superimposed. The *third column* shows the displacement of the correspondences in pixels between the first and last frames. The *fourth column* shows the segmentation results given by our algorithm: the x-axis is the index for each point and the y-axis is the group 1 or 2

Table 3 Classification errors of various subspace clustering algorithms on the Kanatani sequences (Sugaya and Kanatani 2004)

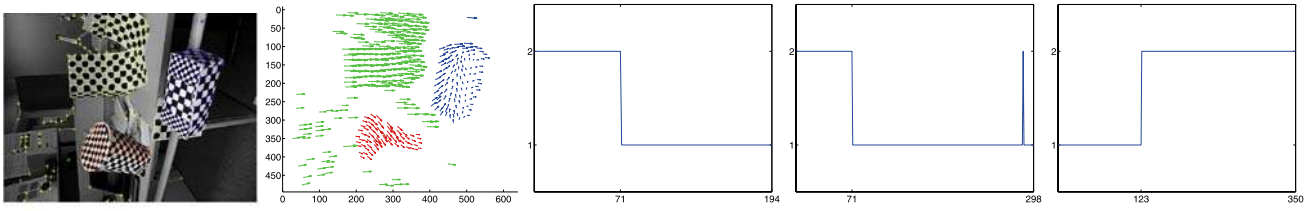
Sequence	*kanatani1*	*kanatani2*	*kanatani3*
# of points P	136	63	73
# of frames F	30	17	100
CK (Costeira and Kanade 1998)	39.7%	28.7%	41.2%
Ichimura (Ichimura 1999)	7.4%	19.9%	31.7%
SS (Kanatani 2001)	70.7%	0.5%	1.1%
ASS (Kanatani and Matsunaga 2002)	18.2%	0.3%	32.5%
MSL (Sugaya and Kanatani 2004)	0.0%	0.0%	0.0%
RANSAC	35.0%	6.1%	17.6%
SVD+GPCA	0.0%	0.0%	0.0%

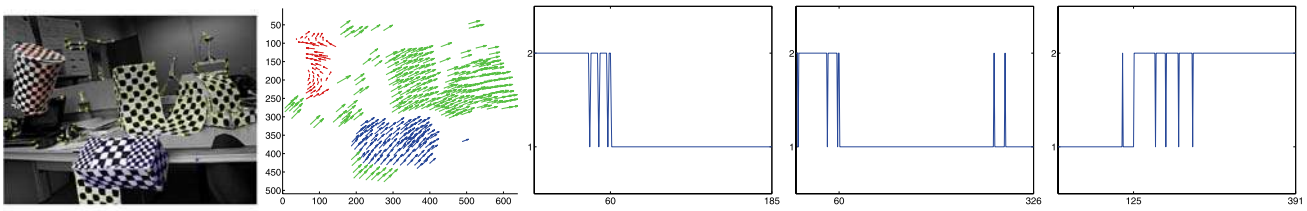
Table 4 Computation times for MSL, RANSAC and SVD+GPCA on the Kanatani sequences

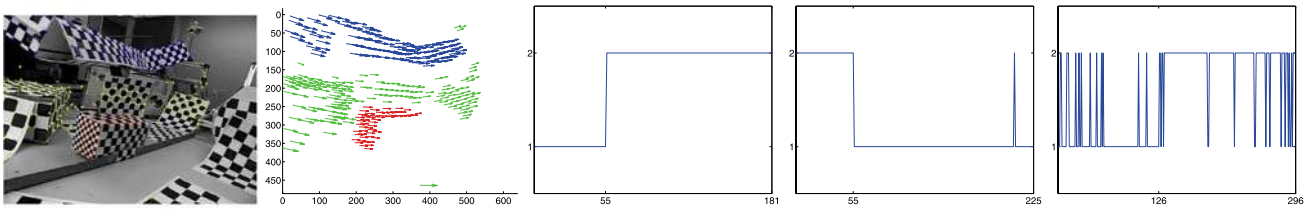
Sequence	MSL	RANSAC	SVD+GPCA
kanatani1	80.76s	0.379s	0.001s
kanatani2	1.99s	0.598s	0.097s
kanatani3	4.16s	0.326s	0.114s

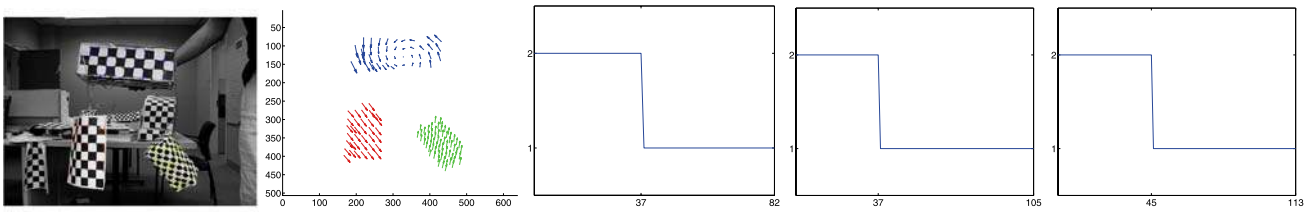
thresholding the entries of $\mathbb{Q}$, which is done by maximizing the sum of squared entries of $\mathbb{Q}$ in different groups.

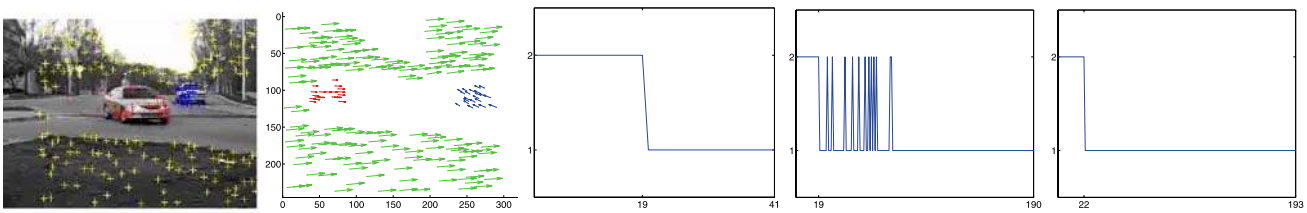
2. **Ichimura**: this method (Ichimura 1999) finds the kth motion subspace by thresholding the kth most discriminant row of $\mathbb{Q}$.
3. **Subspace Separation (SS)**: this method (Kanatani 2001) applies CK's algorithm to a matrix obtained by scaling the entries of $\mathbb{Q}$ using the geometric Akaike's information criteria (GAIC) for linear subspaces.
4. **Affine Subspace Separation (ASS)**: this method (Kanatani and Matsunaga 2002) is the same as SS, except that it weights the entries of $\mathbb{Q}$ using the GAIC for affine subspaces.











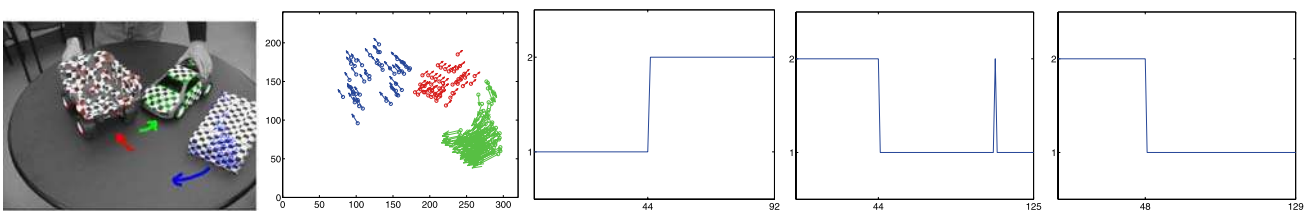
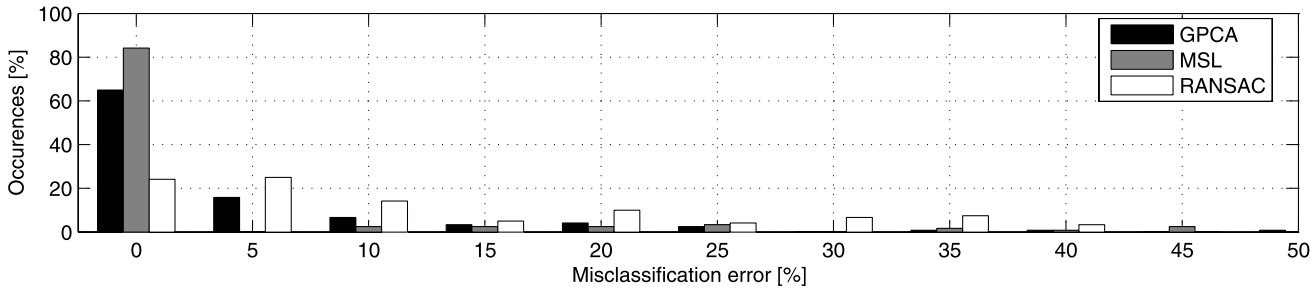
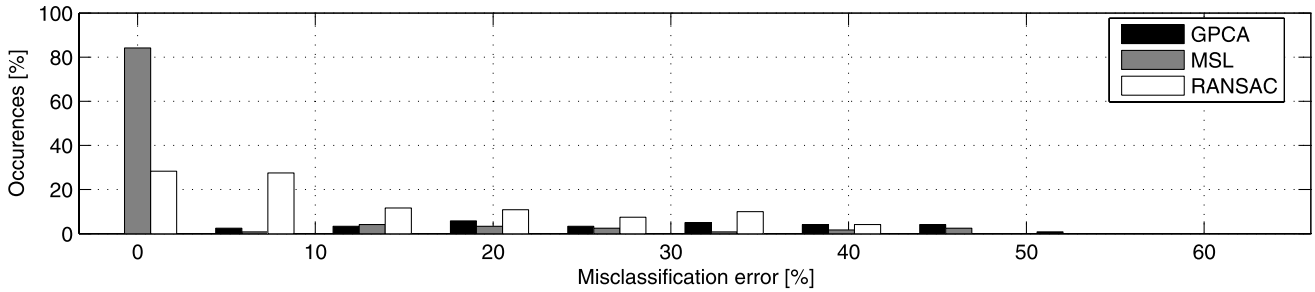


Fig. 4 Motion segmentation results on sequences with *complete data*. The *first column* shows the first frame of each sequence. The *second column* shows the displacement of the correspondences between the first and the last frames. The *third-fifth columns* show the clustering of the correspondences given by our algorithm for groups 1-2, 1-3 and 2-3. The x-axis is the index for each point and the y-axis is the group 1 or 2



(a) Sequences with two motions



(b) Sequences with three motions

Fig. 5 Histograms with the percentage of sequences in which each method achieves a certain classification error

5. **Multistage Stage Learning (MSL)**: this method (Sugaya and Kanatani 2004) obtains an initial segmentation by using the CK's method adapted to the 2-D plane degeneracy. This initial segmentation is then refined using a multi-stage learning technique that applies EM for multiple degenerate subspaces, followed by EM for multiple affine subspaces, followed by EM for multiple linear subspaces.
6. **SVD+GPCA**: This is the algorithm proposed in this paper for complete data with no outliers.
7. **PF+GPCA**: This is the algorithm proposed in this paper for missing data with no outliers.
8. **RANSAC+GPCA**: This is the algorithm proposed in this paper for data with outliers.
9. **RANSAC**: instead of using RANSAC to project the data onto $\mathbb{R}^5$, one can use it to fit multiple subspaces of dimension $d = 4$ directly. This can be done by applying RANSAC recursively, thereby fitting one dominant subspace at a time. The first subspace is fit by applying RANSAC to all the data, with points in other subspaces considered as outliers. The remaining subspaces are obtained in an analogous fashion, after removing the inliers to the previously estimated subspaces.

These algorithms are compared using the following performance measures

$$\text{outlier detection rate} = \frac{\#\text{ of correctly detected outliers}}{\text{total }\#\text{ of outliers}}, \tag{28}$$

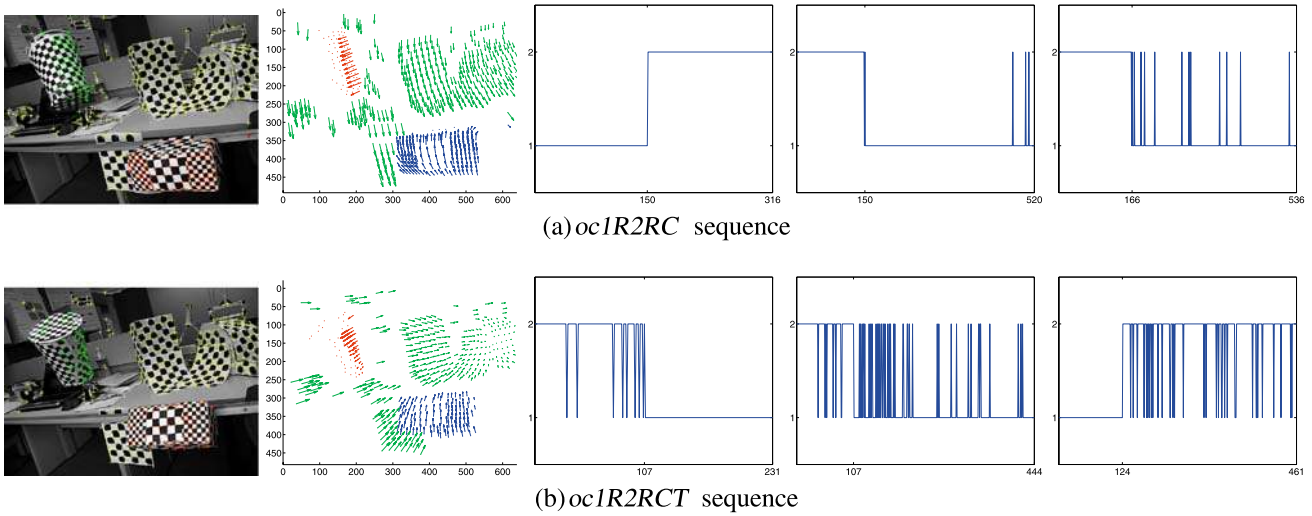
$$\text{classification error} = \frac{\#\text{ of misclassified inliers}}{\text{total }\#\text{ of inliers}}. \tag{29}$$

For data with no outliers, we compute only the classification error, with all point trajectories considered as inliers. In addition, we also we also compare the computation times (CPU times) of a MATLAB® implementation of our algorithm and of RANSAC, and a C++ implementation of MSL. The reference machine used for all the experiments is an Intel Xeon MP with 8 processors at 3.66 GHz and 32 GB of RAM (but for each simulation each algorithm exploits only one processor, without any parallelism).

6.1 Comparison with Multi-frame Algorithms on the Kanatani Sequences

We first test our algorithm on the three Kanatani sequences, which are shown in Fig. 3. The *kanatani1* and *kanatani2* sequences are taken by a moving camera tracking a car moving, respectively, in front of a parking lot and a building. These sequences contain degenerate and partially dependent motions. The *kanatani3* sequence is taken by a moving camera looking at a person moving his head. This sequence contains full and independent motions.

Table 3 compares the segmentation results reported in (Sugaya and Kanatani 2004) for the CK, Ichimura, SS, ASS, and MSL algorithms against the results given by SVD+GPCA and RANSAC. Notice that CK, Ichimura, SS, ASS and RANSAC fail to give the correct segmentation,



(a) *oc1R2RC* sequence

(b) *oc1R2RCT* sequence

Fig. 6 Motion segmentation results on sequences with *missing data*. The *first column* shows the first frame of each sequence. The *second column* shows the displacement of the correspondences between the first and the last frames. The *third-fifth columns* show the clustering of the correspondences given by our algorithm for groups 1-2, 1-3 and 2-3, respectively. The x-axis is the index for each point and the y-axis is the group 1 or 2

while SVD+GPCA and MSL achieve perfect classification for all three sequences. The comparison is somewhat unfair, because CK, Ichimura, SS and ASS cannot handle partially dependent motions, while SVD+GPCA, MSL and RANSAC can. On the other hand, our algorithm is purely algebraic, while the others use iterative refinement to deal with noise. Nevertheless, the only algorithm that has a performance comparable to ours is MSL, which is based on solving a series of EM-like iterative optimization problems, at the expense of a significant increase in computation. As shown in Table 4, SVD+GPCA takes about 0.1 seconds in MATLAB® , while MSL takes from 2 to 80 seconds in C++.

6.2 Comparison with Multi-frame Affine Algorithms on Sequences with Complete Data

We now compare SVD+GPCA against MSL and RANSAC on the sequences with complete data. This database consists of 155 sequences with complete data, which correspond to the checkerboard, traffic and articulated sequences. Figure 4 shows some of the sequences. We compare our method against MSL and RANSAC only, because they can handle degenerate and partially dependent motions.

Tables 5, 6, 7, 8 show statistics with the classification errors and computation times for the different types of sequences. Figure 5 shows histograms with the number of sequences in which each algorithm achieved a certain classification error. More detailed statistics with the classification errors and computation times of each algorithm on each one of the 155 sequences can be found at *http://www.vision.jhu.edu*.

Table 5 Performance statistics for two groups

Checkerboard	SVD+GPCA	MSL	RANSAC
Average	6.09%	4.46%	13.70%
Median	1.03%	0.00%	8.18%
Traffic	SVD+GPCA	MSL	RANSAC
Average	1.41%	2.23%	10.72%
Median	0.00%	0.00%	6.31%
Articulated	SVD+GPCA	MSL	RANSAC
Average	2.88%	7.23%	9.05%
Median	0.00%	0.00%	1.77%
All	SVD+GPCA	MSL	RANSAC
Average	4.59%	4.14%	12.50%
Median	0.38%	0.00%	8.05%

Table 6 Computation times averages for two groups

	SVD+GPCA	MSL	RANSAC
Checkerboard	342ms	7h 4m	158ms
Traffic	286ms	21h 34m	212ms
Articulated	187ms	9h 47m	270ms
All	314ms	11h 4m	182ms

By looking at the results, we can draw the following conclusions about the performance of SVD+GPCA, MSL and RANSAC on sequences with complete data.

Table 7 Performance statistics for three groups

Checkerboard	SVD+GPCA	MSL	RANSAC
Average	31.95%	10.38%	25.77%
Median	32.93%	4.61%	27.49%
Traffic	SVD+GPCA	MSL	RANSAC
Average	19.83%	1.80%	22.64%
Median	19.55%	0.00%	21.11%
Articulated	SVD+GPCA	MSL	RANSAC
Average	16.85%	2.71%	24.36%
Median	16.85%	2.71%	24.36%
All	SVD+GPCA	MSL	RANSAC
Average	28.66%	8.23%	25.07%
Median	28.26%	1.76%	26.30%

Table 8 Computation times averages for three groups

	SVD+GPCA	MSL	RANSAC
Checkerboard	820ms	2d 6h	1.067s
Traffic	539ms	1d 8h	1.295s
Articulated	125ms	1m 19.993s	2.169s
All	724ms	1d 23h	1.176s

SVD+GPCA For SVD+GPCA, we have to comment separately the results for sequences with two and three motions. For two motions, SVD+GPCA has a classification error of 4.59% with an average computation time of 324 ms. The errors are higher on the checkerboard sequences (6.09%), which constitute the majority of the database. However, for the traffic and articulated sequences SVD+GPCA is the most accurate method, with errors of 1.41% and 2.88%, respectively.

For sequences with three motions the results are completely different: the increase of computation time is reasonable (about 738 ms), but the segmentation error is significantly higher (about 29%). This is expected, because of two reasons. First, the number of coefficients fitted by GPCA grows exponentially with the number of motions, while the number of point trajectories stays on the same order of magnitude. This causes the estimation of the coefficients of the polynomial to be less accurate. Second, and most importantly, a linear estimation of the coefficients neglects nonlinear constraints. The larger the number of motion, the more nonlinear constraints are neglected, and so the estimation of the coefficients is less and less accurate.

MSL If we look only at the average classification error, we can see that MSL is the most accurate method. Furthermore, its segmentation results remain consistent when going from two to three motions. However, the MSL method has two major drawbacks. First, the EM algorithm can get stuck in a local minimum. This is reflected by high classification errors for some sequences with small reprojection error. Second, and more importantly, the complexity does not scale favorably with the number of points and frames, as the computation times grow in the order of minutes, hours and days, while for SVD+GPCA the computation times remain in the order of milliseconds. This may prevent the use of the MSL algorithm in practice, even considering its excellent accuracy.

RANSAC The results for this purely statistic algorithm are similar to what we found for SVD+GPCA. On sequences with two motions, RANSAC gives relatively good results, though it is clearly less accurate than GPCA+SVD and MLS. Also, notice that RANSAC has the shortest computation times. For sequences with three motions the accuracy of RANSAC is not satisfactory. This is expected, because as the number of motions increases, the probability of drawing a set of points from the same group reduces significantly. That said, RANSAC scales better than SVD+GPCA when going from two to three motions. In fact, RANSAC outperforms SVD+GPCA in this case. Another drawback of RANSAC is that its performance varies between two runs on the same data. Our experiments present averaged results over 1000 trials.

Overall, we conclude that SVD+GPCA is more accurate on sequences with two degenerate and partially dependent motions, while MSL is more accurate on sequences with full and independent motions.

6.3 Performance of our Algorithm on Sequences with Missing Data

We now test the performance of PF+GPCA on 11 sequences with missing data. Some of the sequences are shown in Fig. 6. The missing data are represented by image points that go out of sight, because of rotating objects, one object occluding another, etc. The percentage of missing data ranges from 4.48% to 35.57% for these sequences.

Tables 9 and 10 show the segmentation results for sequences with two and three motions, respectively. By looking at the results, we can make the following observations.

Sequences oc1R2RCT_X and oc1R2RC_X For these 6 sequences, the amount of missing data varies between 4.48% and 12.56%. PF+GPCA works quite well, giving a perfect segmentation for one of the sequences and a maximum classification error of 5.21% for the remaining sequences.

Table 9 Classification errors, percentage of missing data and number of iterations for PF for sequences with missing data (2 groups)

Sequence	P	F	n	% M. D.	PF+GPCA	# it.
oc1R2RCT_g12	231	30	2	10.13%	4.33%	10
oc1R2RCT_g13	444	30	2	9.04%	3.15%	16
oc1R2RCT_g23	461	30	2	4.83%	5.21%	24
oc1R2RC_g12	316	40	2	12.56%	0.00%	14
oc1R2RC_g13	520	40	2	11.46%	1.92%	26
oc1R2RC_g23	536	40	2	4.48%	3.17%	14
oc2R3RCRT_g12	192	35	2	34.36%	43.75%	50
oc2R3RCRT_g13	447	35	2	21.02%	13.42%	40
oc2R3RCRT_g23	417	35	2	20.24%	14.63%	23
Average	396	35	2	14.23%	9.95%	24

Table 10 Classification errors, percentage of missing data and number of iterations for PF for sequences with missing data (3 groups)

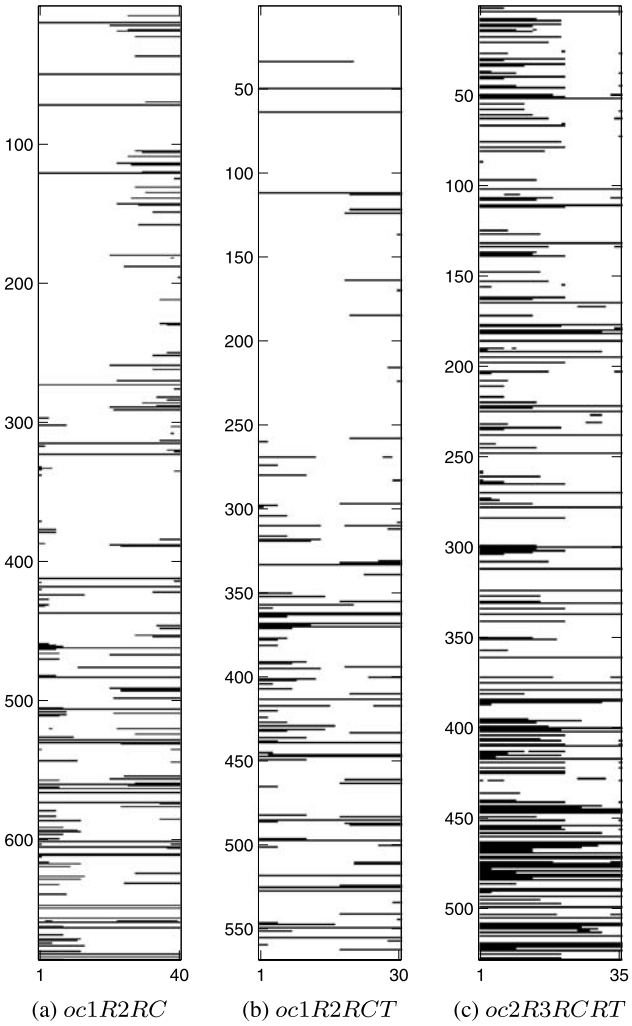
Sequence	P	F	n	% M. D.	PF+GPCA	# it.
oc1R2RC	686	40	3	8.98%	35.57 %	12
oc1R2RCT	568	30	3	7.55%	28.17 %	17
oc2R3RCRT	528	35	3	23.14%	25.95 %	30
Average	594	35	3	13.22%	29.89 %	20

Sequences oc2R3RCRT_g12, oc2R3RCRT_g13 and oc2R3RCRT_g23 For these sequences the amount of missing data is relevant (above 20%) and PF+GPCA is not able to give a good segmentation. This is because when the missing data are structured PF is more likely to converge to a local minimum. In this case, most of the missing data are concentrated in trajectories 440 to 550, as can be seen in Fig. 7. Notice also that in general the number of iterations required for convergence of PF is larger. In fact, the maximum of 50 iterations is achieved for one of the sequences.

Sequences with Three Motions For these sequences, PF+GPCA does not give good results, similarly to what we have already seen in the case of complete data.

6.4 Performance of Our Algorithm on Sequences with Outliers

In order to test the performance of our algorithm as a function of the percentage of outliers, we artificially added 0–25% of outliers to each motion sequence with complete data in the following manner. For each frame of each video sequence, we draw a percentage of outlying feature points independently and identically distributed from a uniform distribution in $[1, w] \times [1, h]$, where w and h are respectively the width and height of the image. We then applied RANSAC+GPCA and RANSAC to the point trajectories of each one of the motion sequences augmented with the so generated outlying trajectories.

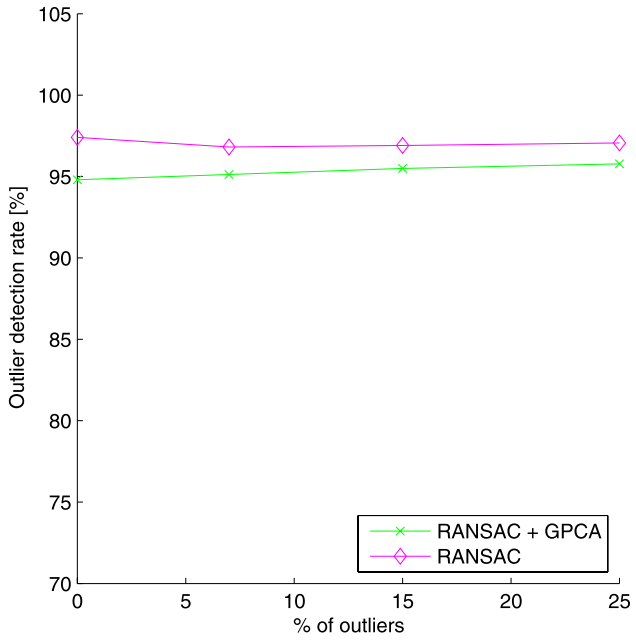


(a) $oc1R2RC$ (b) $oc1R2RCT$ (c) $oc2R3RCRT$

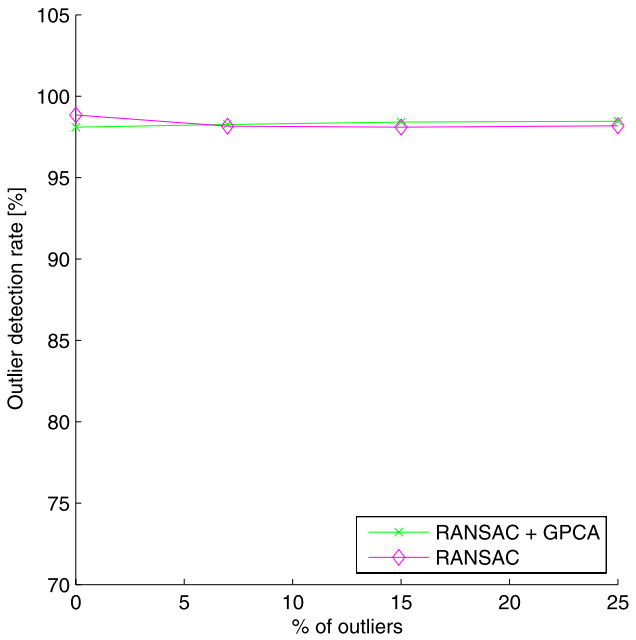
Fig. 7 Mask of missing data for the sequences $oc1R2RC$, $oc1R2RCT$ and $oc2R3RCRT$. The x-axis is the frame number and the y-axis is the feature point number. The color is black when a data point is missing, and zero otherwise

Figure 8 shows the percentage of correctly detected outliers as a function of the percentage of outliers given by RANSAC+GPCA and RANSAC. Notice that over 95% of the outliers are correctly detected by either method. Notice also that RANSAC+GPCA performs worse than RANSAC for two motions, and almost identically for three motions.

Figure 9 shows the classification error as a function of the percentage of outliers given by RANSAC+GPCA and RANSAC. A first point to notice is that the classification error without outliers higher than the errors reported in Sect. 6.2. This is expected, because in Sect. 6.2 all points are considered as inliers, while here inliers need to be detected. Since the detection of inliers is not perfect, the classification error increases, even with no true outliers in the data. Notice also that for two motions, RANSAC+GPCA performs

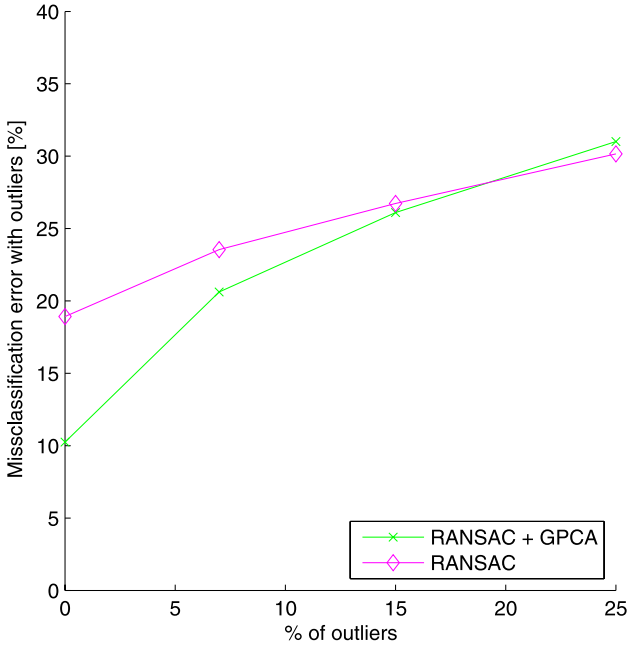


(a) Sequences with two motions

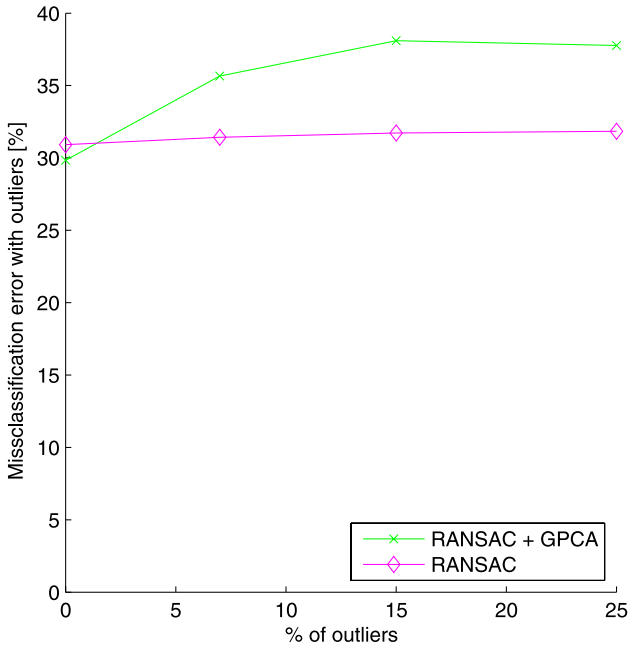


(b) Sequences with three motions

Fig. 8 Percentage of correctly detected outliers versus percentage of outliers



(a) Sequences with two motions



(b) Sequences with three motions

Fig. 9 Classification error as a function of the percentage of outliers

better than RANSAC, though the difference in performance reduces as the percentage of outliers increases. For three motions, however, RANSAC+GPCA performs better only when the percentage of outliers is small. This further confirms that RANSAC+GPCA is very effective for two motions, but has difficulties with three or more motions.

Tuning Thresholds for Outlier Detection The results in Figs. 8 and 9 are obtained with a fix value for the thresholds used by RANSAC+GPCA and RANSAC to determine if a point trajectory is an outlier or not. In order to set the values of these thresholds, we computed the percentage of false negatives and false positives obtained by each algorithm on the 155 sequences. These percentages are computed as follows. First, each motion sequence is contaminated with 15%

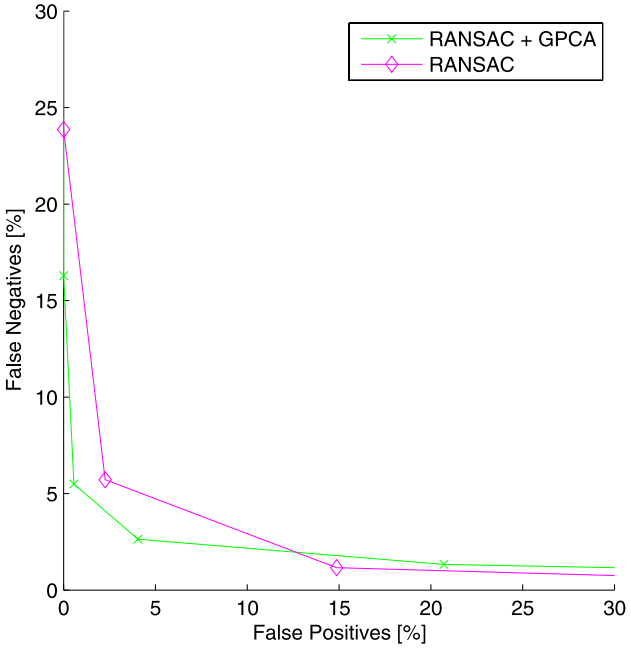


Fig. 10 Percentage of false negatives versus percentage of false positives for outlier detection for 155 motion sequences with 15% of outliers

of outliers. Then, each algorithm is applied to all sequences using several values for their respective thresholds. For each threshold and each sequence, this gives the number of false positives and false negatives. For each threshold, these numbers are then aggregated over all the sequences in order to obtain the percentages of false negatives and false positives shown in Fig. 10. From this figure, we chose the thresholds for RANSAC+GPCA and RANSAC that give 3% of false negatives. Notice from the figure that the performance of RANSAC+GPCA is slightly better than that of RANSAC in terms of false positives versus false negatives for outlier detection.

7 Conclusions and Future Work

We have presented a geometric algorithm for 3-D motion segmentation from multiple affine views, which deals with complete and incomplete data, and independent, partially dependent, full and degenerate motions. The algorithm uses SVD (complete data), PowerFactorization (missing data), or RANSAC (data with outliers) to project the data onto a five dimensional space, and GPCA to cluster the projected subspaces.

Experiments on a database of 120 motion sequences with two motions and complete data showed that our method is more accurate and robust than existing statistical methods. However, the performance of our approach deteriorates dramatically on sequences with three motions. This is because GPCA uses linear least squares to fit a large number of nonlinearly related coefficients, and so the estimated coefficients are inaccurate when the data is corrupted by noise and outliers. Our recent work (Tron and Vidal 2007) shows that the method of Yan and Pollefeys (2005) can give improved performance for sequences with two and three motions.

Open research avenues include improving the performance of the algorithm with three or more motions. The development of algorithms that exploit nonlinear constraints in the estimation of the polynomial used by GPCA will likely significantly reduce classification errors.

Acknowledgements This paper is an extended version of Vidal and Hartley (2004). We thank Mr. M. Behnisch for his help with data collection and tracking, Dr. K. Kanatani for providing his datasets and code, and Drs. Y. Yan and M. Pollefeys for providing their datasets. This work has been funded with startup funds from the Whiting School of Engineering of The Johns Hopkins University, by grants NSF CAREER IIS-0447739 and ONR N00014-05-10836, and by National ICT Australia (NICTA). NICTA is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.

References

Boult, T. E., & Brown, L. G. (1991). Factorization-based segmentation of motions. In *IEEE workshop on motion understanding* (pp. 179–186).

Buchanan, A., & Fitzgibbon, A. (2000). Damped Newton algorithms for matrix factorization with missing data. In *IEEE conference on computer vision and pattern recognition* (pp. 316–322).

Costeira, J., & Kanade, T. (1998). A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, *29*(3), 159–179.

De la Torre, F., & Black, M. J. (2001). Robust principal component analysis for computer vision. In *IEEE international conference on computer vision* (pp. 362–369).

Fan, Z., Zhou, J., & Wu, Y. (2006). Multibody grouping by inference of multiple subspaces from high-dimensional data using oriented-frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(1), 91–105.

Fischler, M. A., & Bolles, R. C. (1981). RANSAC random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *26*, 381–395.

Gear, C. W. (1998). Multibody grouping from motion images. *International Journal of Computer Vision*, *29*(2), 133–150.

Gruber, A., & Weiss, Y. (2004). Multibody factorization with uncertainty and missing data using the EM algorithm. In *IEEE conference on computer vision and pattern recognition* (Vol. I, pp. 707–714).

Hartley, R. (2003). PowerFactorization: an approach to affine reconstruction with missing and uncertain data. In *Australia–Japan advanced workshop on computer vision*.

Ichimura, N. (1999). Motion segmentation based on factorization method and discriminant criterion. In *IEEE international conference on computer vision* (pp. 600–605).

Kanatani, K. (2001). Motion segmentation by subspace separation and model selection. In *IEEE international conference on computer vision* (Vol. 2, pp. 586–591).

Kanatani, K., & Matsunaga, C. (2002). Estimating the number of independent motions for multibody motion segmentation. In *European conference on computer vision* (pp. 25–31).

Ng, A., Weiss, Y., & Jordan, M. (2001). On spectral clustering: analysis and an algorithm. In *Neural information processing systems*.

Oliensis, J., & Genc, Y. (2001). Fast and accurate algorithms for projective multi–image structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 546–559.

OpenCV (2000). http://sourceforge.net/projects/opencvlibrary.

Poelman, C. J., & Kanade, T. (1997). A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(3), 206–218.

Shum, H. Y., Ikeuchi, K., & Reddy, R. (1995). Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(9), 854–867.

Sugaya, Y., & Kanatani, K. (2002). Outlier removal for feature tracking by subspace separation. In *Proceedings of the 8th symposium on sensing via imaging information* (pp. 603–608).

Sugaya, Y., & Kanatani, K. (2004). Geometric structure of degeneracy for multi-body motion segmentation. In *Workshop on statistical methods in video processing*.

Tomasi, C., & Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, *9*, 137–154.

Tron, R., & Vidal, R. (2007). A benchmark for the comparson of 3-D motion segmentation algorithms. In *IEEE conference on computer vision and pattern recognition* (pp. 1–8).

Vidal, R., & Hartley, R. (2004). Motion segmentation with missing data by PowerFactorization and Generalized PCA. In *IEEE conference on computer vision and pattern recognition* (Vol. II, pp. 310–316).

Vidal, R., Ma, Y., & Sastry, S. (2005). Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(12), 1–15.

Vidal, R., Ma, Y., Soatto, S., & Sastry, S. (2006). Two-view multibody structure from motion. *International Journal of Computer Vision*, *68*(1), 7–25.

Vidal, R., & Oliensis, J. (2002). Structure from planar motions with small baselines. In *European conference on computer vision* (pp. 383–398).

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *IEEE international conference on computer vision* (pp. 975–982).

Wu, Y., Zhang, Z., Huang, T. S., & Lin, J. Y. (2001). Multibody grouping via orthogonal subspace decomposition. In *IEEE conference on computer vision and pattern recognition* (Vol. 2, pp. 252–257).

Yan, J., & Pollefeys, M. (2005). A factorization approach to articulated motion recovery. In *IEEE conference on computer vision and pattern recognition* (Vol. II, pp. 815–821).

Yan, J., & Pollefeys, M. (2006). A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *European conference on computer vision* (pp. 94–106).

Zelnik-Manor, L., & Irani, M. (2003). Degeneracies, dependencies and their implications in multi-body and multi-sequence factorization. In *IEEE conference on computer vision and pattern recognition* (Vol. 2, pp. 287–293).