A Unified Algebraic Approach to 2-D and 3-D Motion Segmentation*

René Vidal[1,2] and Yi Ma[3]

[1] Center for Imaging Science, Johns Hopkins University, Baltimore MD 21218, USA
[2] National ICT Australia, Canberra ACT 0200, Australia, rvidal@cis.jhu.edu
[3] Dept. of Elect. and Comp. Eng., UIUC, Urbana, IL 61801, USA, yima@uiuc.edu

Abstract. We present an analytic solution to the problem of estimating multiple 2-D and 3-D motion models from two-view correspondences or optical flow. The key to our approach is to view the estimation of multiple motion models as the estimation of a single *multibody motion model*. This is possible thanks to two important algebraic facts. First, we show that all the image measurements, regardless of their associated motion model, can be *fit* with a real or complex *polynomial*. Second, we show that the parameters of the motion model associated with an image measurement can be obtained from the *derivatives* of the polynomial at the measurement. This leads to a novel motion segmentation algorithm that applies to most of the two-view motion models adopted in computer vision. Our experiments show that the proposed algorithm outperforms existing algebraic methods in terms of efficiency and robustness, and provides a good initialization for iterative techniques, such as EM, which is strongly dependent on correct initialization.

1 Introduction

A classic problem in visual motion analysis is to estimate a motion model for a set of 2-D feature points as they move in a video sequence. Ideally, one would like to fit a single model that describes the motion of all the features. In practice, however, different regions of the image obey different motion models due to depth discontinuities, perspective effects, multiple moving objects, etc. Therefore, one is faced with the problem of fitting multiple motion models to the image, without knowing which pixels are moving according to the same model. More specifically:

Problem 1 (Multiple-motion estimation and segmentation). Given a set of image measurements $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ *taken from two views of a motion sequence related by a collection of* n *(*n *known) 2-D or 3-D motion models* $\{\mathcal{M}_i\}_{i=1}^n$*, estimate the motion models without knowing which image measurements correspond to which motion model.*

Related literature. There is a rich literature addressing the 2-D motion segmentation problem using the so-called layered representation [1] or different variations of the Expectation Maximization (EM) algorithm [2,3,4]. These approaches alternate between the segmentation of the image measurements (E-step) and the estimation of the motion

* The authors thank Jacopo Piazzi and Frederik Schaffalitzky for fruitful discussions. Research funded with startup funds from the departments of BME at Johns Hopkins and ECE at UIUC.

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3021, pp. 1–15, 2004.

parameters (M-step) and suffer from the disadvantage that the convergence to the optimal solution strongly depends on correct initialization [5,6]. Existing initialization techniques estimate the motion parameters from local patches and cluster these motion parameters using K-means [7], normalized cuts [5], or a Bayesian version of RANSAC [6]. The only existing algebraic solution to 2-D motion segmentation is based on bi-homogeneous polynomial factorization and can be found in [9].

The 3-D motion segmentation problem has received relatively less attention. Existing approaches include combinations of EM with normalized cuts [8] and factorization methods for orthographic and affine cameras [10,11]. Algebraic approaches based on polynomial and tensor factorization have been proposed in the case of multiple translating objects [12] and in the case of two [13] and multiple [14] rigid-body motions.

Our contribution. In this paper, we address the initialization of iterative approaches to motion estimation and segmentation by proposing a *non-iterative* algebraic solution to Problem 1 that applies to most 2-D and 3-D motion models in computer vision, as detailed in Table 1. The key to our approach is to view the estimation of multiple motion models as the estimation of a *single*, though more complex, *multibody motion model* that is then factored into the original models. This is achieved by (1) eliminating the feature segmentation problem in an algebraic fashion, (2) fitting a single multibody motion model to all the image measurements, and (3) segmenting the multibody motion model into its individual components. More specifically, our approach proceeds as follows:

1. *Eliminate Feature Segmentation*: Find an algebraic equation that is satisfied by all the image measurements, regardless of the motion model associated with each measurement. For the motion models considered in this paper, the i^{th} motion model will be typically defined by an algebraic equation of the form $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \mathcal{M}_i) = 0$. Therefore an algebraic equation that is satisfied by all the data is
$$g(\boldsymbol{x}_1, \boldsymbol{x}_2, \mathcal{M}) = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \mathcal{M}_1) f(\boldsymbol{x}_1, \boldsymbol{x}_2, \mathcal{M}_2) \cdots f(\boldsymbol{x}_1, \boldsymbol{x}_2, \mathcal{M}_n) = 0. \qquad (1)$$
Such an equation represents a single *multibody motion model* whose parameters $\mathcal{M}$ encode those of the original motion models $\{\mathcal{M}_i\}_{i=1}^n$.
2. *Multibody Motion Estimation*: Estimate the parameters $\mathcal{M}$ of the multibody motion model from the given image measurements. For the motion models considered in this paper, the parameters $\mathcal{M}$ will correspond to the coefficients of a real or complex polynomial p_n of degree n. We will show that if n is known such parameters can be estimated *linearly* after embedding the image data into a higher-dimensional space.
3. *Motion Segmentation*: Recover the parameters of the original motion models from the parameters of the multibody motion model $\mathcal{M}$, i.e.
$$\mathcal{M} \rightarrow \{\mathcal{M}_i\}_{i=1}^n. \qquad (2)$$
We will show that the individual motion parameters $\mathcal{M}_i$ can be computed from the *derivatives* of p_n evaluated at a collection of n image measurements.

This new approach offers two important technical advantages over previously known algebraic solutions to the segmentation of 3-D translational [12] and rigid-body motions (fundamental matrices) [14] based on homogeneous *polynomial factorization*:

Table 1. 2-D and 3-D motion models considered in this paper

Motion models	Model equations	Model parameters	Equivalent to clustering
2-D translational	$\boldsymbol{x}_2 = \boldsymbol{x}_1 + T_i$	$\{T_i \in \mathbb{R}^2\}_{i=1}^n$	Hyperplanes in $\mathbb{C}^2$
2-D similarity	$\boldsymbol{x}_2 = \lambda_i R_i \boldsymbol{x}_1 + T_i$	$\{(R_i, T_i) \in SE(2), \lambda_i \in \mathbb{R}^+\}_{i=1}^n$	Hyperplanes in $\mathbb{C}^3$
2-D affine	$\boldsymbol{x}_2 = A_i \begin{bmatrix} \boldsymbol{x}_1 \\ 1 \end{bmatrix}$	$\{A_i \in \mathbb{R}^{2\times3}\}_{i=1}^n$	Hyperplanes in $\mathbb{C}^4$
3-D translational	$0 = \boldsymbol{x}_2^T [T_i]_\times \boldsymbol{x}_1$	$\{T_i \in \mathbb{R}^3\}_{i=1}^n$	Hyperplanes in $\mathbb{R}^3$
3-D rigid-body	$0 = \boldsymbol{x}_2^T F_i \boldsymbol{x}_1$	$\{F_i \in \mathbb{R}^{3\times3} : \mathrm{rank}(F_i) = 2\}_{i=1}^n$	Bilinear forms in $\mathbb{R}^{3\times3}$
3-D homography	$\boldsymbol{x}_2 \sim H_i \boldsymbol{x}_1$	$\{H_i \in \mathbb{R}^{3\times3}\}_{i=1}^n$	Bilinear forms in $\mathbb{C}^{2\times3}$

1. It is based on *polynomial differentiation* rather than *polynomial factorization*, which greatly improves the efficiency, accuracy and robustness of the algorithm.
2. It applies to either feature correspondences or optical flows and includes most of the two-view motion models in computer vision: 2-D translational, similarity, and affine, or 3-D translational, rigid body motions (fundamental matrices), or motions of planar scenes (homographies), as shown in Table 1. The unification is achieved by embedding some of the motion models into the complex domain, which resolves cases such as 2-D affine motions and 3-D homographies that could not be solved in the real domain.

With respect to extant probabilistic methods, our approach has the advantage that it provides a global, non-iterative solution that does not need initialization. Therefore, our method can be used to initialize any iterative or optimization based technique, such as EM, or else in a layered (multiscale) or hierarchical fashion at the user's discretion.

Noisy image data. Although the derivation of the algorithm will assume noise free data, the algorithm is designed to work with moderate noise, as we will soon point out.

Notation. Let $\boldsymbol{z}$ be a vector in $\mathbb{R}^K$ or $\mathbb{C}^K$ and let $\boldsymbol{z}^T$ be its transpose. A homogeneous polynomial of degree n in $\boldsymbol{z}$ is a polynomial $p_n(\boldsymbol{z})$ such that $p_n(\lambda \boldsymbol{z}) = \lambda^n p_n(\boldsymbol{z})$ for all λ in $\mathbb{R}$ or $\mathbb{C}$. The space of all homogeneous polynomials of degree n in K variables, $R_n(K)$, is a vector space of dimension $M_n(K) = \binom{n+K-1}{K-1} = \binom{n+K-1}{n}$. A particular basis for $R_n(K)$ is obtained by considering all the monomials of degree n in K variables, that is $\boldsymbol{z}^I = z_1^{n_1} z_2^{n_2} \cdots z_K^{n_K}$ with $0 \leq n_j \leq n$ for $j = 1, \ldots, K$, and $n_1 + n_2 + \cdots + n_K = n$. Therefore, each polynomial $p_n(\boldsymbol{z}) \in R_n(K)$ can be written as a linear combination of a vector of coefficients $\boldsymbol{c} \in \mathbb{R}^{M_n(K)}$ or $\mathbb{C}^{M_n(K)}$ as

$$p_n(\boldsymbol{z}) = \boldsymbol{c}^T \nu_n(\boldsymbol{z}) = \sum c_{n_1, n_2, \cdots, n_K} z_1^{n_1} z_2^{n_2} \cdots z_K^{n_K}, \tag{3}$$

where $\nu_n : \mathbb{R}^K(\mathbb{C}^K) \rightarrow \mathbb{R}^{M_n(K)}(\mathbb{C}^{M_n(K)})$ is the *Veronese map* of degree n [12] defined as $\nu_n : [z_1, \ldots, z_K]^T \mapsto [\ldots, \boldsymbol{z}^I, \ldots]^T$ with I chosen in the degree-lexicographic order. The Veronese map is also known as the *polynomial embedding* in the machine learning community.

2 2-D Motion Segmentation by Clustering Hyperplanes in $\mathbb{C}^K$

2.1 Segmentation of 2-D Translational Motions: Clustering Hyperplanes in $\mathbb{C}^2$

The case of feature points. Under the 2-D translational motion model the two images are related by one out of n possible 2-D translations $\{T_i \in \mathbb{R}^2\}_{i=1}^n$. That is, for each feature pair $\boldsymbol{x}_1 \in \mathbb{R}^2$ and $\boldsymbol{x}_2 \in \mathbb{R}^2$ there exists a 2-D translation $T_i \in \mathbb{R}^2$ such that

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + T_i. \tag{4}$$

Therefore, if we interpret the displacement of the features $(\boldsymbol{x}_2 - \boldsymbol{x}_1)$ and the 2-D translations T_i as complex numbers $(\boldsymbol{x}_2 - \boldsymbol{x}_1) \in \mathbb{C}$ and $T_i \in \mathbb{C}$, then we can re-write equation (4) as

$$\boldsymbol{b}_i^T \boldsymbol{z} \doteq \begin{bmatrix} T_i & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -(\boldsymbol{x}_2 - \boldsymbol{x}_1) \end{bmatrix} = 0 \in \mathbb{C}^2. \tag{5}$$

The above equation corresponds to a hyperplane in $\mathbb{C}^2$ whose normal vector $\boldsymbol{b}_i$ encodes the 2-D translational motion T_i. Therefore, the segmentation of n 2-D translational motions $\{T_i \in \mathbb{R}^2\}_{i=1}^n$ from a set of correspondences $\{\boldsymbol{x}_1^j \in \mathbb{R}^2\}_{j=1}^N$ and $\{\boldsymbol{x}_2^j \in \mathbb{R}^2\}_{j=1}^N$ is equivalent to clustering data points $\{\boldsymbol{z}^j \in \mathbb{C}^2\}_{j=1}^N$ lying on n complex hyperplanes with normal vectors $\{\boldsymbol{b}_i \in \mathbb{C}^2\}_{i=1}^n$. As we will see in short, other 2-D and 3-D motion segmentation problems are also equivalent to clustering data lying on complex hyperplanes in $\mathbb{C}^3$ and $\mathbb{C}^4$. Therefore, rather than solving the hyperplane clustering problem for the case $K = 2$, we now present a solution for hyperplanes in $\mathbb{C}^K$ with arbitrary K by adapting the Generalized PCA algorithm of [15] to the complex domain.

Eliminating feature segmentation. We first notice that each point $\boldsymbol{z} \in \mathbb{C}^K$, regardless of which motion model $\{\boldsymbol{b}_i \in \mathbb{C}^K\}_{i=1}^n$ is associated with it, must satisfy the following homogeneous polynomial of degree n in K complex variables

$$p_n(\boldsymbol{z}) = \prod_{i=1}^n (\boldsymbol{b}_i^T \boldsymbol{z}) = \sum_I c_I \boldsymbol{z}^I = \sum c_{n_1,\ldots,n_K} z_1^{n_1} z_2^{n_2} \cdots z_K^{n_K} = \boldsymbol{c}^T \nu_n(\boldsymbol{z}) = 0, \tag{6}$$

where the coefficient vector $\boldsymbol{c} \in \mathbb{C}^{M_n(K)}$ represents the *multibody motion parameters*.

Estimating multibody motion. Since the polynomial p_n must be satisfied by all the data points $\boldsymbol{Z} = \{\boldsymbol{z}^j \in \mathbb{C}^K\}_{j=1}^N$, we obtain the following linear system on $\boldsymbol{c}$

$$\boxed{L_n \boldsymbol{c} = 0 \quad \in \mathbb{C}^N,} \tag{7}$$

where $L_n = [\nu_n(\boldsymbol{z}^1), \nu_n(\boldsymbol{z}^2), \ldots, \nu_n(\boldsymbol{z}^N)]^T \in \mathbb{C}^{N \times M_n(K)}$. One can show that there is a unique solution for $\boldsymbol{c}$ (up to a scale factor) if $N \geq M_n(K) - 1$ and at least $K - 1$ points belong to each hyperplane. Furthermore, since the last entry of each $\boldsymbol{b}_i$ is equal to one, then so is the last entry of $\boldsymbol{c}$. Therefore, one can solve for $\boldsymbol{c}$ uniquely. In the presence of noise, one can solve for $\boldsymbol{c}$ in a least-squares sense as the singular vector of L_n associated with its smallest singular value, and then normalize so that $\boldsymbol{c}_{M_n(K)} = 1$.

Segmenting the multibody motion. Given $\boldsymbol{c}$, we now present an algorithm for computing the motion parameters $\boldsymbol{b}_i$ from the derivatives of p_n. To this end, we consider the derivative of $p_n(\boldsymbol{z})$,

$$Dp_n(\boldsymbol{z}) = \frac{\partial p_n(\boldsymbol{z})}{\partial \boldsymbol{z}} = \sum_{i=1}^{n} \prod_{\ell \neq i} (\boldsymbol{b}_\ell^T \boldsymbol{z}) \boldsymbol{b}_i, \tag{8}$$

and notice that if we evaluate $Dp_n(\boldsymbol{z})$ at a point $\boldsymbol{z} = \boldsymbol{y}_i$ that corresponds to the i^{th} motion model, i.e. if $\boldsymbol{y}_i$ is such that $\boldsymbol{b}_i^T \boldsymbol{y}_i = 0$, then we have $Dp_n(\boldsymbol{y}_i) \sim \boldsymbol{b}_i$. Therefore, given $\boldsymbol{c}$ we can obtain the motion parameters as

$$\boxed{\boldsymbol{b}_i = \left. \frac{Dp_n(\boldsymbol{z})}{e_K^T Dp_n(\boldsymbol{z})} \right|_{\boldsymbol{z}=\boldsymbol{y}_i},} \tag{9}$$

where $e_K = [0, \ldots, 0, 1]^T \in \mathbb{C}^K$ and $\boldsymbol{y}_i \in \mathbb{C}^K$ is a nonzero vector such that $\boldsymbol{b}_i^T \boldsymbol{y}_i = 0$.

The rest of the problem is to find one vector $\boldsymbol{y}_i \in \mathbb{C}^K$ in each one of the hyperplanes $\mathcal{H}_i = \{\boldsymbol{z} \in \mathbb{C}^K : \boldsymbol{b}_i^T \boldsymbol{z} = 0\}$ for $i = 1, \ldots, n$. To this end, notice that we can always choose a point $\boldsymbol{y}_n$ lying on one of the hyperplanes as any of the points in the data set $\boldsymbol{Z}$. However, in the presence of noise and outliers, an arbitrary point in $\boldsymbol{Z}$ may be far from the hyperplanes. The question is then how to compute the *distance* from each data point to its closest hyperplane, *without* knowing the normals to the hyperplanes. The following lemma allows us to compute a first order approximation to such a distance:

Lemma 1. *Let $\tilde{\boldsymbol{z}} \in \mathcal{H}_i$ be the projection of a point $\boldsymbol{z} \in \mathbb{C}^K$ onto its closest hyperplane $\mathcal{H}_i$. Also let $\Pi = (I - e_K e_K^T)$. Then the Euclidean distance from $\boldsymbol{z}$ to $\mathcal{H}_i$ is given by*

$$\|\boldsymbol{z} - \tilde{\boldsymbol{z}}\| = \frac{|p_n(\boldsymbol{z})|}{\|\Pi Dp_n(\boldsymbol{z})\|} + O\big(\|\boldsymbol{z} - \tilde{\boldsymbol{z}}\|^2\big). \tag{10}$$

Therefore, we can choose a point in the data set close to one of the subspaces as:

$$\boxed{\boldsymbol{y}_n = \quad \arg\min_{\boldsymbol{z} \in \boldsymbol{Z}} \frac{|p_n(\boldsymbol{z})|}{\|\Pi Dp_n(\boldsymbol{z})\|},} \tag{11}$$

and then compute the normal vector at $\boldsymbol{y}_n$ as $\boldsymbol{b}_n = Dp_n(\boldsymbol{y}_n)/(e_K^T Dp_n(\boldsymbol{y}_n))$. In order to find a point $\boldsymbol{y}_{n-1}$ in one of the remaining hyperplanes, we could just remove the points on $\mathcal{H}_n$ from $\boldsymbol{Z}$ and compute $\boldsymbol{y}_{n-1}$ similarly to (11), but minimizing over $\boldsymbol{Z} \setminus \mathcal{H}_n$, and so on. However, the above process is not very robust in the presence of noise. Therefore, we propose an alternative solution that penalizes choosing a point from $\mathcal{H}_n$ in (11) by dividing the objective function by the distance from $\boldsymbol{z}$ to $\mathcal{H}_n$, namely $|\boldsymbol{b}_n^T \boldsymbol{z}|/\|\Pi \boldsymbol{b}_n\|$. That is, we can choose a point on or close to $\cup_{i=1}^{n-1} \mathcal{H}_i$ as

$$\boldsymbol{y}_{n-1} = \arg\min_{\boldsymbol{z} \in \boldsymbol{Z}} \frac{\frac{|p_n(\boldsymbol{z})|}{\|\Pi Dp_n(\boldsymbol{z})\|} + \delta}{\frac{|\boldsymbol{b}_n^T \boldsymbol{z}|}{\|\Pi \boldsymbol{b}_n\|} + \delta}, \tag{12}$$

where $\delta > 0$ is a small positive number chosen to avoid cases in which both the numerator and the denominator are zero (e.g. with perfect data). By repeating this process for the remaining hyperplanes, we obtain the following hyperplane clustering algorithm:

Algorithm 1 (Clustering hyperplanes in $\mathbb{C}^K$) *Given data points* $\boldsymbol{Z} = \{\boldsymbol{z}^j \in \mathbb{C}^K\}_{j=1}^N$

```
solve for $\boldsymbol{c} \in \mathbb{C}^{M_n(K)}$ from the linear system $[\nu_n(\boldsymbol{z}^1), \nu_n(\boldsymbol{z}^2), \ldots, \nu_n(\boldsymbol{z}^N)]^T \boldsymbol{c} = 0$;
set $p_n(\boldsymbol{z}) = \boldsymbol{c}^T \nu_n(\boldsymbol{z})$;
for $i = n : 1$,
    $\boldsymbol{y}_i = \arg\min_{\boldsymbol{z} \in \boldsymbol{Z}} \frac{\frac{|p_n(\boldsymbol{z})|}{\|\Pi D p_n(\boldsymbol{z})\|} + \delta}{\frac{|\boldsymbol{b}_{i+1}^T \boldsymbol{z}| \cdots |\boldsymbol{b}_n^T \boldsymbol{z}|}{\|\Pi \boldsymbol{b}_{i+1}\| \cdots \|\Pi \boldsymbol{b}_n\|} + \delta}; \qquad \boldsymbol{b}_i = \frac{D p_n(\boldsymbol{y}_i)}{e_K^T D p_n(\boldsymbol{y}_i)};$   (13)
end.
```

Notice that one could also choose the points $\boldsymbol{y}_i$ in a purely algebraic fashion, e.g., by intersecting a random line with the hyperplanes, or else by dividing the polynomial $p_n(\boldsymbol{z})$ by $\boldsymbol{b}_n^T \boldsymbol{z}$. However, we have chosen to present Algorithm 1 instead, because it has a better performance with noisy data and is not very sensitive to the choice of δ.

The case of translational optical flow. Imagine now that rather than a collection of feature points we are given the optical flow $\{\boldsymbol{u}_j \in \mathbb{R}^2\}_{j=1}^N$ between two consecutive views of a video sequence. If we assume that the optical flow is piecewise constant, i.e. the optical flow of every pixel in the image takes only n possible values $\{T_i \in \mathbb{R}^2\}_{i=1}^n$, then at each pixel $j \in \{1, \ldots, N\}$ there exists a motion T_i such that

$$\boldsymbol{u}_j = T_i. \tag{14}$$

The problem is now to estimate the n motion models $\{T_i\}_{i=1}^n$ from the optical flow $\{\boldsymbol{u}_j\}_{j=1}^N$. If $N \geq M_n(2) - 1 \sim O(n)$, this problem can be solved using the same technique as in the case of feature points (Algorithm 1 with $K = 3$) after replacing $\boldsymbol{x}_2 - \boldsymbol{x}_1 = \boldsymbol{u}$.

2.2 Segmentation of 2-D Similarity Motions: Clustering Hyperplanes in $\mathbb{C}^3$

The case of feature points. In this case, we assume that for each feature point $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ there exists a 2-D rigid-body motion $(R_i, T_i) \in SE(2)$ and a scale $\lambda_i \in \mathbb{R}^+$ such that

$$\boldsymbol{x}_2 = \lambda_i R_i \boldsymbol{x}_1 + T_i = \lambda_i \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \boldsymbol{x}_1 + T_i. \tag{15}$$

Therefore, if we interpret the rotation matrix as a unit number $R_i = \exp(\theta_i \sqrt{-1}) \in \mathbb{C}$, and the translation vector and the image features as points in the complex plane T_i, $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{C}$, then we can write the 2-D similarity motion model as the following hyperplane in $\mathbb{C}^3$:

$$\boldsymbol{b}_i^T \boldsymbol{z} \doteq \begin{bmatrix} \lambda_i R_i & T_i & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ 1 \\ -\boldsymbol{x}_2 \end{bmatrix} = 0. \tag{16}$$

Therefore, the segmentation of 2-D similarity motions is equivalent to clustering hyperplanes in $\mathbb{C}^3$. As such, we can apply Algorithm 1 with $K = 3$ to a collection of

$N \geq M_n(3) - 1 \sim O(n^2)$ image measurements $\{\boldsymbol{z}^j \in \mathbb{C}^3\}_{j=1}^N$, with at least two measurements per motion model, to obtain the motion parameters $\{\boldsymbol{b}_i \in \mathbb{C}^3\}_{i=1}^n$. The original *real* motion parameters are then given as

$$\lambda_i = |\boldsymbol{b}_{i1}|, \;\; \theta_i = \angle \boldsymbol{b}_{i1}, \;\; \text{and} \;\; T_i = [\mathrm{Re}(\boldsymbol{b}_{i2}), \mathrm{Im}(\boldsymbol{b}_{i2})]^T, \;\; \text{for} \;\; i = 1, \ldots, n. \quad (17)$$

The case of optical flow. Let $\{\boldsymbol{u}_j \in \mathbb{R}^2\}_{j=1}^N$ be N measurements of the optical flow at the N pixels $\{\boldsymbol{x}_j \in \mathbb{R}^2\}_{j=1}^N$. We assume that the optical flow field can be modeled as a collection of n 2-D similarity motion models as $\boldsymbol{u} = \lambda_i R_i \boldsymbol{x} + T_i$. Therefore, the segmentation of 2-D similarity motions from measurements of optical flow can be solved as in the case of feature points, after replacing $\boldsymbol{x}_2 = \boldsymbol{u}$ and $\boldsymbol{x}_1 = \boldsymbol{x}$.

2.3 Segmentation of 2-D Affine Motions: Clustering Hyperplanes in $\mathbb{C}^4$

The case of feature points. In this case, we assume that the images are related by a collection of n 2-D affine motion models $\{A_i \in \mathbb{R}^{2\times 3}\}_{i=1}^n$. That is, for each feature pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ there exist a 2-D affine motion A_i such that

$$\boldsymbol{x}_2 = A_i \begin{bmatrix} \boldsymbol{x}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} \; a_{12} \; a_{13} \\ a_{21} \; a_{22} \; a_{23} \end{bmatrix}_i \begin{bmatrix} \boldsymbol{x}_1 \\ 1 \end{bmatrix}. \quad (18)$$

Therefore, if we interpret $\boldsymbol{x}_2$ as a complex number $\boldsymbol{x}_2 \in \mathbb{C}$, but we still think of $\boldsymbol{x}_1$ as a vector in $\mathbb{R}^2$, then we have

$$\boldsymbol{x}_2 = \boldsymbol{a}_i^T \begin{bmatrix} \boldsymbol{x}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} + a_{21}\sqrt{-1} & a_{12} + a_{22}\sqrt{-1} & a_{13} + a_{23}\sqrt{-1} \end{bmatrix}_i \begin{bmatrix} \boldsymbol{x}_1 \\ 1 \end{bmatrix}. \quad (19)$$

The above equation represents the following hyperplane in $\mathbb{C}^4$

$$\boldsymbol{b}_i^T \boldsymbol{z} = \begin{bmatrix} \boldsymbol{a}_i^T \; 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ 1 \\ -\boldsymbol{x}_2 \end{bmatrix} = 0, \quad (20)$$

where the normal vector $\boldsymbol{b}_i \in \mathbb{C}^4$ encodes the affine motion parameters and the data point $\boldsymbol{z} \in \mathbb{C}^4$ encodes the image measurements $\boldsymbol{x}_1 \in \mathbb{R}^2$ and $\boldsymbol{x}_2 \in \mathbb{C}$. Therefore, the segmentation of 2-D affine motion models is equivalent to clustering hyperplanes in $\mathbb{C}^4$. As such, we can apply Algorithm 1 with $K = 4$ to a collection of $N \geq M_n(4) - 1 \sim O(n^3)$ image measurements $\{\boldsymbol{z}^j \in \mathbb{C}^4\}_{j=1}^N$, with at least three measurements per motion model, to obtain the motion parameters $\{\boldsymbol{b}_i \in \mathbb{C}^3\}_{i=1}^n$. The original affine motion models are then obtained as

$$A_i = \begin{bmatrix} \mathrm{Re}(\boldsymbol{b}_{i1}) \; \mathrm{Re}(\boldsymbol{b}_{i2}) \; \mathrm{Re}(\boldsymbol{b}_{i3}) \\ \mathrm{Im}(\boldsymbol{b}_{i1}) \; \mathrm{Im}(\boldsymbol{b}_{i2}) \; \mathrm{Im}(\boldsymbol{b}_{i3}) \end{bmatrix} \in \mathbb{R}^{2\times 3}, \quad \text{for} \;\; i = 1, \ldots, n. \quad (21)$$

The case of affine optical flow. In this case, the optical flow $\boldsymbol{u}$ is modeled as being generated by a collection of n affine motion models $\{A_i \in \mathbb{R}^{2\times 3}\}_{i=1}^n$ of the form $\boldsymbol{u} = A_i \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}$. Therefore, the segmentation of 2-D affine motions can be solved as in the case of feature points, after replacing $\boldsymbol{x}_2 = \boldsymbol{u}$ and $\boldsymbol{x}_1 = \boldsymbol{x}$.

3 3-D Motion Segmentation

3.1 Segmentation of 3-D Translational Motions: Clustering Hyperplanes in $\mathbb{R}^3$

The case of feature points. In this case, we assume that the scene can be modeled as a mixture of purely translational motion models, $\{T_i \in \mathbb{R}^3\}_{i=1}^n$, where T_i represents the translation (calibrated case) or the *epipole* (uncalibrated case) of object i relative to the camera between the two frames. A solution to this problem based on polynomial factorization was proposed in [12]. Here we present a much simpler solution based on polynomial differentiation.

Given the images $\boldsymbol{x}_1 \in \mathbb{P}^2$ and $\boldsymbol{x}_2 \in \mathbb{P}^2$ of a point in object i in the first and second frame, they must satisfy the well-known epipolar constraint for linear motions

$$-\boldsymbol{x}_2^T[T_i]_\times \boldsymbol{x}_1 = T_i^T(\boldsymbol{x}_2 \times \boldsymbol{x}_1) = T_i^T \boldsymbol{\ell} = 0, \tag{22}$$

where $\boldsymbol{\ell} = (\boldsymbol{x}_2 \times \boldsymbol{x}_1) \in \mathbb{R}^3$ is known as the *epipolar line* associated with the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$. Therefore, the segmentation of 3-D translational motions is equivalent to clustering data (epipolar lines) lying on a collection of hyperplanes in $\mathbb{R}^3$ whose normal vectors are the n epipoles $\{T_i\}_{i=1}^n$. As such, we can apply Algorithm 1 with $K = 3$ to $N \geq M_n(3) - 1 \sim O(n^2)$ epipolar lines $\{\boldsymbol{\ell}^j = \boldsymbol{x}_1^j \times \boldsymbol{x}_2^j\}_{j=1}^N$, with at least two epipolar lines per motion, to estimate the epipoles $\{T_i\}_{i=1}^n$ from the derivatives of the polynomial $p_n(\boldsymbol{\ell}) = (T_1^T\boldsymbol{\ell}) \cdots (T_n^T\boldsymbol{\ell})$. The only difference is that in this case the last entry of each epipole is *not* constrained to be equal to one. Therefore, when choosing the points $\boldsymbol{y}_i$ in equation (13) we should take $\Pi = I$ not to eliminate the last coordinate. We therefore compute the epipoles up to an unknown scale factor as

$$\boxed{T_i = Dp_n(\boldsymbol{y}_i)/\|Dp_n(\boldsymbol{y}_i)\|, \qquad i = 1, \ldots, n,} \tag{23}$$

where the unknown scale is lost under perspective projection.

The case of optical flow. In the case of optical flow generated by purely translating objects we have $\boldsymbol{u}^T[T_i]_\times \boldsymbol{x} = 0$, where $\boldsymbol{u}$ is interpreted as a three vector $[\mathrm{u}, \mathrm{v}, 0]^T \in \mathbb{R}^3$. Thus, one can estimate the translations $\{T_i \in \mathbb{R}^3\}_{i=1}^n$ as before by replacing $\boldsymbol{x}_2 = \boldsymbol{u}$ and $\boldsymbol{x}_1 = \boldsymbol{x}$.

3.2 Segmentation of 3-D Rigid-Body Motions: Clustering Quadratic Forms in $\mathbb{R}^{3\times 3}$

Assume that the motion of the objects relative to the camera between the two views can be modeled as a mixture of 3-D rigid-body motions $\{(R_i, T_i) \in SE(3)\}_{i=1}^n$ which are represented with a nonzero rank-2 *fundamental matrix* F_i. A solution to this problem based on the factorization of bi-homogeneous polynomials was proposed in [14]. Here we present a much simpler solution based on taking derivatives of the so-called multibody epipolar constraint (see below), thus avoiding polynomial factorization.

Given an image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, there exists a motion i such that the following epipolar constraint is satisfied

$$\boldsymbol{x}_2^T F_i \boldsymbol{x}_1 = 0. \tag{24}$$

Therefore, the following *multibody epipolar constraint* [14] must be satisfied by the number of independent motions n, the fundamental matrices $\{F_i\}_{i=1}^n$ and the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, regardless of the object to which the image pair belongs

$$p_n(\boldsymbol{x}_1, \boldsymbol{x}_2) \doteq \prod_{i=1}^{n} \left(\boldsymbol{x}_2^T F_i \boldsymbol{x}_1\right) = 0. \tag{25}$$

It was also shown in [14] that the multibody epipolar constraint can be written in bilinear form as $\nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1) = 0$, where $\mathcal{F} \in \mathbb{R}^{M_n(3)\times M_n(3)}$ is the so-called *multibody fundamental matrix*, which can be linearly estimated from $N \geq M_n(3)^2 - 1 \sim O(n^4)$ image pairs in general position with at least 8 pairs corresponding to each motion.

We now present a new solution to the problem of estimating the fundamental matrices $\{F_i\}_{i=1}^n$ from the multibody fundamental matrix $\mathcal{F}$ based on taking derivatives of the multibody epipolar constraint. Recall that, given a point $\boldsymbol{x}_1 \in \mathbb{P}^2$ in the first image frame, the epipolar lines associated with it are defined as $\boldsymbol{\ell}_i \doteq F_i \boldsymbol{x}_1 \in \mathbb{R}^3, i = 1, \ldots, n$. Therefore, if the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ corresponds to motion i, i.e. if $\boldsymbol{x}_2^T F_i \boldsymbol{x}_1 = 0$, then

$$\frac{\partial}{\partial \boldsymbol{x}_2} \nu_n(\boldsymbol{x}_2)^T \mathcal{F} \nu_n(\boldsymbol{x}_1) = \sum_{i=1}^{n} \prod_{\ell \neq i} (\boldsymbol{x}_2^T F_\ell \boldsymbol{x}_1)(F_i \boldsymbol{x}_1) = \prod_{\ell \neq i} (\boldsymbol{x}_2^T F_\ell \boldsymbol{x}_1)(F_i \boldsymbol{x}_1) \sim \boldsymbol{\ell}_i. \tag{26}$$

In other words, the partial derivative of the multibody epipolar constraint with respect to $\boldsymbol{x}_2$ evaluated at $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is proportional to *the* epipolar line associated with $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the second view.[1] Therefore, given a set of image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ and the multibody fundamental matrix $\mathcal{F} \in \mathbb{R}^{M_n(3)\times M_n(3)}$, we can estimate a collection of epipolar lines $\{\boldsymbol{\ell}^j\}_{j=1}^N$. Remember from Section 3.1 that in the case of purely translating objects the epipolar lines were readily obtained as $\boldsymbol{x}_1 \times \boldsymbol{x}_2$. Here the calculation is more involved because of the rotational component of the rigid-body motions. Nevertheless, given a set of epipolar lines we can apply Algorithm 1 with $K = 3$ and $\Pi = I$ to estimate the n epipoles $\{T_i\}_{i=1}^n$ up to a scale factor, as in equation (23). Therefore, if the n epipoles are different,[2] then we can immediately compute the n fundamental matrices $\{F_i\}_{i=1}^n$ by assigning the image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ to group i if $i = \arg\min_{\ell=1,\ldots,n} (T_i^T \boldsymbol{\ell}^j)^2$ and then applying the eight-point algorithm to the image pairs in group $i = 1, \ldots, n$.

3.3 Segmentation of 3-D Homographies: Clustering Quadratic Forms in $\mathbb{C}^{2\times 3}$

The motion segmentation scheme described in the previous section assumes that the displacement of each object between the two views relative to the camera is nonzero, i.e. $T_i \neq 0$, otherwise the individual fundamental matrices are zero. Furthermore, it also

[1] Similarly, the partial derivative of the multibody epipolar constraint with respect to $\boldsymbol{x}_1$ evaluated at $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is proportional to *the* epipolar line associated with $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in the first view.

[2] Notice that this is not a strong assumption. If two individual fundamental matrices share the same (left) epipoles, one can consider the right epipoles (in the first image frame) instead, because it is extremely rare that two motions give rise to the same left and right epipoles. In fact, this happens only when the rotation axes of the two motions are equal to each other and parallel to the translation direction [14].

requires that the 3-D points be in general configuration, otherwise one cannot uniquely recover each fundamental matrix from its epipolar constraint. The latter case occurs, for example, in the case of planar structures, i.e. when the 3-D points lie on a plane [16].

Both in the case of purely rotating objects (relative to the camera) or in the case of a planar 3-D structure, the motion model between the two views $\boldsymbol{x}_1 \in \mathbb{P}^2$ and $\boldsymbol{x}_2 \in \mathbb{P}^2$ is described by a homography matrix $H \in \mathbb{R}^{3\times 3}$ such that [16]

$$\boldsymbol{x}_2 \sim H\boldsymbol{x}_1 = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \boldsymbol{x}_1. \tag{27}$$

Consider now the case in which we are given a set of image pairs $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^N$ that can be modeled with n *independent* homographies $\{H_i\}_{i=1}^n$ (see Remark 2). Note that the n homographies do *not* necessarily correspond to n different rigid-body motions. This is because it could be the case that one rigidly moving object consists of two or more planes, hence its rigid-body motion will lead to two or more homographies. Therefore, the n homographies can represent anything from 1 up to n rigid-body motions. In either case, it is evident from the form of equation (27) that we cannot take the product of all the equations, as we did with the epipolar constraints, because we have two linearly independent equations per image pair. Nevertheless, we show now that one can still solve the problem by working in the complex domain, as we describe below.

We interpret the second image $\boldsymbol{x}_2 \in \mathbb{P}^2$ as a point in $\mathbb{CP}$ by considering the first two coordinates in $\boldsymbol{x}_2$ as a complex number and appending a one to it. However, we still think of $\boldsymbol{x}_1$ as a point in $\mathbb{P}^2$. With this interpretation, we can rewrite (27) as

$$\boldsymbol{x}_2 \sim H\boldsymbol{x}_1 \doteq \begin{bmatrix} h_{11} + h_{21}\sqrt{-1} & h_{12} + h_{22}\sqrt{-1} & h_{13} + h_{23}\sqrt{-1} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \boldsymbol{x}_1, \tag{28}$$

where $H \in \mathbb{C}^{2\times 3}$ now represents a *complex homography*[3]. Let $\boldsymbol{w}_2$ be the vector in $\mathbb{CP}$ perpendicular to $\boldsymbol{x}_2$, i.e. if $\boldsymbol{x}_2 = (z, 1)$ then $\boldsymbol{w}_2 = (1, -z)$. Then we can rewrite (28) as the following *complex* bilinear constraint

$$\boldsymbol{w}_2^T H\boldsymbol{x}_1 = 0, \tag{29}$$

which we call the *complex homography constraint*. We can therefore interpret the motion segmentation problem as one in which we are given image data $\{\boldsymbol{x}_1^j \in \mathbb{P}^2\}_{j=1}^N$ and $\{\boldsymbol{w}_2^j \in \mathbb{CP}\}_{j=1}^N$ generated by a collection of n complex homographies $\{H_i \in \mathbb{C}^{2\times 3}\}_{i=1}^n$. Then each image pair $(\boldsymbol{x}_1, \boldsymbol{w}_2)$ has to satisfy the *multibody homography constraint*

$$\prod_{i=1}^n (\boldsymbol{w}_2^T H_i \boldsymbol{x}_1) = \nu_n(\boldsymbol{w}_2)^T \mathcal{H} \nu_n(\boldsymbol{x}_1) = 0, \tag{30}$$

regardless of which one of the n complex homographies is associated with the image pair. We call the matrix $\mathcal{H} \in \mathbb{C}^{M_n(2)\times M_n(3)}$ the *multibody homography*. Now, since the multibody homography constraint (30) is linear in the multibody homography $\mathcal{H}$, we

[3] Strictly speaking, we embed each real homography matrix into an affine complex matrix.

can linearly solve for $\mathcal{H}$ from (30) given $N \geq M_n(2)M_n(3) - (M_n(3)+1)/2 \sim O(n^3)$ image pairs in general position[4] with at least 4 pairs per moving object.

Given the multibody homography $\mathcal{H} \in \mathbb{C}^{M_n(2)\times M_n(3)}$, the rest of the problem is to recover the individual homographies $\{H_i\}_{i=1}^n$. In the case of fundamental matrices discussed in Section 3.2, the key for solving the problem was the fact that fundamental matrices are of rank 2, hence one can cluster epipolar lines based on the epipoles. In principle, we cannot do the same with real homographies $H_i \in \mathbb{R}^{3\times 3}$, because in general they are full rank. However, if we work with complex homographies $H_i \in \mathbb{C}^{2\times 3}$ they automatically have a right null space which we call the *complex epipole* $\boldsymbol{e}_i \in \mathbb{C}^3$. Then, similarly to (26), we can associate a *complex epipolar line*

$$\boldsymbol{\ell}^j \sim \left.\frac{\partial \nu_n(\boldsymbol{w}_2)^T \mathcal{H} \nu_n(\boldsymbol{x}_1)}{\partial \boldsymbol{x}_1}\right|_{(\boldsymbol{x}_1,\boldsymbol{w}_2)=(\boldsymbol{x}_1^j,\boldsymbol{w}_2^j)} \in \mathbb{CP}^2 \qquad (31)$$

with each image pair $(\boldsymbol{x}_1^j, \boldsymbol{w}_2^j)$. Given this set of $N \geq M_n(3) - 1$ complex epipolar lines $\{\boldsymbol{\ell}^j\}_{j=1}^N$, with at least 2 lines per moving object, we can apply Algorithm 1 with $K = 3$ and $\Pi = I$ to estimate the n complex epipoles $\{\boldsymbol{e}_i \in \mathbb{C}^3\}_{i=1}^n$ up to a scale factor, as in equation (23). Therefore, if the n complex epipoles are different, we can cluster the original image measurements by assigning image pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ to group i if $i = \arg\min_{\ell=1,\ldots,n} |\boldsymbol{e}_\ell^T \boldsymbol{\ell}^j|^2$. Once the image pairs have been clustered, the estimation of each homography, either real or complex, becomes a simple linear problem.

Remark 1 (Direct extraction of homographies from $\mathcal{H}$). There is yet another way to obtain individual H_i from $\mathcal{H}$ without segmenting the image pairs first. Once the complex epipoles $\boldsymbol{e}_i$ are known, one can compute the following linear combination of the rows of H_i (up to scale) from the derivatives of the multibody homography constraint at $\boldsymbol{e}_i$

$$\boldsymbol{w}^T H_i \sim \left.\frac{\partial \nu_n(\boldsymbol{w})^T \mathcal{H} \nu_n(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{e}_i} \in \mathbb{CP}^2, \quad \forall \boldsymbol{w} \in \mathbb{C}^2. \qquad (32)$$

In particular, if we take $\boldsymbol{w} = [1,0]^T$ and $\boldsymbol{w} = [0,1]^T$ we obtain the first and second row of H_i (up to scale), respectively. By choosing additional $\boldsymbol{w}$'s one obtains more linear combinations from which the rows of H_i can be linearly and uniquely determined.

Remark 2 (Independent homographies). The above solution assumes that the complex epipoles are different (up to a scale factor). We take this assumption as our definition of independent homographies, even though it is more restrictive than saying than the real homographies $H_i \in \mathbb{R}^{3\times 3}$ are different (up to a scale factor). However, one can show that, under mild conditions, e.g., the third rows of each H_i are different, the null spaces of the complex homographies are indeed different for different real homographies.[5]

[4] The multibody homography constraint gives two equations per image pair, and there are $(M_n(2)-1)M_n(3)$ complex entries in $\mathcal{H}$ and $M_n(3)$ real entries (the last row).

[5] The set of complex homographies that share the same null space is a five-dimensional subset (hence a zero-measure subset) of all real homography matrices. Furthermore, one can complexify any other two rows of H instead of the first two. As long as two homography matrices are different, one of the complexifications will give different complex epipoles.

Remark 3 (One rigid-body motion versus multiple ones). A homography is generally of the form $H = R + T\pi^T$ where π is the plane normal. If the homographies come from different planes (different π) undergoing the same rigid-body motion, the proposed scheme would work just fine since different normal vectors π will cause the complex epipoles to be different. However, if multiple planes with the same normal vector $\pi = [0, 0, 1]^T$ undergo pure translational motions of the form $T_i = [T_{xi}, T_{yi}, T_{zi}]^T$, then all the complex epipoles are equal to $\boldsymbol{e}_i = [\sqrt{-1}, -1, 0]^T$. To avoid this problem, one can complexify the first and third rows of H instead of the first two. The new complex epipoles are $\boldsymbol{e}_i{=}[T_{xi}{+}T_{zi}\sqrt{-1}, T_{yi}, -1]^T$, which are different for different translations.

4 Experiments on Real and Synthetic Images

2-D translational. We tested our polynomial differentiation algorithm (PDA) by segmenting 12 frames of a sequence consisting of an aerial view of two robots moving on the ground. The robots are purposely moving slowly, so that it is harder to distinguish the flow from the noise. At each frame, we applied Algorithm 1 with $K = 2$ and $\delta = 0.02$ to the optical flow[6] of all $N = 240 \times 352$ pixels in the image and segmented the image measurements into $n = 3$ translational motion models. The leftmost column of Figure 1 displays the x and y coordinates of the optical flow for frames 4 and 10, showing that it is not so simple to distinguish the three clusters from the raw data. The remaining columns of Figure 1 show the segmentation of the image pixels. The motion of the two robots and that of the background are correctly segmented. We also applied Algorithm 1 to the optical flow of the flower garden sequence. Figure 2 shows the optical flow of one frame and the segmentation of the pixels into three groups: the tree, the grass, and the background. Notice that the boundaries of the tree can be assigned to any group, and in this case they are grouped with the grass.

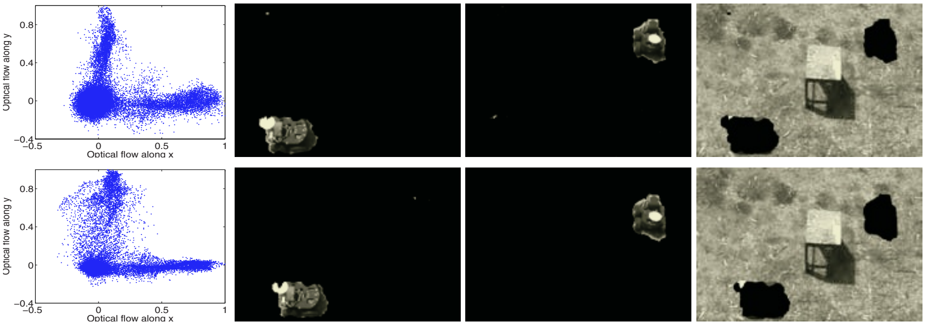


Fig. 1. Segmenting the optical flow of the two-robot sequence by clustering lines in $\mathbb{C}^2$

[6] We compute optical flow using Black's code at http://www.cs.brown.edu/people/black/ignc.html.

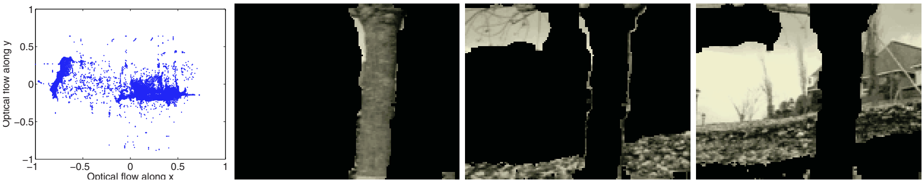
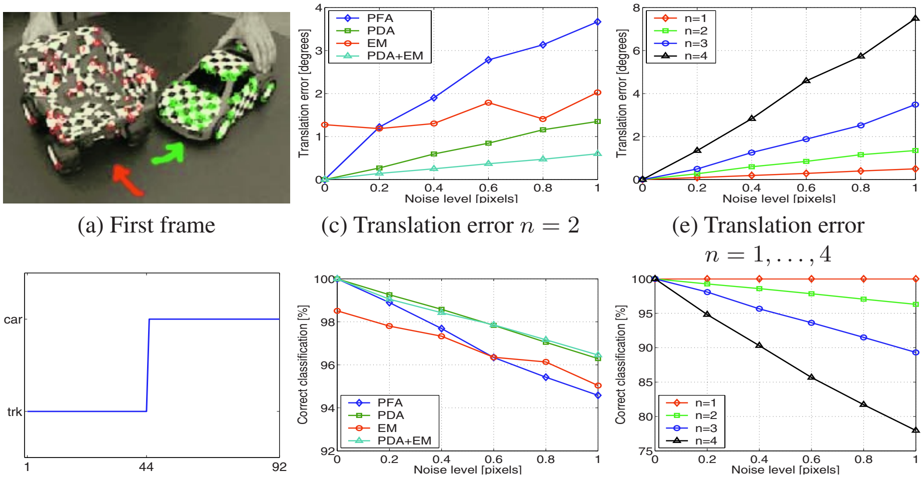


Fig. 2. Segmenting the optical flow of the flower-garden sequence by clustering lines in $\mathbb{C}^2$

3-D translational motions. Figure 3(a) shows the first frame of a 320 × 240 video sequence containing a truck and a car undergoing two 3-D translational motions. We applied Algorithm 1 with $K = 3$, $\Pi = I$ and $\delta = 0.02$ to the (real) epipolar lines obtained from a total of $N = 92$ features, 44 in the truck and 48 in the car. The algorithm obtained a perfect segmentation of the features, as shown in Figure 3(b), and estimated the epipoles with an error of 5.9° for the truck and 1.7° for the car. We also tested the performance of PDA on synthetic data corrupted with zero-mean Gaussian noise with s.t.d. between 0 and 1 pixels for an image size of 500 × 500 pixels. For comparison purposes, we also implemented the polynomial factorization algorithm (PFA) of [12] and a variation of the Expectation Maximization algorithm (EM) for clustering hyperplanes in $\mathbb{R}^3$. Figures 3(c) and (d) show the performance of all the algorithms as a function of the level of noise for $n = 2$ moving objects. The performance measures are the mean error between the estimated and the true epipoles (in degrees), and the mean percentage of correctly segmented features using 1000 trials for each level of noise. Notice that



(a) First frame

(c) Translation error $n = 2$

(e) Translation error $n = 1, \ldots, 4$

(b) Feature segmentation

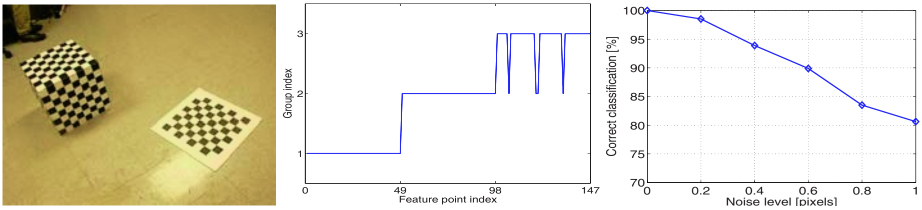
(d) % of correct classif. $n = 2$

(f) % of correct classif. $n = 1, \ldots, 4$

Fig. 3. Segmenting 3-D translational motions by clustering planes in $\mathbb{R}^3$. Left: segmenting a real sequence with 2 moving objects. Center: comparing our algorithm with PFA and EM as a function of noise in the image features. Right: performance of PFA as a function of the number of motions

PDA gives an error of less than $1.3°$ and a classification performance of over 96%. Thus our algorithm PDA gives approximately 1/3 the error of PFA, and improves the classification performance by about 2%. Notice also that EM with the normal vectors initialized at random (EM) yields a nonzero error in the noise free case, because it frequently converges to a local minimum. In fact, our algorithm PDA outperforms EM. However, if we use PDA to initialize EM (PDA+EM), the performance of both EM and PDA improves, showing that our algorithm can be effectively used to initialize iterative approaches to motion segmentation. Furthermore, the number of iterations of PDA+EM is approximately 50% with respect to EM randomly initialized, hence there is also a gain in computing time. Figures 3(e) and (f) show the performance of PDA as a function of the number of moving objects for different levels of noise. As expected, the performance deteriorates with the number of moving objects, though the translation error is still below $8°$ and the percentage of correct classification is over 78%.

3-D homographies. Figure 4(a) shows the first frame of a 2048×1536 video sequence with two moving objects: a cube and a checkerboard. Notice that although there are only *two* rigid motions, the scene contains *three* different homographies, each one associated with each one of the visible planar structures. Furthermore, notice that the top side of the cube and the checkerboard have approximately the same normals. We manually tracked a total of $N = 147$ features: 98 in the cube (49 in each of the two visible sides) and 49 in the checkerboard. We applied our algorithm in Section 3.3 with $\Pi = I$ and $\delta = 0.02$ to segment the image data and obtained a 97% of correct classification, as shown in Figure 4(b). We then added zero-mean Gaussian noise with standard deviation between 0 and 1 pixels to the features, after rectifying the features in the second view in order to simulate the noise free case. Figure 4(c) shows the mean percentage of correct classification for 1000 trials per level of noise. The percentage of correct classification of our algorithm is between 80% and 100%, which gives a very good initial estimate for any of the existing iterative/optimization/EM based motion segmentation schemes.



(a) First frame (b) Feature segmentation (c) $\%$ of correct classification

Fig. 4. Segmenting 3-D homographies by clustering complex bilinear forms in $\mathbb{C}^{2\times 3}$

5 Conclusions

We have presented a unified algebraic approach to 2-D and 3-D motion segmentation from feature correspondences or optical flow. Contrary to extant methods, our approach does not iterate between feature segmentation and motion estimation. Instead, it computes a single multibody motion model that is satisfied by all the image measurements and then extracts the original motion models from the derivatives of the multibody one. Various experiments showed that our algorithm not only outperforms existing algebraic methods with much limited applicability, but also provides a good initialization for iterative techniques, such as EM, which are strongly dependent on correct initialization.

References

1. Darrel, T., Pentland, A.: Robust estimation of a multi-layered motion representation. In: IEEE Workshop on Visual Motion. (1991) 173–178
2. Jepson, A., Black, M.: Mixture models for optical flow computation. In: CVPR. (1993) 760–761
3. Ayer, S., Sawhney, H.: Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In: ICCV. (1995) 777–785
4. Weiss, Y.: Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In: CVPR. (1997) 520–526
5. Shi, J., Malik, J.: Motion segmentation and tracking using normalized cuts. In: ICCV. (1998) 1154–1160
6. Torr, P., Szeliski, R., Anandan, P.: An integrated Bayesian approach to layer extraction from image sequences. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001) 297–303
7. Wang, J., Adelson, E.: Layered representation for motion analysis. In: CVPR. (1993) 361–366
8. Feng, X., Perona, P.: Scene segmentation from 3D motion. In: CVPR. (1998) 225–231
9. Vidal, R., Sastry, S.: Segmentation of dynamic scenes from image intensities. In: IEEE Workshop on Vision and Motion Computing. (2002) 44-49
10. Costeira, J., Kanade, T.: Multi-body factorization methods for motion analysis. In: ICCV. (1995) 1071–1076
11. Kanatani, K.: Motion segmentation by subspace separation and model selection. In: ICCV. (2001) 586–591
12. Vidal, R., Ma, Y., Sastry, S.: Generalized principal component analysis (GPCA). In: CVPR. (2003) 621-628
13. Wolf, L., Shashua, A.: Two-body segmentation from two perspective views. In: CVPR. (2001) 263–270
14. Vidal, R., Ma, Y., Soatto, S., Sastry, S.: Two-view multibody structure from motion. To appear in International Journal of Computer Vision (2004)
15. Vidal, R., Ma, Y., J. Piazzi: A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials. In: CVPR. (2004)
16. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge (2000)