Fast and accurate motion segmentation using Linear Combination of Views

Vasileios Zografos
zografos@isy.liu.se
Klas Nordberg
klas@isy.liu.se

Computer Vision Laboratory
Linköping University
SE-581 83 Linköping, Sweden

Abstract

We introduce a simple and efficient procedure for the segmentation of rigidly moving objects, imaged under an affine camera model. For this purpose we revisit the theory of "linear combination of views" (LCV), proposed by Ullman and Basri [20], which states that the set of 2d views of an object undergoing 3d rigid transformations, is embedded in a low-dimensional linear subspace that is spanned by a small number of basis views. Our work shows, that one may use this theory for motion segmentation, and cluster the trajectories of 3d objects using only two 2d basis views. We therefore propose a practical motion segmentation method, built around LCV, that is very simple to implement and use, and in addition is very fast, meaning it is well suited for real-time SfM and tracking applications. We have experimented on real image sequences, where we show good segmentation results, comparable to the state-of-the-art in literature. If we also consider computational complexity, our proposed method is one of the best performers in combined speed and accuracy.

1 Introduction

The motion segmentation problem deals with the partitioning of a sequence of images into distinct regions, where each region describes a separate motion. We will restrict ourselves to 3d rigid motions and non-dense object representations, that is, when sparse sets of feature points are tracked over time, and their trajectories analysed in the images. In addition, under the assumption that we generally encounter small depth variations in the imaged scene, we may consequently utilise an affine camera model.

In [17] the authors have shown that under the affine model, the feature point trajectories of a single rigid motion lie in a low-dimensional linear subspace, and trajectories from multiple rigid objects will lie in a union of such linear subspaces. Hence, the problem of motion segmentation is simplified to one of clustering data points drawn from a union of affine subspaces. The vast majority of motion segmentation methods in literature, make use of this simplifying assumption, and are in essence subspace fitting and clustering approaches. We name a few of the most successful methods (relative to the standard dataset [19]) such as: Sparse Subspace Clustering (SSC) [6], Spectral Curvature Clustering (SCC) [4], and quite recently Principle Angles Configuration (PAC) [24], and Local Best-Fit Flats (SLBF)

BMVC 2011 http://dx.doi.org/10.5244/C.25.12

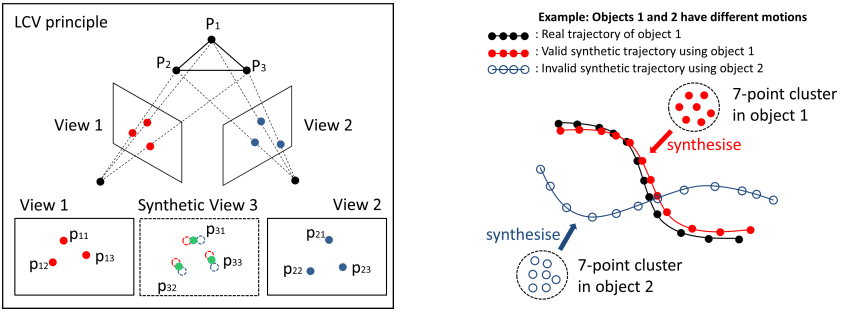


Figure 1: The main principle of LCV synthesis is illustrated on the left. On the right, we show how this idea can be used for generating motion affinities, leading to segmentation.

[25]. Older methods such as LSA [23], ALC [10] MSL [16], GPCA [22] and RANSAC-type subspace fitting approaches have since been surpassed. Of course, one is not limited to subspace methods for solving the motion segmentation problem, or indeed an affine camera model. Of particular interest are for example, the epipolar geometry method by [21], and the earlier work by [18], and the 6-point invariants approach by [12].

In this work we propose a novel approach, which similarly assumes an affine camera model, but does not belong in the usual domain of subspace clustering methods. Our approach is a simple geometric solution, that is based on the observation that one may synthesise the motion trajectories of feature points in an image sequence, by using an algebraic combination of other feature points that belong to the same object. In order to synthesise the point trajectories, we use the "linear combination of views" (LCV) theory by [20], which although pertains to an affine camera model, offers a fast, accurate and practical solution.

In the next section, we will explore the fundamental aspects of LCV theory and its existing applications and uses. We continue with a detailed description of our method and propose a very simple *synthesise*, *test* and *classify* algorithm for motion segmentation, which is very fast, straightforward to implement and has a single parameter that is automatically tuned. Section 4 summarises our experiments on real image sequences, where we illustrate that our LCV-based method compares very favourably against other methods in literature, in terms of accuracy. If we also factor in the overall execution speed, we can see that our approach is amongst the best. This means that it can actually be used in practice, for real-time structure-from-motion and tracking applications, unlike most of the other existing popular methods. We conclude with an overview in Section 5.

2 LCV theory

The LCV theory was originally described by Ullman and Basri in [20], and simply states that under an affine camera model, the set of views depicting an object, which is undergoing 3d rigid transformations, can in principle be expressed as linear combinations of a small number of views of that object. This theory allows us to synthesise novel, valid (as far as 3d rigid transformations are concerned) views of an object, using a small number (usually 2) of stored views of the object, instead of generating and storing a full 3d model (Fig. 1). This idea was initially proposed for recognition of wireframe objects, but has since been used for view synthesis [8, 13] and general object recognition using more complicated features

[1, 14]. In addition, there is evidence to suggest that similar view-based approaches may be used by the human visual system for object recognition [2].

We begin with the fundamental principles of LCV theory. Consider a 3d scene with N points that belong to the same object. For simplicity, we think of bodies that follow the same 3d rigid motion as comprising a single object. Furthermore, we assume an affine camera projection model, where the 3d points are projected into F images (or frames). The relation between a 3d point with homogeneous coordinates $\mathbf{P}_j=[X_j,Y_j,Z_j,1]^T$, j=1,...,N and its projection to 2d image coordinates $\mathbf{p}_{ij}=[x^i_j,y^i_j]^T$, in image i= 1,...,F, is given by $\mathbf{p}_{ij}=\mathbf{m}_i\,\mathbf{P}_j$ where $\mathbf{m}_i$ is the 2×4 affine camera projection matrix related to image i. The set of N 2d points on the object, in image i, together define a *view* of the object:

$$\mathbf{v}_i = [\mathbf{p}_{i1}\,\mathbf{p}_{i2}\,...\,\mathbf{p}_{iN}] = \mathbf{m}_i\,[\mathbf{P}_1\,\mathbf{P}_2\,...\,\mathbf{P}_N] = \mathbf{m}_i\,\mathbf{S}, \quad i = 1,...,F. \tag{1}$$

In the following, we refer to the 4×N matrix $\mathbf{S}$ as the *shape matrix* of the object. Alternatively, we may also define the *trajectory* of a single 3d point j, as the collection of its 2d projections in all the images:

$$\mathbf{t}_j = \begin{bmatrix} \mathbf{p}_{1j} \\ \vdots \\ \mathbf{p}_{Fj} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_F \end{bmatrix} \mathbf{P}_j = \mathbf{M}\,\mathbf{P}_j, \quad j = 1,...,N. \tag{2}$$

We refer to the 2F×4 matrix $\mathbf{M}$ as the *motion matrix* (also known as the *joint projection matrix*). If we consolidate all the trajectories or all the views of all the points on the object, we obtain the combined *measurement matrix*:

$$\mathbf{W} = \begin{bmatrix} x^1_1 & x^1_2 & ... & x^1_N \\ y^1_1 & y^1_2 & ... & y^1_N \\ \vdots & \vdots & \ddots & \vdots \\ x^F_1 & x^F_2 & ... & x^F_N \\ y^F_1 & y^F_2 & ... & y^F_N \end{bmatrix} = [\mathbf{t}_1\,...\,\mathbf{t}_N] = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_F \end{bmatrix} = \mathbf{M}\,\mathbf{S}. \tag{3}$$

We can see that the 2F×N matrix $\mathbf{W}$ factors into the motion matrix $\mathbf{M}$ and the shape matrix $\mathbf{S}$. It follows then that rank($\mathbf{W}$)≤min (rank($\mathbf{M}$),rank($\mathbf{S}$))≤4. This rank constraint on $\mathbf{W}$ was independently observed by both Tomasi and Kanade [17] and by Ullman and Basri [20], but has been used in different ways.

2.1 The Tomasi-Kanade factorisation

Tomasi and Kanade [17] and later Costeira and Kanade [5] used singular value decomposition (SVD) to obtain a factorisation, similar to (3), of the 2d measurements into motion and shape. $\mathbf{W}$ can be decomposed into:

$$\mathbf{W} = \mathbf{U}_{[2F\times4]}\Sigma_{[4\times4]}\mathbf{V}^T_{[n\times4]} \;\Rightarrow\; \mathbf{M} = \mathbf{U}\Sigma^{1/2}, \quad \mathbf{S} = \Sigma^{1/2}\mathbf{V}^T \tag{4}$$

with Σ being the diagonal matrix of the 4 largest singular values of $\mathbf{W}$. This factorisation is not unique, and for 3d reconstruction we need additional rotation and translation constraints. However, for the purpose of motion segmentation this is not necessary. The columns of $\mathbf{U}$ form a set of basis vectors that span the $\mathbb{R}^{2F}$ space of $\mathbf{W}$, which is denoted as the *Joint Image*

Space (JIS) [15]. In fact, according to [15], each column in $\mathbf{W}$, is a point in the JIS and due to the rank 4 constraint, the entire trajectory space of the object (i.e. all the columns in $\mathbf{W}$) lie in a 4d linear subspace of the JIS. Furthermore, each point in the JIS, is linearly spanned by 4 other points and the coefficients of the linear combination are a function of the 3d shape alone. In other words, the JIS represents a connection between 2d and 3d, where the camera parameters have been eliminated.

The Tomasi-Kanade factorisation, provides a convenient basis onto the JIS, and has been used very effectively for motion segmentation. So for instance, assuming k rigidly moving objects, their trajectories will lie in a union of k 4d linear subspaces in $\mathbb{R}^{2F}$. Consequently, the motion segmentation problem simplifies to one of subspace clustering. The majority of recent motion segmentation approaches are essentially subspace methods that use this factorisation and cluster trajectories from 4d linear subspaces. Where they differ is on the way they define their clustering measure (e.g. sparsity, subspace distance, curvature etc).

2.2 Ullman and Basri - LCV

We can instead look at the row-space of $\mathbf{W}$, where each pair of rows represents the shape of the object in a single view. Each row of $\mathbf{W}$ (or "semi-view"), is a point in $\mathbb{R}^N$ space, denoted as the *Joint Point Space* (JPS) [15]. Again, due to the rank 4 constraint, all the semi-views of the object, occupy a 4d linear subspace in the JPS. Similar to the JIS, each point in the JPS is linearly spanned by 4 other points, and the coefficients of the linear combination are a function of the camera parameters alone. The JPS represents a connection between 2d and camera parameters, where the 3d shape has been eliminated.

We can see this given any 2 basis views, defined by the affine cameras $\mathbf{m}_1, \mathbf{m}_2$. In principle, it is possible to reconstruct the 3d shape $\mathbf{S}$ by:

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \end{bmatrix} \mathbf{S} = \mathbf{M}_{12}\,\mathbf{S} \Rightarrow \mathbf{S} = \mathbf{M}_{12}^{-1} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}. \tag{5}$$

Intuitively, we may think of this expression as the result of triangulating the 3d points in $\mathbf{S}$ given their projection in the two basis views based on the affine camera model. In the current application we are not interested in an explicit representation of $\mathbf{S}$, and instead use it to show that we can reconstruct a third view $\mathbf{v}_3$ from the two basis views as:

$$\mathbf{v}_3 = \mathbf{m}_3\,\mathbf{S} = \mathbf{m}_3\,\mathbf{M}_{12}^{-1} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}. \tag{6}$$

(6) is valid if $\mathbf{M}_{12}$ has full rank, but the general case can be managed if we instead write:

$$\mathbf{v}_3 = \mathbf{Q} \begin{bmatrix} \mathbf{1} \\ \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix}. \tag{7}$$

Consequently, the third view $\mathbf{v}_3$ is a linear combination of the two basis views, given by the elements of the 2×5 matrix $\mathbf{Q}$. The latter depends only on $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ and in a non-linear way. We can expand (7) to the familiar set of equations presented in [20]:

$$x_j^3 = a_0 + a_1 x_j^1 + a_2 y_j^1 + a_3 x_j^2 + a_4 y_j^2 \quad \text{and} \quad y_j^3 = b_0 + b_1 x_j^1 + b_2 y_j^1 + b_3 x_j^2 + b_4 y_j^2, \tag{8}$$

where (x_j^3, y_j^3) are the coordinates of the novel third view $\mathbf{v}_3$, and (x_j^1, y_j^1), (x_j^2, y_j^2) are the coordinates of the two basis views $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively. The expression in (8) is valid for

all the points $j \in N$ in the third view. In many practical applications, the matrices $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ are not known and, therefore, neither is $\mathbf{Q}$. From 5 or more corresponding points, visible in all the three views $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, however, the linear combination coefficients (a,b) can be estimated by solving the following linear system:

$$\begin{bmatrix} x_1^3, \ldots, x_r^3 \\ y_1^3, \ldots, y_r^3 \end{bmatrix} = \underbrace{\begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix} 1, \ldots, 1 \\ x_1^1, \ldots, x_r^1 \\ y_1^1, \ldots, y_r^1 \\ x_1^2, \ldots, x_r^2 \\ x_1^2, \ldots, x_r^2 \end{bmatrix}, \quad \text{where} \quad r \geq 5. \tag{9}$$

Correspondence is typically provided by a feature tracking algorithm (e.g. [9]). The (a,b) coefficients from (9) will be valid for all the points r in the solution set, and in the absence of measurement errors, they will also be valid for all the feature points on the same object.

Before we continue, it should be mentioned that although (7) is valid in the general case, there are degenerate cases when it is not. These are restricted to the case when the rigid motion between the two basis views are only in-plane rotations or only translations, and the third view is defined by a general motion $\mathbf{m}_3$, e.g., an out-of-plane rotation. However, as long as the third motion is restricted in the same way as the first two views, (7) is again valid. Furthermore, in the case that (7) is valid, there are multiple solutions for $\mathbf{Q}$. In fact, the solution space is an affine space of at least dimension 2. This implies that the least squares problem described above too has an affine solution space of at least dimension 2.

3 LCV for motion segmentation

Our proposal is based on a very simple principle: given a set of LCV coefficients, we can generate a synthetic trajectory of a point p_j and then compare this synthetic trajectory with the real trajectory of p_j we have tracked. If the similarity between the two is high, then this indicates that the particular set of LCV coefficients describes well the 3d motion of the point p_j. If it also happens that the LCV coefficients were estimated from a separate set of points c, then it is very likely that p_j and c lie on the same object (see Fig. 1).

Based on this idea, we propose a motion segmentation solution, which has 3 distinct steps. First is a motion hypothesis sampling step, where we generate a large number C of possible LCV coefficient matrices $\mathbf{Q}_i^c$, c=1,...,C, i=1,...,F. We may consider each individual set of $2\times5\times F$ coefficients as representing a (not necessarily unique) 3d motion hypothesis. In order to generate each $\mathbf{Q}_i^c$, we require a small number of corresponding points across three views (two basis views and a target view). Here we use 7 points, which keeps the size of the linear system in (9) manageable, while providing an accurate enough solution. One of course may use an arbitrary large point set to solve (9), but apart from the obvious practical problem of ensuring that the points lie on the same object, increasing their number will not result in any considerable improvement in accuracy (see Fig. 3(b)).

We begin by randomly sampling a large number C of initial points in a single frame. For each of these points, we take its 6-nearest neighbours (in Euclidean distance) in the same frame. Here, since the number of initial points C is large, it makes little practical difference which frame to use, so we always use the first frame in the sequence. Given these C 7-point clusters, we proceed by solving (9) for each cluster and at each frame, using always the first and last frames of the sequence as the fixed pair of basis views. In the end, we obtain a set

$\mathbf{Q}_i^c$ for every $c \in C$. Overall this is a very fast procedure, since we are only inverting a 5×7 matrix at each iteration.

The second step, involves generating synthetic trajectories for each scene point p_j and comparing them to the real trajectories. A synthetic trajectory is generated, one point per frame, using the LCV coefficients $\mathbf{Q}_i^c$ already calculated from each cluster c. We denote the synthetic trajectory of p_j as $\hat{t}_{j|c} = \left[\hat{p}_{1j}, ..., \hat{p}_{Fj}\right]^T$. A synthetic $\hat{t}_{j|c}$ and its real counterpart t_j are then compared, forming an entry in the $N\times C$ matrix:

$$\mathbf{E}(j,c) = K(\left\|t_j - \hat{t}_{j|c}\right\|_H / F), \tag{10}$$

where $\|.\|_H$ is the robust smooth Huber norm $\|x\|_H = \sqrt{1+x^2/\tau^2} - 1$, with τ being the switch threshold between the L_1 and L_2 norms. $K()$ is some kernel function that adjust the weighting of the error norm. The choice of K usually depends on the problem at hand, on the method used or is simply an arbitrary choice. Most motion segmentation methods in literature, use the generic Gaussian kernel. In our case, we have opted for the inverse multiquadric kernel:

$$K(x,\sigma) = \left(x^2 + \sigma^2\right)^{-1/2}, \tag{11}$$

which performs equally well to the Gaussian kernel, but has heavier tails for larger residuals and its parameter σ is easier to tune. Note here that (10) is a geometric error residual in image space, and we are using this rather than the algebraic residual from the solution of (9) since the former is more robust, and does not scale unpredictably with the number of points used to solve (9).

Equation (10) represents a similarity criterion (or affinity) between single points j and a 7-point cluster c (i.e. 7-tuple affinity). This is because the set of equations in (9) define a relationship between 5 or more points, but not just for two points. Our end goal here, is to obtain a relationship between two points at a time (pairwise affinity) that we can use for clustering. If we observe that the equations in (9) are independent of the ordering of the 7 points used, then according to the work by [7] on multi-way affinities, the full $N\times N$ pairwise affinity matrix can be approximated as $\mathbf{A} \approx \mathbf{E}\mathbf{E}^T$, up to some scalar multiplication.

We may now use the affinity matrix $\mathbf{A}$ in the third step of the algorithm and recover the final motion clusters using any standard clustering algorithm. We chose *spectral clustering* by [11], which is simple and it is known to perform well in practice, even when there is considerable noise in the affinity matrix (i.e. when some of the 7-point clusters do not lie on the same object or when there is substantial noise in the feature tracking). We have introduced a small modification to [11] that increases the speed of the algorithm. Instead of performing the eigen-decomposition on the Laplacian matrix $\mathbf{L}$, we solve the generalised eigen-problem $\mathbf{AV}=\mathbf{DVU}$ where $\mathbf{U},\mathbf{V}$ contain the k-largest eigenvalues and the corresponding eigenvectors respectively. $\mathbf{D}(j,j) = \sum_{j=1}^{N} \mathbf{A}(j,:)$ is the diagonal *degree* matrix of $\mathbf{A}$. Whilst the generalised eigen-problem is slightly more expensive than the standard eigen-problem, it avoids calculating $\mathbf{L}$ explicitly, a step which normally involves expensive matrix multiplications. As such, the generalised approach is computationally cheaper overall and it solves a "min-cut" criterion similar to [11]. The full motion segmentation scheme is shown in Algorithm 1.

3.1 On parameter setting

Our motion segmentation method has essentially two important parameters; the number of 7-point samples C and the spectral clustering parameter σ in (11). The other secondary

Algorithm 1 Motion segmentation

Input: samples C, motions k, measurement matrix $\mathbf{W}$, **Output:** $N\times 1$ label vector (k labels)

Select C random 7-point samples from $\mathbf{W}$ **//STEP 1**
For each 7-point cluster index $c \in C$ {
  Set basis views $\mathbf{v}_1 = p_{1c}$ and $\mathbf{v}_2 = p_{Fc}$
  For each frame index $i \in F$ {
    Pre-calculate LCV coefficients $\mathbf{Q}_i^c$ of $\mathbf{v}_3$ from $\mathbf{v}_1$ and $\mathbf{v}_2$ using (9) }}

For each point index $j \in N$ { **//STEP 2**
  Select basis views $\mathbf{v}_1 = p_{1j}$ and $\mathbf{v}_2 = p_{Fj}$
  For each 7-point cluster index $c \in C$ {
    For each frame index $i \in F$ {
      Synthesise target view $\mathbf{v}_i = p_{ij}$ from $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{Q}_i^c$ using (8)
      Set $\hat{t}_{j|c}$=[$\hat{t}_{j|c}$, $\hat{p}_{ij}$] }
    Calculate the affinity $\mathbf{E}(j,c)$ between $\hat{t}_{j|c}$ and t_j using (10) }}

Spectral clustering on $\mathbf{A} \approx \mathbf{E}\mathbf{E}^T$ with k clusters **//STEP 3**

parameters are fixed constants, such as the Huber norm τ=15 and the number of moving objects k, which is always assumed known. In terms of the samples C, we have experimented with different sizes and present their effects on accuracy and speed in the experiments section. A good rule of thumb, is to set C=100·k or alternatively one can ignore C as a tunable parameter and always set it to the maximum C=N. The most important however, is the σ parameter that determines the width of the kernel function and as a result, the weighting of the kernel mapping. Similar parameters and their optimal setting is a well known problem in spectral clustering research. The choice of σ can play an important role on the final segmentation results, depending of course on the sensitivity of the method and the difficulty of the problem. Some authors choose to manually tune σ for each problem, but this limits the practical applicability of the segmentation. Others set σ as some function of the data, while others automatically search for the optimal σ over a small range of possible values. We have chosen to follow the third approach, because we have found that it is more reliable in practice and allows the method to be applied to a greater selection of problems. The scheme for searching for the optimal σ is similar to the one proposed by [11]. We sample a few values of σ, say 10 uniform samples between $[10^{-1}, 10^{-4}]$ and perform spectral clustering with each σ sample. The optimal result, will be the one where the distortion between the clusters is at a minimum. There are many different ways to define cluster distortion or some generic quality measure. We simply use the k-means error, which is calculated for "free" during spectral clustering.

4 Experiments

In this section we evaluate the LCV motion segmentation method on real data from the Hopkins155 public dataset [19]. The Hopkins155 has been established as the standard evaluation dataset in motion segmentation research. It contains 155 rigid (general, articulated and degenerate) motion sequences of 2 and 3 objects, with automatically extracted trajectories that have been manually refined. Recently, the Hopkins155 has been extended with a few additional sequences that contain 4 and 5 rigid and nonrigid motions. For all the experiments that follow, the only parameter assumed known is the number of motions k. The rest are

automatically tuned as mentioned previously in Section 3.1.

We have tested our method (LCV) by performing 500 separate runs of the whole dataset (original 155 motions). The average misclassification errors for 2 and 3 motions are presented in Table 1. We have also included the best performing methods from segmentation literature (SCC-MS [8], SSC-N [6], SLBF-MS [25] and PAC [24]) for comparison, as well as the baseline RANSAC method that fits linear subspaces. The results for the other methods have been obtained from the recent work by [25] or from their respective publications. We can see that the LCV method performs very well, close to the state-of-the-art methods and it is actually better than the SCC-MS. On average, LCV is only around 0.5% worse than the best performingm method, which in practice amounts to only 1 additional misclassified point in a typical scene from the Hopkins155 with 250 points. Our method is also amongst the most stable, with low standard deviation over the 500 runs for 2 and 3 motions. The misclassification error CDFs are also included in Fig. 2. We see that our method gives some of the lowest errors for over 90% of the 2 motion sequences. It does contain however a few high misclassification errors for the remaining 10%, which slightly errode its overal score, but it is still a very strong performer overall. A similar behaviour can be observed for the 3 motions, with some reasonable and expected degradation. The same cannot be said for SCC-MS or RANSAC, which becomes almost random.

We also present the result from the extended dataset of 4 sequences with 4 and 5 motions and nonrigid objects in Table 2. We only show the results of our method, since this is a relatively new extension and the other methods have not published results yet. Although there are very few sequences available, we can still observe reasonable performance in par with the 2 and 3 motions previously. It is also very interesting to see that our method does not break down for instances of weakly nonrigid motions (3 objects).

Where the LCV method really excels however, is in the segmentation speed. We show the execution time (in seconds) from a single run on the full Hopkins155 set. The timings for SSC-N, SCC-MS, SLBF-MS and RANSAC are quoted from [25] and have been obtained on an Intel Core 2 CPU @ 2.66 GHz. PAC is quoted from [24] and has been obtained on an AMD Quad Core CPU @ 2.44 GHz. Our method has been evaluated on a slightly slower Intel Core 2 CPU @ 2.4 GHz. All methods use unoptimised Matlab code. The total and average times are displayed in Table 3. The fastest overall is RANSAC but with by far the highest error. The most accurate methods such as PAC, SSC-N and SLBF-MS are also the slowest, which limits their practical application. PAC for instance requires 41 hours to complete the whole database. Our method and SCC-MS have amongst the best combined speed and accuracy numbers, meaning that they can be used for motion segmentation in practice. However, ours is faster and also more accurate than SCC-MS. Together with RANSAC, LCV is the only other method that requires less than 1 second per sequence, on average.

We have also analysed the size of the 7-point clusters C and its effect on speed an accuracy (see Fig. 3(a)). As expected, a higher C means slower performance (blue curve, left axis) but also lower errors (green dashed curve, right axis). The indicated position on the figure, shows approximately where we expect to be if we set C=100·k, where k is the number of objects. In the same figure in (b), we show the average misclassification error over the whole dataset, as a function of the number of points we use to solve the linear system in (9). As we can see, 5 points are not enough to provide a robust solution. Conversely, if we use a large number of points, it becomes difficult to ensure that all points lie on the same object, which degrades the segmentation accuracy. Note, that the speed remains largely unchanged.

Method	SCC-MS[4]	SSC-N[6]	SLBF-MS[25]	PAC[24]	RANSAC	LCV
2 motions (120 sequences)						
Mean (500 runs)	1.77	1.00	0.98	0.96	5.56	1.25
std (500 runs)	0.25	0.00	0.00	0.00	n/a	0.07
3 motions (35 sequences)						
Mean (500 runs)	5.89	2.62	2.64	2.22	22.94	3.97
std (500 runs)	1.43	0.92	0.00	n/a	n/a	0.69

Table 1: Average classification errors (%) and their standard deviation over the 500 runs for Hopkins155 sequences. PAC is quoted only for a single run.

	4 motions (2 seq.)	5 motions (1 seq.)	Non-rigid motions (1 seq.)
Mean (500 runs)	7.98	0.00	0.00
std (500 runs)	2.31	0.00	0.03

Table 2: Average classification errors (%) and their standard deviation for extended Hopkins155 sequences. Results available only for the LCV method.

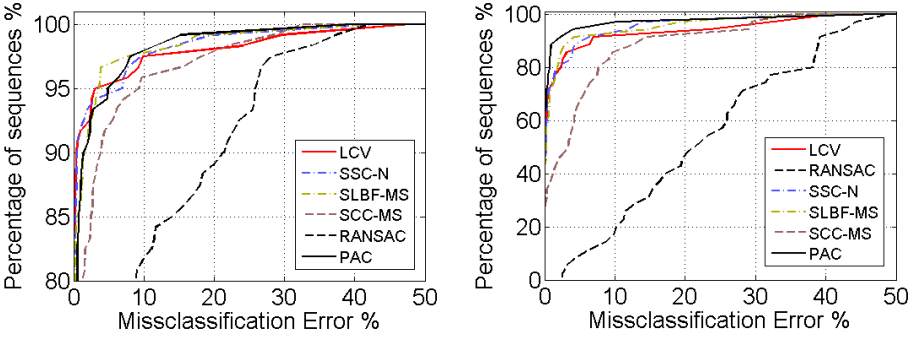
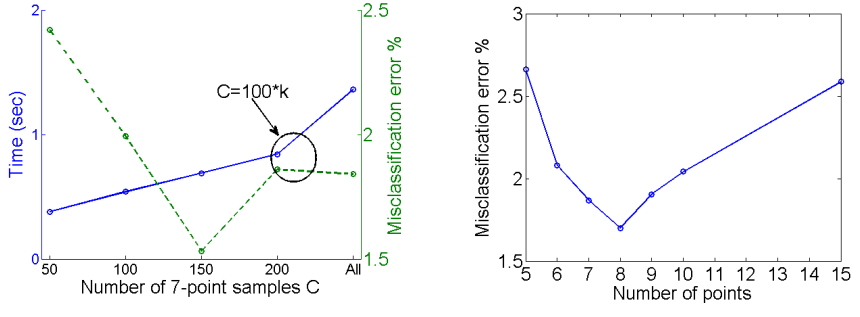


Figure 2: The misclassification error CDFs for 2 motions (left) and 3 motions (right) for the different methods. Note that for the 2 motions, the y-axis starts at 80%.



(a) (b)

Figure 3: Analysis on accuracy and speed for different numbers of clusters C (a) and different point sizes of each cluster (b).

Method	RANSAC	SCC-MS[4]	SLBF-MS[25]	SSC-N[6]	PAC[24]	LCV
Average time (sec)	0.387	1.264	10.83	165	952.25	0.93
Total time (sec)	60	196	1680	25620	147600	145
Average error (%)	9.48	2.70	1.35	1.36	1.24	1.86

Table 3: The average and total runtime for the different methods on the full Hokpins155 dataset (155 sequences).

5 Conclusion

We have presented a simple approach for the segmentation of an arbitrary number of 3d rigid motions from 2d images. Using the theory of linear combination of views, we generate synthetic trajectories of each point and compare them with the real, tracked ones. We have experimented on the Hopkins155 dataset and shown very good performance in both speed and accuracy. Furthermore, our method has a single parameter which is automatically tuned, making it an ideal method for real-time, practical application. The obvious future extension of this method, would be to cope with feature points with noisy or missing trajectories and the existence of outliers, that is trajectories that do not correspond to rigid 3d motions.

Acknowledgements

Funded by the EU FP7/2007-2013 programme, grant agreement No 247947 - GARNICS.

References

[1] G. Bebis, S. Louis, T. Varol, and A. Yfantis. Genetic object recognition using combinations of views. *IEEE Transactions on Evolutionary Computation*, 6(2):132–146, 2002.

[2] H. Bulthoff and S. Edelman. Psychophysical support for a two-dimensional view interpolation theory of object recognition. In *Proc. Natl. Acad. Sci.*, pages 60–64, USA, 1992.

[3] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *Int. J. Comput. Vision*, 81(3):317–330, 2009.

[4] G. Chen and G. Lerman. Motion Segmentation by SCC on the Hopkins 155 Database. In *ICCV*, 2009.

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

[6] E. Elhamifar and R. Vidal. Sparse Subspace Clustering. In *CVPR*, 2009.

[7] V. M. Govindu. A tensor decomposition for geometric grouping and segmentation. In *CVPR*, pages 1150–1157, 2005.

[8] M. E. Hansard and B. F. Buxton. Parametric view-synthesis. In *ECCV*, pages 191–202, 2000.

[9] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, 1981.

[10] Y. Ma, H. Derksen, W. Hong, and J. Wright. Segentation of multivariate mixed data via lossy codding and compression. *IEEE PAMI*, 29(9):1546–1562, 2007.

[11] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[12] K. Nordberg and V. Zografos. Multibody motion segmentation using the geometry of 6 points in 2d images. In *ICPR*, pages 1783–1787, 2010.

[13] G. Peters and C. von der Malsburg. View reconstruction by linear combination of sample views. In *BMVC*, number 1 in LNCS, pages 223–232, 2001.

[14] J. Revaud, G. Lavoue, Y. Ariki, and A. Baskurt. Fast and cheap object recognition by linear combination of views. In *ACM international conference on Image and video retrieval*, pages 194–201, 2007.

[15] A. Shashua. Trilinear tensor: The fundamental construct of multiple-view geometry and its applications. In *Algebraic Frames for the Perception-Action Cycle*, volume 1315 of *LNCS*, pages 190–206. Springer, 1997.

[16] Y. Sugaya and K. Kanatani. Geometric Structure of Degeneracy for Multi-body Motion Segentation. In *SMVC*, 2004.

[17] C. Tomasi and T. Kanade. Shape from motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137–154, 1992.

[18] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. R. Soc. Lond. A*, 356:1231–1340, 1998.

[19] P. Tron and R. Vidal. A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. In *CVPR*, 2007.

[20] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE PAMI*, 13(10):992 –1006, 1991.

[21] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. Segmentation of Dynamic Scenes from the Multibody Fundamental Matrix. In *ECCV Workshop on Vision Modeling of Dynamic Scenes*, 2002.

[22] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis. *IEEE PAMI*, 27(12), 2005.

[23] J. Yan and M. Pollefeys. A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Non-degenerate. In *ECCV*, 2006.

[24] L. Zappella, E. Provenzi, X. Lladó, and J. Salvi. Adaptive motion segmentation algorithm based on the principal angles configuration. In *ACCV*, volume 6494, pages 15–26, 2011.

[25] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Randomized hybrid linear modeling by local best-fit flats. In *CVPR*, pages 1927–1934, 2010.