J Intell Robot Syst
DOI 10.1007/s10846-015-0195-1

Using In-frame Shear Constraints for Monocular Motion Segmentation of Rigid Bodies

Siddharth Tourani · K Madhava Krishna

Received: 22 July 2014 / Accepted: 9 January 2015

Abstract It is a well known result in the vision literature that the motion of independently moving objects viewed by an affine camera lie on affine subspaces of dimension four or less. As a result a large number of the recently proposed motion segmentation algorithms model the problem as one of clustering the trajectory data to its corresponding affine subspace. While these algorithms are elegant in formulation and achieve near perfect results on benchmark datasets, they fail to address certain very key real-world challenges, including perspective effects and motion degeneracies. Within a robotics and autonomous vehicle setting, the relative configuration of the robot and moving object will frequently be degenerate leading to a failure of subspace clustering algorithms. On the other hand, while gestalt-inspired motion similarity algorithms have been used for motion segmentation, in the moving camera case, they tend to over-segment or under-segment the scene based on their parameter values. In this paper we present a principled approach that incorporates the strengths of both approaches into a cohesive motion segmentation algorithm capable of dealing with the degenerate cases, where camera motion follows that of the moving object. We first generate a set of prospective motion models for the various moving and stationary objects in the video sequence by a RANSAC-like procedure. Then, we incorporate affine and long-term gestalt-inspired motion similarity constraints, into a multi-label Markov Random Field (MRF). Its inference leads to an over-segmentation, where each label belongs to a particular moving object or the background. This is followed by a model selection step where we merge clusters based on a novel motion coherence constraint, we call **in-frame shear**, that tracks the in-frame change in orientation and distance between the clusters, leading to the final segmentation. This oversegmentation is deliberate and necessary, allowing us to assess the relative motion between the motion models which we believe to be essential in dealing with degenerate motion scenarios.We present results on the Hopkins-155 benchmark motion segmentation dataset (Tron and Vidal 2007), as well as several on-road scenes where camera and object motion are near identical. We show that our algorithm is competitive with the state-of-the-art algorithms on Tron and Vidal (2007) and exceeds them substantially on the more realistic on-road sequences.

Keywords Motion segmentation · Robot vision · Subspace clustering

S. Tourani (✉) · K. M. Krishna
Robotics Research Center, IIIT Hyderabad, Gachibowli, Hyderabad, 500032 Andhra Pradesh, India
e-mail: tourani.siddharth@gmail.com

K. M. Krishna
e-mail: mkrishna@iiit.ac.in

1 Introduction

With the imminence of mobile robots interacting with dynamic human environments, motion segmentation becomes a necessary function that robots should perform. For outdoor vehicles and robotic aids, it seems unavoidable that the camera mounted on the robot be mobile. Thus, there is a need for algorithms that are capable of handling situations where both the robot-mounted camera and object of interest are in motion. While there is already a vast amount of literature on motion segmentation, there are comparatively fewer algorithms that explicitly take into account the camera motion.

In video taken from a stationary camera, motion segmentation is straight forward. A background model can be learnt, and used to do a foreground-background segmentation leading to the moving foreground to be segmented out.

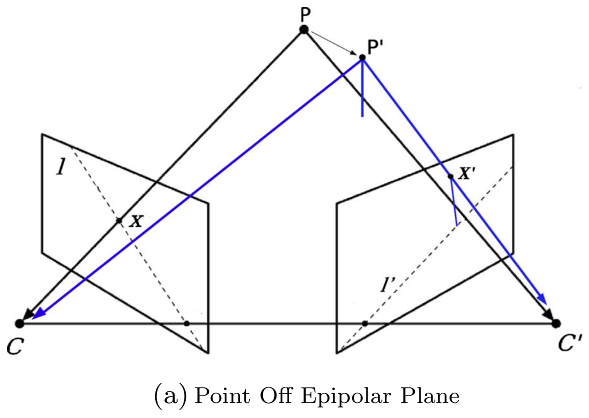
In the case of a moving camera, seperating stationary objects from non-stationary ones, is rather challenging, as the camera motion causes most of the pixels to move. The apparent motion of points, is a combined effect of camera motion, object depth and perspective effects and noise.

Optical flow vectors for nearby stationary objects may have a larger magnitude, than those for far-away objects that are in motion, thus one cannot exclusively use optical flow as the basis for a motion segmentation algorithm.

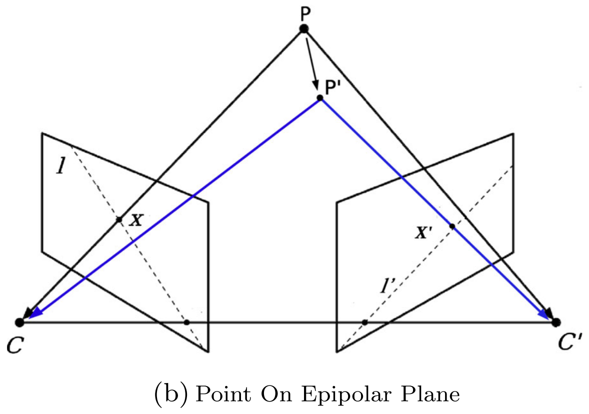
Likewise, the geometric constraints imposed by epipolar geometry [15], do not hold for certain relative motion configurations between the camera and the moving object, as shown below. The fundamental matrix relates two objects in a rigid scene, by the equation,

$$x'^T Fx = 0 \tag{1}$$

x and x' are 2D image co-ordinates of a 3D point viewed from two views. Geometrically Fx represents the epipolar line in l' in the primed co-ordinate system. If the 3D point P corresponding to x and x' are stationary, then in the case of zero-noise $|l'x'| = 0$ holds. This constraint, known as the epipolar constraint, can be used to distinguish between moving and stationary points, in most cases. Fig. 1a shows a typical case where the point P has moved to P' its projection in the second image x' lies above the line l'.



(a) Point Off Epipolar Plane



(b) Point On Epipolar Plane

Fig. 1 f the epipolar constraint functions and fails. In (**a**) the 3D point P on moving to P' off the epipolar plane is projected into the primed camera frame C' above the epipolar line l'. In (**b**) P' still lies on the epipolar plane and is projected right onto the l'. In (**b**) the epipolar constraint cannot be used to detect P' as moving

However, if P were moving in on the epipolar plane itself, as in Fig. 1b such that it's image lies on the epipolar line, yielding a zero value for epipolar constraint. Such a configuration is called a degenerate configuration. These degenerate configurations arise when the camera and object motion are either parallel or anti-parallel. The epipolar constraint fails for these degenerate scenarios. In the case of autonomous vehicles, one can argue that these situations, are more common than the non-degenerate case, which will occur mostly at road intersections.

In the design of a motion segmentation algorithm that works in degenerate scenarios, we have to incorporate other relative motion cues, that can capture the difference in motion between multiple objects. We use notions of motion similarity based on gestalt theory, temporal consistency and local spatial coherence, to design our motion segmentation algorithm.

- **Gestalt Theory:** Points tracked on an object should move similarly.
- **Temporal Consistency:** The above similarity should hold over time as well.
- **Local Spatial Coherence:** Points sampled from a small region should have similar motion.

The above principles are quite general, and do not make any strong assumptions about either the scene, camera model or motion between objects.

Our algorithm fits the template of multi-structure model fitting algorithms, such as those of [11, 12]. These algorithms have three steps. The first, is a model sampling step. In it sampling is used to generate a pre-define number of models. The second, is an energy minimization step, in which smoothness constraints are imposed on the sample points to get a spatially coherent labelling of the data. The third step, is a model reduction step that reduces the number of labels on the basis of some label cost criterion like Bayesian Information Criterion or Minimum Description Length.

In the first step of our algorithm, a fixed number of affine motion models are generated from randomly drawn, spatially-local subsets of the tracked points. We use a RANSAC-like procedure to estimate the motion models of the sampled points. As a result of this, some of the trajectory points are classified as model inliers. The remaining, unclassified, points are labelled on the basis of a trade-off between spatial proximity to a particular cluster (where the inliers of the model occur in the image) and model fit residual. Following this, an MRF energy minimization is carried out over the tracked points, incorporating the model-residuals as data terms. The smoothness terms penalize relative change in flow and motion, causing the resulting motion models to respect moving object boundaries. To obtain the final segmentation, we have a model selection step, that reduces the number of motion models to the number of moving objects in the scene, on the basis of a novel inter-model motion consistency constraint. This constraint captures the notion of in-frame shear between the models over-time as shown in Fig. 2. Figure 2a and b show the initial and final frames of the sequence. One can see that the motion model on the car (in red), can be thought of as shearing away from the one on the ground (in

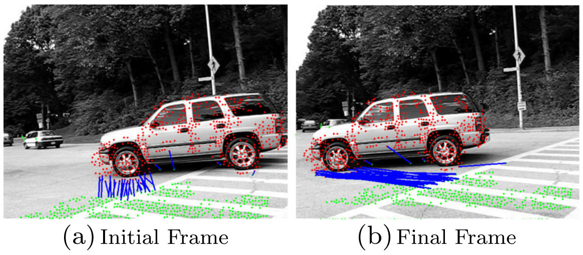


Fig. 2 Illustration of In-Frame Shear. In Fig. (**a**), the blue lines between the red and green clusters are close together. By frame (**b**), they have changed orientation by nearly 90 degrees, and are considerably stretched

green). As this is plainly apparent from the sequence of images and does not take into account any extraneous 3D information, we prefix the phrase "in-frame" to shear.

Using these principles and constraints we propose an algorithm that out-performs state-of-the art algorithms on sequences taken from publicly available datasets [17] and [21]. An additional advantage of our algorithm is it does not require the number of moving objects in the scene nor the dimensions of the subspaces in which they lie to be manually given, as is the case of most of the other algorithms we have compared the performance with like [3, 33, 43].

To sum up, in this paper, we propose a new rigid-body motion segmentation algorithm that

- Has no prior requirement of the number of motions in the scene
- No requirement of the dimensionality of the motion subspaces
- Works on commonly encountered outdoor scenes, which are degenerate and outperforms all other known methods
- Works on sequences that have significant camera motion
- Bypasses any need for camera-motion estimation using shear constraints

2 Related Works

Motion Segmentation is a well-studied problem with algorithms inspired from various sources like, matrix factorization [4, 23, 40], tensor decomposition [3, 26, 43], statistical model estimation [7, 35, 39] and perception [29, 34, 38]. The problem has been analyzed for affine [3, 5, 6, 23], perspective [36, 40, 41] and even catadioptric camera models [42]. We however found that, while these approaches do tend to do well on benchmark datasets, they are not robust enough to handle real on-road scenarios. Instead the systems that give reasonable real-world performance [1, 32], tend to rely more heavily on optical flow differencing, and multi-view geometric constraints, as they are computationally-inexpensive. However, flow differencing will only work when there is a reasonable difference between the movement of the camera and that of the moving object. Likewise, the epipolar constraint, tends to fail in the degenerate cases where the camera is following the object. We present a survey of what we consider to be the three main approaches to motion segmentation. The approaches are summarized in Table 1.

2.1 Algorithms based on Perspective Geometric Constraints (Epipolar Geometry and Planar-Parallax Decomposition)

In [41], Vidal et. al studied the case of detecting multiple motion, between two perspective views, by extending the epipolar constraint and its corresponding fundamental matrix to the multi-body setting [40]. The method simultaneously recovers multiple fundamental matrices by polynomial factorization. As mentioned before, these algorithms based on epipolar geometry are unable to handle the case of degenerate motion.

In the robotics context, [1, 20, 24], are all able to handle degenerate scenarios, using the Flow-Vector Bound constraint which is able to model degeneracy with precision. However, in all of these cases, an accurate estimate of the camera translation in the world frame, as well as depth in the local frame. As, [20] is a stereo based algorithm, the estimate of the camera translation can be obtained accurately, however, depth is still an issue beyond distances of a few metres. In the monocular case, obtaining an accurate estimate of either depth or translation is much harder. In [1], the camera translation is found through a computationally intensive, unstable V-SLAM system running as a back-end to the motion detection. In [24], translation is found out through an odometry sensor. In comparison, our paper bypasses the need for any robust pose estimation

Table 1 Summary of the recent motion segmentation algorithms (along with their attributes) relevant to our algorithm

Method type	Algorithm	Handles degeneracy Y/N	Prior knowledge
Epipolar Geometry	Kundu et al. [24]	Y	EM
	Vidal et al. [41]	N	–
Epipolar Geometry + Optical Flow	Namdev et al.[1]	Y	EM
	Romero-Cano et al. [20]	Y	EM
Optical Flow	Brox et al. (2010) [22]]	N	NM
	Ochs et al. (HOSC) [49]	N	–
	Lezama et al. [2]	N	NM
Affine Subspace Clustering	Elhamifar et al. (SSC) [3]]	N	NM+DS
	Chen et al. (SCC) [43]	Y	DS+NM
	Zografos et al. (LCV) [33]	N	NM
	Jain et al. (SGC) [26]	N	NM
	Zapella et al. (ASA) [46]	Y	NM
	Zapella et al. (ELSA) [48]	N	–
	Chin et el. (ORK) [44]	N	–
	RANSAC [27]	N	DS
OF+Subspace Clustering + Shear	Ours	Y	–

Legend: EM: Ego-Motion, **NM:** Number of Motions, **DS:** Dimensions of Subspaces,**OF**: Optical flow of the camera by making use of in-frame gestalt cues.

To overcome the limitation of the epipolar constraint, vision researchers explored enforcing multi-view constraint. The most prominent of this being the trifocal tensor [15]. Generally it was observed that for small-camera motions, such as those arising from footage taken from a video camera, the trilinear constraint [15] was ill-conditioned and extremely sensitive to noise.

An alternate line of research explored enforcing multi-view rigidity constraints using the so-called "*planar-parallax*" decomposition. A scene is modelled as points belonging to a 3D planar surface in the scene (constituting the planar part) and the plane outliers(constituting the parallax). This decomposition allowed for the introduction of new multi-view constraints like the structural-consitency constraint [50], the parallax flow field [52] and the homography tensor [53]. While the "structural consistency" constraint is able to handle degenerate motions atleast theoretically, it remains untested in uncontrolled outdoor settings.

2.2 Algorithms based on Affine Camera Models (Subspace Clustering)

Under the affine projection model, point trajectories associated with each moving object across multiple frames lie on affine subspaces of dimension of 4 or less. Therefore, motion segmentation, can be achieved by clustering point trajectories into different motion subspaces. This has led to several affine subspace clustering algorithms being developed [3, 26, 43, 44] which achieve near perfect results on the Hopkins-155 dataset [27]. However, when we ran, these algorithms on our datasets we frequently found that the moving objects were clustered with the background. The reason for this inconsistency are two-fold. Firstly, the clips in the Hopkins- 155 dataset have very little perspective effects and thus the affine camera model assumption is reasonable. Secondly, in the degenerate case, both the camera and the moving object(s) will belong to the same subspace, leading to subspace clustering algorithms labeling the background and the object as the same. Another major drawback of the subspace clustering, is that most of them need to be given the affine dimensions of the subspaces on which the points are supposed to lie. Even the ones that are able to do so are not robust, as will be shown in the results section.

2.3 Clustering Based on Trajectory Similarity (Optical Flow)

In [14, 22, 29], impressive results were shown by grouping point trajectories, that were analyzed for motion differences between pairs of trajectories. Similar techniques were used by [2] with additional constraints for occlusion modeling. These algorithms are based on gestalt [18] theory and while not relying on geometry, still achieve state-of-the-art results on several challenging datasets. This indicates that motion similarity is quite a powerful constraint for motion segmentation.

Each of these approaches has it's strengths and limitations as mentioned above. Our algorithm can be thought of as a cohesive and judicious combination of the above approaches, yielding a motion segmentation algorithm that gives state-of-the-art accuracy on both the benchmark and exceeds in real-world scenarios. In addition, in Section 4 we introduce a motion coherence constraint on the basis of which we can reason about motion models calculated from the scene.

3 Proposed Approach

Our proposed motion segmentation algorithm has three stages. First, we do a coarse foreground-background segmentation, based only on the epipolar constraint. Based, on this segmentation, we generate a pool of M affine motion models, $\mathcal{M} = \{M_1, ..., M_M\}$, by a RANSAC [30] procedure over the entire sequence of frames instead of just between two frames, allowing for a motion model hypothesis that is representative of the sample points over the entire sequence. The goal of this step is to generate one motion model for each of the N-independently moving objects in the scene. However as we have no prior knowledge as to the location of the models in the sequence, $M \gg N$ motion models are instantiated in an attempt to increase the likelihood of capturing the correct N-motion models.

The foreground-background segmentation allows for adequate sampling of both foreground and background regions by building affine models from points belonging primarily to either the foreground or the background. As generally a large portion of the tracked points belong to the static background, our sampling strategy makes it more likely that the moving objects which in most cases will belong to the foreground are also sampled.

The unsampled points, which thus far remain unclassified, are then clustered with one of the instantiated M motion models, based on a trade off-between, motion model prediction accuracy and spatial proximity to the cluster center.

In the second step, to refine the segmentation we perform a multi-label MRF minimization, incorporating the motion model likelihood in the data term along with pairwise motion similarity and attribute constraints (whether the model belongs to the background or foreground), for a motion-coherent over-segmentation of the trajectory data. As the sampling was done at random on the image in the previous step, the motion models generated do not obey object boundary constraints. So, incorporating these long-term motion-similarity constraints gives an over-segmentation of the scene that respects the boundaries of the moving object. Usually, as a result of the energy minimization, some of the tracked points are reallocated to different motion models resulting in a reduction of the number of models required to explain the trajectories. We call this reduced model set $\mathcal{M}_{red}$.

The over-segmentation in the previous step is desirable and even-necessary to give a final segmentation without having prior knowledge of the number of moving objects in the scene. In the final stage model selection phase, the motion models from $\mathcal{M}_{red}$ are merged to complete the segmentation of the scene into individual moving parts. To decide whether two models should be merged or not, we introduce a novel motion consistency predicate, we termed in-frame shear, that should be satisfied between the two models in order for model merging to take place. The end result is the desired segmentation into scene and individual moving objects. We call the final model set $\mathcal{M}^*$.

Figure 3 shows an overview of the proposed algorithm.

We introduce here the trajectory matrix, which we use subsequently. $\mathbf{W} \in R^{2F\times P}$ is the trajectory matrix which contains the $\mathbf{x}$ and $\mathbf{y}$ image coordinates of the P feature points tracked through F frames. The algorithm terminates when, the trajectories from $\mathbf{W}$, are split amongst $\mathcal{M}^* = \{M_1^*, \ldots, M^*_{|\mathcal{M}^*|}\}$ models.

$$W = \begin{bmatrix} x_{1,1} \ldots x_{1,P} \\ y_{1,1} \ldots y_{1,P} \\ \vdots \ddots \vdots \\ x_{F,1} \ldots x_{F,P} \\ y_{F,1} \ldots y_{F,P} \end{bmatrix} \tag{2}$$

For tracking we use the KLT-Tracking algorithm [45].

3.1 Initial Foreground-Background Segmentation

It has been shown in [25], [13] that an initial foreground-background (**fg-bg**) segmentation improves the accuracy of motion segmentation

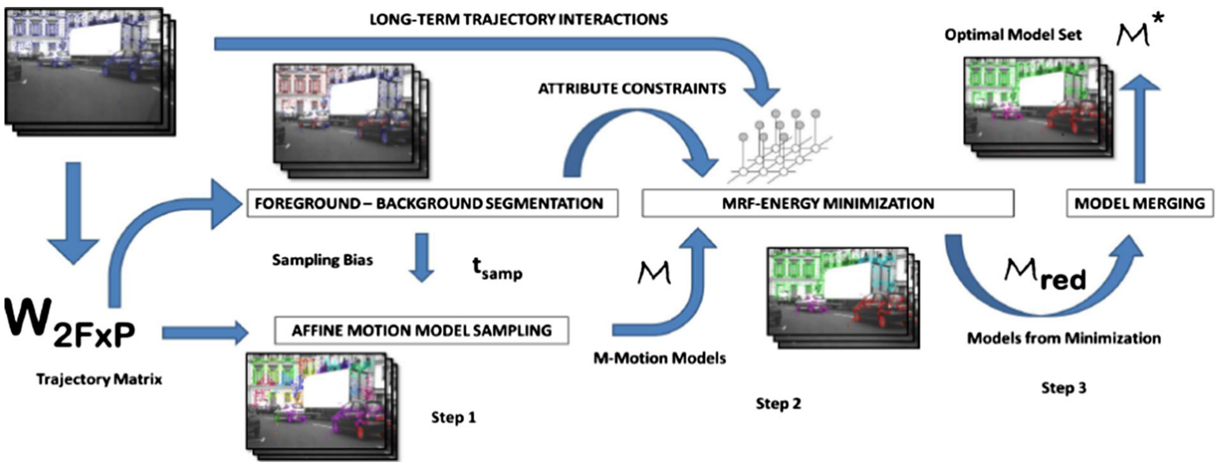


Fig. 3 An overview of the proposed approach

algorithms. In both, [10] and [13], image saliency is used to perform the initial segmentation. We instead rely on the epipolar constraint [15], to provide us with a conservative background estimate.

Between a base frame (in our case always the first frame), and all other frames we find the inliers. Only those inliers which belong to atleast **50 %** of the frame pairs, are classified as background. The remainder are classified as foreground. The outputs of this stage are the foreground and background index $\mathbf{I}_{fg}$ and $\mathbf{I}_{bg}$. The RANSAC threshold value **t**, is deliberately set very low to have a conservative estimate of the background. The fg-bg segmentation also aids in the biased fg-bg sampling as mentioned earlier. We use the least medians distance, for our epipolar constraint. Figure 4a shows the result of an initial fg-bg segmentation. Figure 4b shows the degenerate case where the object and camera move in parallel. In this case the epipolar constraint ends up failing.

3.2 Biased Affine Motion Model Sampling

From the fg-bg segmentation, we can partition the trajectory matrix into W_{bg} and W_{fg} , where W_{fg} contains only the foreground points and W_{bg} contains only the background points. Sampling separately from the foreground and background makes it more likely that the samples extracted from the trajectory data, will accurately represent the N-independently moving objects in the scene. This covers the case, where only few points are tracked on a foreground object, as frequently happens on non-textured surfaces. It is intuitively obvious that, most of the points sampled from

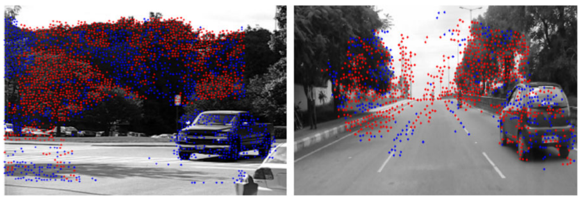

(a) Non-Degenerate (b) Degenerate

Fig. 4 Figure shows a result of the initial foreground-background segmentation. The foreground (epipolar outliers) are shown in *blue*, and background (epipolar inliers) are shown in *red*. **a** In the non-degenerate case most, of the points on the moving vehicle have been categorized as not belonging to the background. **b** In the degenerate case, most of the points on the vehicle belong to the background

a small region, will belong to the same object. Keeping this idea of local coherence, in mind, we randomly sample disk-shaped regions, of a fixed small radius, as the scale of the object in question is a-priori unknown. This region serves as the support set for our affine model and is defined by

$$\hat{\mathbf{W}} = \left\{ w_i | (x_{1,i} - c_x)^2 + (y_{1,i} - c_y)^2 < r^2 \right\} \tag{3}$$

Here, w_i is the i^{th} column of the trajectory matrix, $x_{1,i}$ and $y_{1,i}$ are the x and y - image coordinates of the ith trajectory in the first frame. (c_x, c_y) are the co-ordinates of the center of the disk, and r its radius. To promote a more comprehensive coverage of the scene, the number of times (c_x, c_y) is sampled from the trajectories of W_{fg} to the number of times from W_{bg} is governed by a parameter t_{samp} . The details are shown in Algorithm 1.

To compute our affine model $\mathbf{A}$, we need three trajectories $\mathbf{C}_{pts} = [\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3]$ extracted from $\mathbf{W}$. The affine motion model between frame 1 and f can then be computed by,

$$\mathbf{A}^{1\rightarrow f} = \begin{bmatrix} \mathbf{c}_1^f & \mathbf{c}_2^f & \mathbf{c}_3^f \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{c}_1^1 & \mathbf{c}_2^1 & \mathbf{c}_3^1 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \tag{4}$$

Here, $\mathbf{c_i^f} = [x_{f,i} y_{f,i}]$, are the x and y co-ordinates of the i^{th} control trajectory at frame f. The inverse of the right hand side matrix exists, except in the case where the 3 points are collinear.

Like in Section 3.1, this affine model is computed between the first and subsequent frames, and the inliers for each affine model are computed based on a threshold t_{in} . Only those points which are inliers for atleast **50 %** of the frames are considered model inliers.

The output of the affine sampling algorithm is the motion model tuple $M = \{I_{in}, C_{pts}, S\}$. I_{in} contains the indices of the model inliers. C_{pts} are the image co-ordinates of the 3 control points in the first frame. S gives the attribute of the model, i.e., whether it is a foreground or background model. It should be noted that we do not make any use of the affine models computed in the algorithm subsequently, instead, we use a more robust model-fitting measure described in the subsequent sub-section. We found that keeping the number of RANSAC iterations as low as 10 had no overall reduction in accuracy of our algorithm, while speeding it up somewhat.

Algorithm 1 Biased FG-BG Affine Model Sampling

```
Data: $\mathbf{W}$, $\mathbf{I}_{fg}$, $\mathbf{I}_{bg}$, $\mathbf{t}_{samp}$
Result: Motion Model $\mathbf{M}$
/* Extract disk centre for the affine model */
if $rand(0,1) < t_{samp}$ then
    $c_i \leftarrow rand(1, I_{fg})$        /* Control Pt from fg */
    S={$fg$}
else
    $c_i \leftarrow rand(1, I_{bg})$        /* Control Pt from bg */
    S={$bg$}
end
$\hat{\mathbf{W}}_{2F\times I} \leftarrow$ sampleTrajectoriesFromDisk($W, r, c_i$)
$\mathbf{W}_1 = \mathbf{W}(1:2,:)$                 /* $1^{st}$ frame pts */
$\mathbf{X} \leftarrow$ homogenize($W_1$)
$\mathbf{C}_{best} = \emptyset$
$\mathbf{I}_{best} = \emptyset$
for $i : 1$ to 10 do
    $c \leftarrow rand(3, I)$
    $\mathbf{C}_{pts} = \mathbf{W}_1(:, c)$
    for $f : 2 \rightarrow F$ do
        $\mathbf{W}_f = \mathbf{W}(2f\text{-}1:2f,:)$    /* $f^{th}$ frame pts */
        $\mathbf{Y} \leftarrow$ homogenize($W_f$)
        $\mathbf{A} \leftarrow \mathbf{YX}^{-1}$
        $\mathbf{R} \leftarrow \mathbf{AX\text{-}Y}$
        $\mathbf{I}_f \leftarrow$ computeInliers($\mathbf{R}, \mathbf{t}_{in}$)
    end
    $\mathbf{I} = \{\mathbf{I}_2, \ldots, \mathbf{I}_F\}$
    $\mathbf{I}_{in} \leftarrow$ points from $\mathbf{I}$ with more than 50 % inliers
    // If current model has more inliers, update
    if $|I_{in}| > |I_{best}|$ then
        $\mathbf{I}_{best} = \mathbf{I}_{in}$
        $\mathbf{C}_{best} = \mathbf{C}_{pts}$
    end
    $\mathbf{M} = \{\mathbf{I}_{best}, \mathbf{C}_{best}, \mathbf{S}\}$    // Output motion model
end
```

3.3 Initial Assignment of the Unsampled Points

To deal with points that remain unsampled after the motion model generation, it would make sense to classify the point as belonging to the affine model that explains it the best, i.e, the one with which the trajectory has minimum residual. This is the most common strategy and is useful in situations where the models are distinctive, and has been used in [5] and [11] successfully. However, in the case, where the models are not distinctive and explain similar motions, this may lead to a spatially incoherent labeling. This is because, the residual value would be low and very similar for more than one models. So in addition to a low model residual we also impose a spatial constraint on our assignment.

For each unsampled trajectory w_i , we compute its residual set $r_i = \left\{r_1^i, \ldots, r_M^i\right\}$ from the M affine motion model hypothesis. We sort the elements in r_i to obtain the sorted residual set $\tilde{r}_i$. For the models that yielded the top k-elements r of this set, we calculate Euclidean distance from the image co-ordinates of w_i to the control points of the model to the in the first frame. The trajectory point is alloted to the model with minimum distance.

The model residual is the orthogonal distance to the hypothesis subspace, given by,

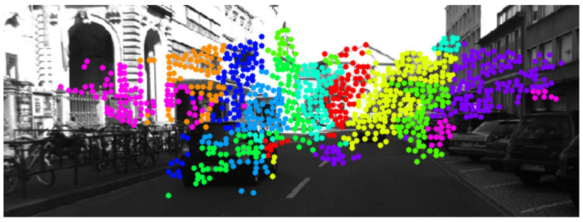
$$\mathbf{r_m^i} = |\mathbf{UU^Tw\text{-}w}| \qquad (5)$$

$\mathbf{U}$ is the first two left-singular vectors of matrix $\mathbf{W_m}$, which contains the inlier trajectories of the $\mathbf{m}^{th}$ motion model. We use the orthogonal distance as error metric, instead of making direct use the calculated affine motion model matrices ($\mathbf{A}$ matrices) because it is more robust to noise. However, it is still sensitive to outliers.

Figure 5 shows the contrast between a minimum residual approach versus top-k minimum residuals approach. As can be seen clearly, spatial coherence is much better in the latter approach. The top-k residual idea was first introduced in [19].

3.4 Segmentation Refinement by Energy-Minimization

As mentioned earlier, and shown in Fig. 5a, the sampling from the previous step, is done on the image space and does not take into account, any motion information. Thus, the generated models do not obey


(a) Minimum Residual

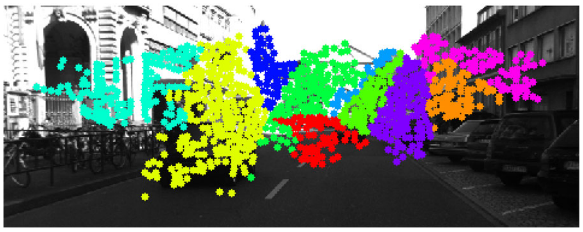

(b) Top-k Residual

Fig. 5 Difference between minimum residual and Top-k residual sampling

the boundaries of moving objects. So, in order to obtain a segmentation that respects boundaries of moving objects we incorporate long-range motion similarity constraints, as well as fg-bg attribute constraints to get a more refined segmentation, that obeys object boundaries. At the same time gets rid of some redundancy in the motion models sampled. To perform the refinement, we frame this trajectory-level motion segmentation as an MRF energy-minimization over the M-motion model label space. Each node in the graph represents one of the P tracked points. The edges in the graph are obtained by performing a Delaunay Triangulation (DT) over the tracked points in the first frame of the sequence. The energy we minimize is,

$$\begin{aligned} E(W, \mathcal{M}) = \lambda_1 \sum_{p \in G} D_{aff}(l_p) \\ + \lambda_2 \sum_{p \in G} \sum_{q \in N(p)} V_{motion}(l_p, l_q) \\ + \lambda_3 \sum_{p \in G} \sum_{q \in N(p)} V_{attr}(l_p, l_q) \end{aligned} \quad (6)$$

where, $\{l_p\} \in \{1, \ldots, M\}$. The $D_p(.)$ terms are the unary potentials representing the data terms, and the $V_{pq}(., .)$ are the pairwise potentials, representing smoothness constraints. G represents the set of tracked points in the sequence, $N(p)$ the spatial neighborhood of the point of the node p as defined by the Delaunay Triangulation (DT) of G. The λ's are trade-off parameters between the various terms. We now define in detail each of the terms in (6).

Data Term: The first term in (6) takes into account the attribute of the graph node. It is given by

The data term indicates how well a particular point fits a particular model. It is given by,

$$D_{aff} = \begin{cases} r_i^p & \text{if } Attr(l_p) = Attr(p) \\ \sigma_{out} & \text{if } Attr(l_p) \neq Attr(p) \end{cases} \quad (7)$$

r_i^p is given by (5).

Smoothness Terms: The first smoothness term captures motion coherence. To assign similar tracks to the same clusters, we use the contrast-sensitive Potts model given below,

$$V_{motion}(l_p, l_q) = \begin{cases} 0 & \text{if } l_p = l_q \\ \alpha_{p,q} & \text{if } l_p \neq l_q \end{cases} \quad (8)$$

The $\alpha_{p,q}$ measures the similarity between the two tracks. $\alpha_{p,q}$ is computed using the distance between the spatial co-ordinates of track points, w_p and w_q for each of the frames, as well as the difference between the time corresponding velocities v_p and v_q as below,

$$\alpha_{p,q} = exp\left(-\frac{(1 + ||x_p - x_q||_2)^2 ||v_p - v_q||_2^2}{2\sigma^2}\right) \quad (9)$$

Such motion potentials have been shown to be effective in ensuring motion coherence between tracks in [2, 29].

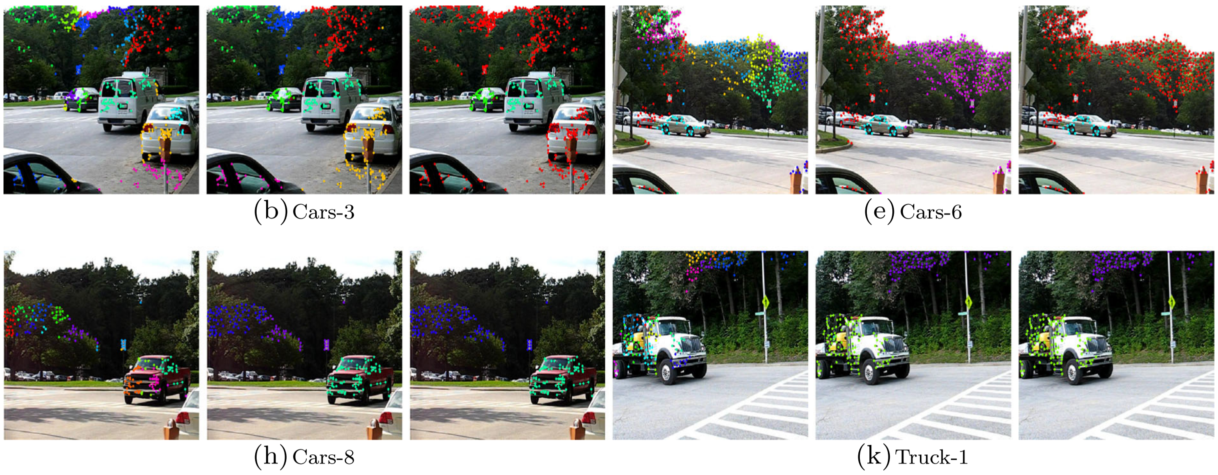


Fig. 6 Results from the Hopkins-155 dataset.The various stages of our proposed approach are shown. For the truck 1 case, only the first two steps suffice in obtaining the motion segmentation

The second smoothness term in (6) captures attribute coherence,

$$V_{attr}(l_p, l_q) = \begin{cases} 0 & \text{if } Attr(l_p) = Attr(l_q) \\ E_c & \text{if } Attr(l_p) \neq Attr(l_q) \end{cases} \tag{10}$$

Similar attribute terms were shown to be effect in object tracking in [25]. Figure 6 show the results of the various stages of the thus far pipeline. As can be seen, we get an over-segmentation of $\mathcal{M}_{red}$ models. For inference, we use the Tree Re-weighted Sequential Belief Propagation Algorithm [16].

4 Model Selection

The model-selection step is a model-merging algorithm that takes as input the set of models $\mathcal{M}_{red}$ and outputs the reduced set $\mathcal{M}^*$. For a particular pair of neighboring models, we merge if for the model points, the model-merging predicate, Algorithm [2], is satisfied.

Initially Algorithm [3] starts of with, $|\mathcal{M}_{red}|$ distinct models. This is represented by the $|\mathcal{M}_{red}| \times |\mathcal{M}_{red}|$ model-relationship matrix B, which will contain either, -1, 0 or 1 as entries. This matrix encodes whether or not two-clusters should be merged, as follows

- if $B_{i,j} = 0$, relation between i and j still unknown.
- if $B_{i,j} = -1$, i and j belong to separate objects.
- if $B_{i,j} = 1$, i and j belong to same object.

The algorithm begins with B having 1 along the diagonals and 0 everywhere else, indicating that each cluster is compatible with itself, and the other relations are unknown. The algorithm terminates when all the relations are known, i.e., B has no elements as 0. For each iteration, the closest pair of clusters whose relationship is unknown are chosen, a greedy minimum distance-based assignment is done on a certain prefixed number of points between both models and given to [2], which gives a decision as to whether or not to merge the pair. We also make use of transitivity in our algorithm. If a relationship holds between clusters a and b and not between a and c then the relationship does hold between a and c. This in addition to enforcing consistency, also speeds up the algorithm. The distance between two clusters is taken as the distance between the centroids of their image co-ordinates in the first frame.

Algorithm 2 shearCheck

```
Data: $M_1, M_2$
Result: Boolean {0 or 1}
$\mathbf{DT}_{M_1} \leftarrow$ getDelaunayTriangulation($\mathbf{M}_1(1:2,:)$)
$\mathbf{DT}_{M_2} \leftarrow$ getDelaunayTriangulation($\mathbf{M}_2(1:2,:)$)
// Get shear measure within models
$\mathbf{shear}_{M_1} = 0$
$\mathbf{stretch}_{M_1} = 0$
foreach edge e in $\boldsymbol{DT}_{M_1}$ do
    $\Delta_{shear} = \sum_{f=2}^{F}$ abs(changeInSlope($e_{f-1}, e_f$))
    $\Delta_{stretch} = \sum_{f=2}^{F}$ abs(changeInLength($e_{f-1}, e_f$))
    $\mathbf{shear}_{M_1} = \mathbf{shear}_{M_1} + \Delta_{shear}$
    $\mathbf{stretch}_{M_1} = \mathbf{stretch}_{M_1} + \Delta_{stretch}$
end
$\mathbf{score}_{M_1} = \mathbf{shear}_{M_1} + \mathbf{stretch}_{M_1}$
$\mathbf{score}_{M_1} = \mathbf{score}_{M_1} / ((F-1) \times \text{numEdges}(\mathbf{DT}_{M_1}))$
Likewise from $\mathbf{DT}_{M_2}$ we get $\mathbf{score}_{M_2}$
// Get shear between models
$\mathbf{E}_{M_1 \to M_2} \leftarrow$ greedyAssignment($M_1, M_2$)
$\mathbf{shear}_{M_1 \to M_2} = 0$
$\mathbf{stretch}_{M_1 \to M_2} = 0$
foreach edge e in $\boldsymbol{E}_{M_1 \to M_2}$ do
    $\Delta_{shear} = \sum_{f=2}^{F}$ abs(changeInSlope($e_{f-1}, e_f$))
    $\Delta_{stretch} = \sum_{f=2}^{F}$ abs(changeInLength($e_{f-1}, e_f$))
    $\mathbf{shear}_{M_1 \to M_2} = \mathbf{shear}_{M_1 \to M_2} + \Delta_{shear}$
    $\mathbf{stretch}_{M_1 \to M_2} = \mathbf{stretch}_{M_1 \to M_2} + \Delta_{stretch}$
end
$\mathbf{score}_{M_1 \to M_2} = \mathbf{shear}_{M_1 \to M_2} + \mathbf{stretch}_{M_1 \to M_2}$
if $score_{M_1 \to M_2} > \lambda \frac{score_{M_1} + score_{M_2}}{2}$ then
    1                          // Models are seperate
else
    0                          // Models to be merged
end
```

Algorithm 3 The Model Merging Algorithm

```
Data: $\mathbf{M}_i, i \in \{1, \ldots, |\mathcal{M}_{red}|\}$ // Reduced Model-set
Result: $\mathcal{M}^* = \{M_1^*, \ldots, M_n^*\}$ // Merged Models
B = $\begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } i \neq j \end{cases}$     // Model-Relationship Matrix
while $\exists B_{i,j} = -1$ do
    $(i, j) \rightarrow$ nearestPairOfUnassignedNeighbors(B)
    $B_{i,j} \rightarrow$ shearCheck($M_i, M_j$)
    Assign consistent relations between i and j
Merge consistent indices based on B
```

4.1 Model Merging Predicate

In the case of a moving camera, motion segmentation is like separating out two convolved signals from each other, which is an innately hard problem. So, in an attempt to bypass the complication induced by a moving camera, we look at relative geometric relationships in-frame that change over-time to detect motion. For example, in Fig. 7a the cluster in green is initially

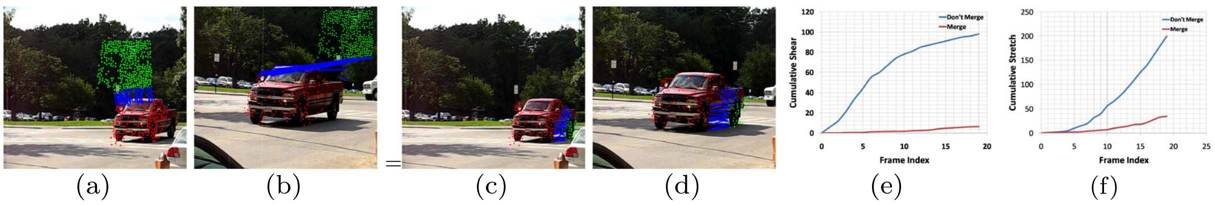


Fig. 7 Illustration of how shear and stretch work. In (**a**) and (**b**) the initial and final frames are shown along with two motion models, belonging to seperate objects. The *blue lines* between the two clusters change in orientation and length from frame (**a**) to frame (**b**). In (**c**) and (**d**), the two motion models belong to the same object. As can been seen, there is not much change in orientation and length, between the two clusters. In (**e**) and (**f**) the cumulative shear and stretch are plotted for both cases, where the models are merged and marked as seperate.As can be clearly seen, the "Don't Merge" case dominates in both situations, showing that shear is a reliable criterion on the basis of which motion segmentation can be performed.

close to the vehicle in the world (cluster in red). The lines between the two initially have a predominantly downward orientation. By the last frame, the predominant orientation has changed by around 90 degrees, as shown in Fig. 7b For such cases, where the two clusters do not belong to the same moving object, the change in orientation (shear) and the change in length (stretch) is far more pronounced. In the case where the models belong to the same object the effect of shear and stretch are far less distinctive as can be seen in Fig. 7c and d. Quantitatively the difference between the two cases can be seen in (e) and (f), which plot cumulative shear and stretch as a function of frame Index, "Don't Merge" representing the case where the motion models belong to different objects, and merge representing the case, where they belong to the same object. Algorithm 2 is designed around this observation.

Given points from two clusters M_1 and M_2, we calculate the internal shear and stretch for each of the models by calculating the average change in orientation and length, over the sequences of frames and edges of the Delaunay Triangulation of the points in the first frame. To calculate the inter-model shear, we first do a greedy minimum distance-based assignment, to assign a one-to-one mapping between points from M_1 to M_2. Similar to the internal shear case, the change in orientation of line-segments and length defined by the mapping are used to calculate the inter-model shear and stretch respectively. In order for the models to remain seperate, the inter-model shear must be atleast a factor of λ greater than the average intra-model shear. For all of our experiments, we set this parameter to 3. One might ask whether both shearing and stretching are required, or can we just use one and reduce computation. Generally we found that, either the shearing or the stretching dominated and not often both simultaneously. For example, Fig. 7c and d in the initialy frames the change in angle changes drastically, while length does so much more slowly. By the end of the sequence, the change in angle had stagnated, whereas the change in length was pronounced. A possible future direction of research is doing higher level traffic reasoning on the basis of shear and stretch. For example, in (h) one could use the information that the length between the models is increasing to infer that the object is approaching the camera. As can be seen by comparing (c) and (g), as well as (d) and (h), that, in the case where the models are seperate the changes in angle and length are much more pronounced.

5 Results

In this section we compare the accuracy and other relevant performance parameters of our proposed algorithms against other publicly available state-of-the-art algorithms. We do so on two datasets, the first being the Hopkins-155 dataset. As our algorithm was designed keeping in mind motion degeneracies that arise during typical on-road scenarios, we compiled a dataset comprised of on-road vehicles in degenerate motion. The important thing to note in them is they are taken from a camera mounted on the car designed for driverless vehicle research, and are real-sequences taken from a fast-moving vehicle, unlike the Hopkins-155 dataset. Also, unlike the Hopkins-155 dataset, the On-Road dataset has pre-dominantly degenerate motion, as evidenced by the failure of the epipolar constraint. That degenerate motions occur rarely in the Hopkins-155 dataset was mentioned in [44]. A consequence of this can also be seen in the high-accuracy of motion-segmentation algorithms not designed to handle degenerate motions like the kind proposed in [44].

5.1 Hopkins-155 Results

The Hopkins 155 [27] Dataset has been the benchmark for motion segmentation algorithms since first introduced in 2007. It consists of 155 sequences: 120 and 35, for the two and three rigid motions, respectively. Each sequence comes with a set of tracked points, and their ground-truth labels. To be noted is that the Hopkins-155 dataset, does trajectory level labelling. That is, even if an object is moving in the last few frames only, the trajectory points corresponding to the object throughout the sequence are labelled as moving.

We compare the error rate of our algorithm with those of state-of-the-art methods for the noise-less scenario, as well as for increasing levels of Gaussian noise. In addition we compare other relevant factors like, computation time, error distribution and accuracy in estimating the number of motions correctly.

The algorithms compared against cover the two most successful approaches to the motion segmentation problem, namely the subspace clustering and trajectory similarity (gestalt) based algorithms. The most basic of the subspace clustering algorithms is RANSAC [27], which samples trajectories and uses their principal components as basis vectors for the corresponding subspace. Generalized Principal Component Analysis (GPCA), introduced in [47], fits a polyomial to the subspace and then classifies it. The order of the polynomial indicates the subspace dimension. Spectral Curvature Clustering (SCC) [43] and Sparse Grassmanian Clustering (SGC) [26] use a higher order motion model compiled in a tensor, which is then unfolded into a matrix, followed by a clustering algorithm. Linear Combination of Views (LCV) [33] samples a set of nearby points and for them synthesizes a trajectory using the first and last frames of the sequence. Clustering is done on the basis of how well the actual trajectory corresponds to the synthesized one. Sparse Subspace Clustering (SSC) [3] models the trajectory points as a sparse combination of sampled trajectories. Adaptive Subspace Affinity [46] and MSL[7] are subspace fitting algorithms capable of handling degenerate sequences. Higher Order Spectral Clustering introduced in [49], uses a combination of a motion model as well as a trajectory similarity metric similar to (9) to cluster the various trajectories in the scene.

Accuracy for this dataset is given by,

$$\texttt{Accuracy \%} \;=\; \frac{\texttt{\# of points classified correctly}}{\texttt{Total \# of points}} \tag{11}$$

One drawback of most of the above algorithms like SCC, SGC, SSC, RANSAC etc. is that they need to be given the dimensions of the subspaces to be estimated or the number of motions in the scene, which is not generally known beforehand. In comparison, there are far fewer algorithms that do motion segmentation, without any information on the number of motions observed in the scene.

Accuracy With Noise Free Data Table 2 lists average error-rates according for the sequnces containing two motions, three-motions and both. Values for the other algorithms were taken from their respective papers. Due to the randomization in our algorithm, we conduct the run for our algorithms 100 times and found the average error rate to be 1.78 %, and its standard deviation to be 0.31 %. We achieved average errors of 1.27 % and 3.41 % for the two and three motions

Table 2 Total Error Rates in Noise-Free Case The first row represents error rates for all (155 seqeunces). The second and third rows are two and three motions, respectively. The values of the algorithms were taken from their corresponding papers

RANSAC	GPCA	MSL	SSC	SCC	SGC	ASA	ORK	LCV	Ours	
Total	9.48	10.02	4.46	2.70	4.10	2.05	1.24	8.91	1.86	2.22
Two motions	5.56	4.59	2.23	1.77	2.89	1.03	0.96	7.83	1.25	1.27
Three motions	22.94	28.66	4.14	5.89	8.25	5.53	2.22	12.62	3.97	4.41

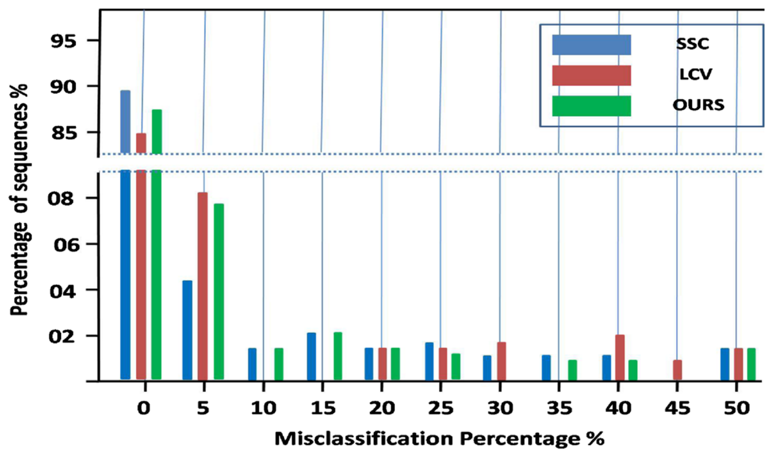


Fig. 8 Error distribution histogram for the Hopkins-155 dataset

case respectively. Our algorithm does so, without any manual user input as to the number of motions and the dimensions of their subspaces. It has the fourth best, overall performance, amongst the evaluated algorithms. Amongst the algorithms that can estimate the number of motions automatically (ORK and MSL) it has by far the highest performance.

Figure 8 shows the error distribution on the Hopkins-155 dataset. Our algorithm is comparable to the state-of-the-art with error exceeding 20 % only for 16 of the 155 sequences.

Test of Robustness For this test, we added artificial spherical-Gaussian noise (zero-mean) to the raw feature points of the trajectory matrix. Figure 9 show, the accuracy of our algorithm suffers more from the noise when compared to SSC and ASA, but outperforms LCV. We use the following covariance matrices, $\Sigma_n = \sigma_n^2 \mathbf{I}$, $\sigma_n = \{0.01, 0.25, 0.5, 1, 2, 4\}$ in our test.

Computation Time In Table 4, we show the average computation time for the entire dataset. For all the algorithms except ASA and SSC, we run the

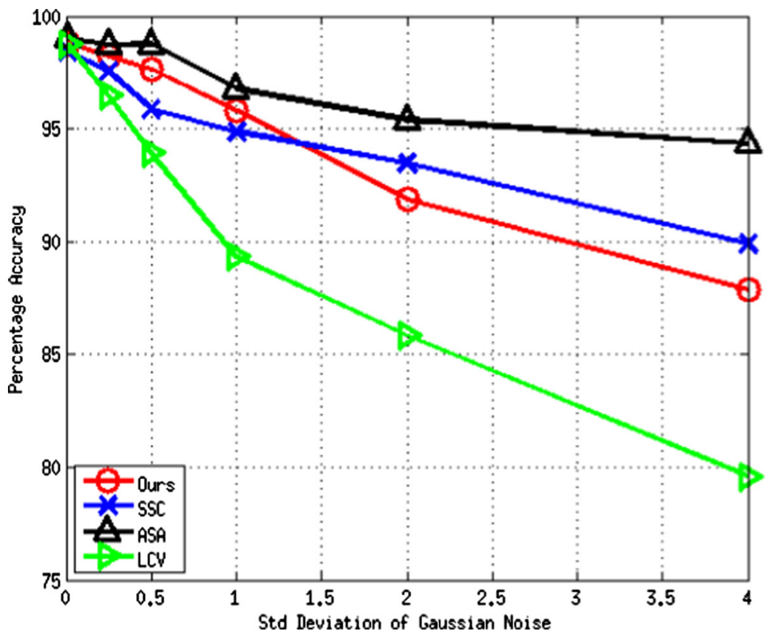


Fig. 9 Percentage accuracy with Gaussian noise added to the trajectory data

computation time is averaged over 100 runs. For, SSC we did 10 runs, and ASA just one due to it's high running time. The tests were conducted on an Intel Core i7-2600K 3.40 GHz × 8 core processor with 16 GB RAM. Our algorithm is implemented as unoptimized matlab code, except for the energy minimization which is written in C. Table 4 shows our algorithm is slower than RANSAC, SCC and LCV but, faster than SSC and ASA by a noticeable amount. The relative slowness of the algorithm can be attributed to the lack of a model-selection step in the other algorithms, as they recieve as input the number of motions, drastically saving in computation time.

Accuracy in estimating number of motions Within a robotics context, it is unrealistic to assume that the number of moving objects in the scene will be known beforehand. Candidate objects likely to move like people and vehicles may also be stationary, so using only semantic information to detect motion might lead to a wrong estimate. Estimating the number of moving objects in the scene, is thus a vital function a motion segmentation algorithm should be able to perform (Fig. 10).

To assess how good our algorithm is at estimating the number of motions present in the scene, we compare it to the recently proposed Enhanced Local Subspace Affinity (ELSA) [48] algorithm. ELSA performs motion segmentation by applying a spectral clustering algorithm, to its computed affinity matrix. It estimates the number of motions by analyzing the eigenvalues of the Symmetric Normalized Laplacian matrix. It proposes three approaches to carry out the analysis. They are Otsu's Method,

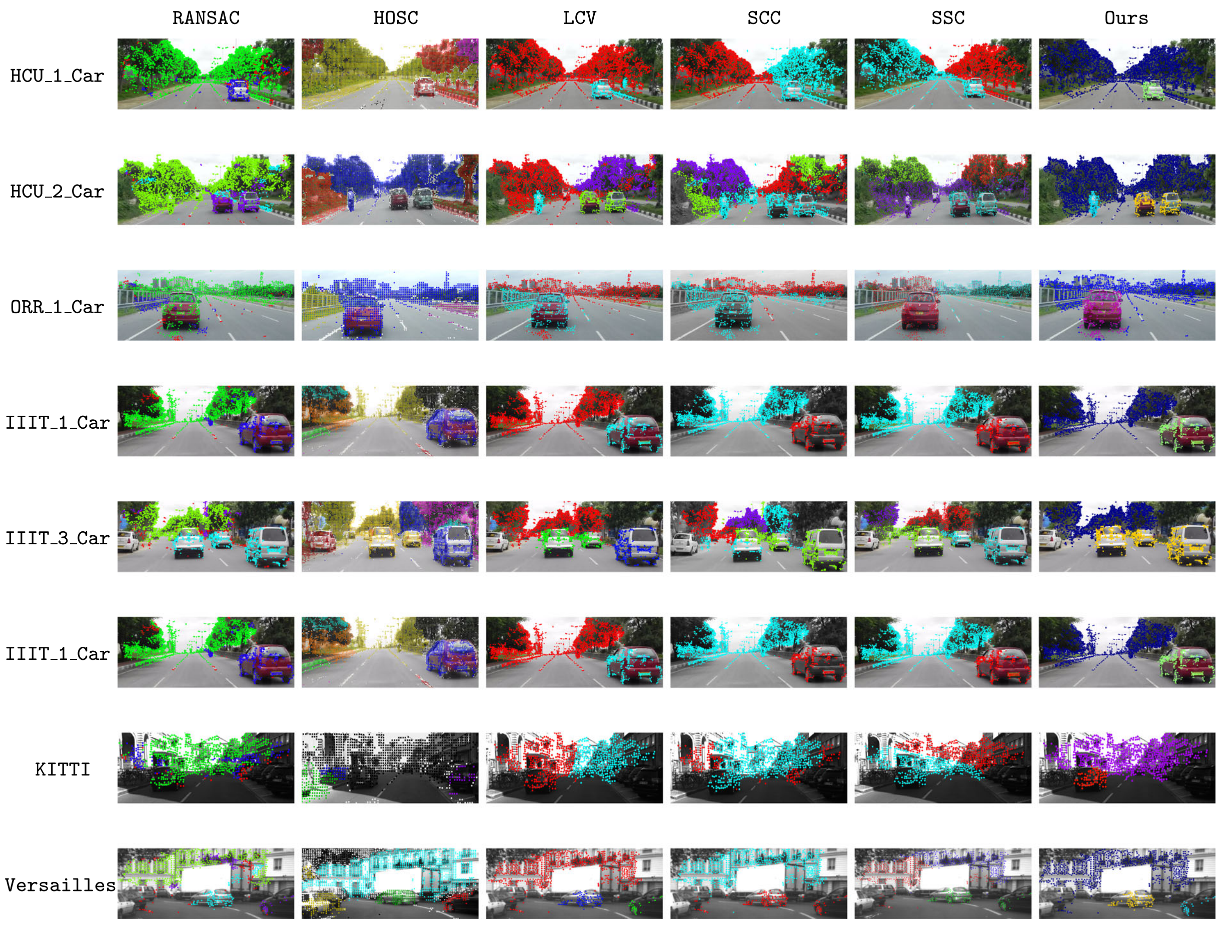


Fig. 10 Comparison of various state-of-the-art motion segmentation algorithms for the On-Road dataset

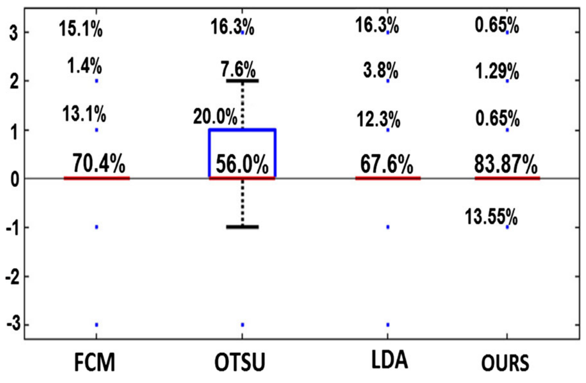


Fig. 11 Box-plot assessing the performancce of various algorithms at estimating the number of motions

Fuzzy C-Means(FCM) and Linear Discriminant Analysis (LDA). We compare our algorithm to all three approaches. The results for the three approaches are taken directly from the paper. Figure 11 shows the box-plot comparison, as can be seen, our algorithm outperforms the other three approaches by a significant margin, having both the largest number of datum between the two-quartile as well as least spread.This indicates that shear is a useful cue in estimating the number of motions.

5.2 On-Road Dataset

This dataset is a compilation of various scenes of vehicles moving on the road. In them the camera is following the moving vehicle, fequently making the object motion degenerate with respect to the camera. We use clips from the KITTI Dataset [21], Versailles-Rond [17], as well as ones taken around the IIIT-H campus. We choose these datasets specifically because we have found them to be challenging to segment giving only optical-flow information to our previous method [1] which incorporates the epipolar constraint, in a Bayes-filter framework. The flow magnitude of the moving object in them is close to that of the stationary world, as observed by the moving camera.

A performance evaluation is done on this dataset specifically to evaluate it's performance on degenerate motions.

For ground-truth in these datasets, we define a mask around the vehicles and take the points within to belong to their corresponding cluster. The remainder of the points, we take as background. The additional moving objects in the scene (like pedestrians or vehicles), are not detected as moving by all of the algorithms including ours, so are marked as stationary. They are either too far back in the background to have sufficient motion parallax or have insufficient points on them to be detected as seperate moving objects.

For accuracy we use (11). In our analysis, we compare our algorithms with those state-of-the-art algorithms for which either the source code or binaries are available. For HOSC [49], only binaries are provided, so no parameter tuning is possible. For the remainder of the algorithms, we give results in Table 3 with best found parameter settings.

However, for all algorithms that need as input the number of motions, we give the same input for each sequence, irrespective of accuracy. For example, in Fig. 10, for the HCU_2_Car dataset, there are four moving objects in the scene (a moped, two cars and the background), as that is the desired output in a motion segmentation task. This reduces the accuracy of the subspace clustering algorithms, as the two cars in the background have similar motion and belong to the same subspace, which the algorithms recognize as such.

Accuracy is averaged over 50 trials, for all algorithms except for SSC, HOSC. For SSC and HOSC we take 1 trial, as the algorithm is completely deterministic. We do not compare with ASA, due to it's lengthy computation time, making it completely unsuitable for near real-time robotic applications.

Table 3 shows, the subspace clustering methods fair relatively poorly for the On-Road datasets. LCV which forms trajectories from linear combinations of feature points that belong to the same object fairs a lot better. Like our proposed algorithm, it sythesizes trajectories by randomly selecting a point, and then using the point and it's neighbors to do so, allowing for spatial coherence in their motion models (trajectories). It however ends up fairing poorly on the KITTI and ORR_1_Car sequences, both of which have vehicles moving slower than the cameras, as shown in Fig. 10. In comparison, our algorithm is able to successfully segment out the moving vehicle in both the scenes. This is due to shear exhisting between the vehicles and the background. In the Versailles sequence, both SSC and SCC classify the two cars together, due to their similarity in heading, even though they are far apart.

Table 3 Error Rates on the On-Road Dataset For the various sequences Error Rates for the various algorithms under consideration

Method	RANSAC	LCV	SSC	SCC	Ours
KITTI	30.71	41.22	35.45	32.33	2.41
IIIT_3_Car	29.66	10.21	39.11	48.49	37.53
HCU_1_Car	8.79	2.16	43.21	40.32	3.56
HCU_2_Car	15.92	35.68	56.11	45.99	12.45
IIIT_1_Car	5.87	0.75	0.75	9.67	0.75
ORR_1_Car	35.85	21.34	37.05	28.93	5.11
Versailles	20.22	1.44	37.44	5.50	2.78

The trajectory similarity based algorithm, Higher Order Spectral Clustering (HOSC) [49] over-segments all the dataset sequences. the pattern of the over-segmentation is similar over all the sequences as shown in Fig. 10. The portion of the background, far away from the camera having low motion parallax is correctly clustered together. However, the portion of the background close to the camera (high motion parallax) is clustered seperately. The algorithm uses only a trajectory similarity metric similar to (9) to cluster the trajectories, resulting in the observed over-segmentation. The HOSC binaries output only the images shown in Fig. 10 and not the classification labels of the points, making it impossible to compare with the ground truth. We therefore can not show the classification error in Table 3, however, as is apparent from Fig. 10 it is outperformed by all of the other algorithms under consideration.

A lack of relative motion between moving objects will cause shear to fail. For example, in the IIIT_3_Car sequence, a lack of relative motion between the 3 vehicles causes all three vehiclesto be classified together. This foreseeable consequence arising from the nature of relative motion constraints probably indicates that using only motion cues, leaves the motion segmentation problem ill-posed. If in addition, an object detector, were used, to recognize all possible moving objects like pedestrians and vehicles, and shear constraints were checked between them and the background (area not containing potentially moving objects), the algorithm would perform more optimally. We hypothesize that the stationary vehicles and pedestrains, detected by an object detector, in the scene would be merged with the background and the moving objects would belong to their own seperate clusters irrespective of the relative motion between the objects.

Table 3 and Fig. 10 shows that incorporating only trajectory similarity cues is insufficient for motion segmentation in the case of a moving camera, as shown in the HOSC case. Likewise, using only subspace information frequently results in an erroneous estimation of the moving objects, due to their similarity in motion with the background. The superior performance of both LCV and our algorithm, can be attributed to a similar initial step. In both cases, motion models are calculated from nearby points, incorporating spatial coherence into the algorithms. Thus spatial coherence is a valuable attribute a motion segmentation algorithm should possess.

5.3 Discussion

In general we found that, for sequences with a reasonable amount of motion, our algorithm was able to accurately detect the number of motions. On the Hopkins dataset, the sequences the algorithm has problems with, were some of the checkerboard sequences, where, the moving object has no motion until at the very end of the sequence, and then it moves slightly. As our algorithm assigns labels based on the shear developed over the entire sequence of frames. Hence, if at the end of the considered number of frames the shear is pronounced enough to be labeled as moving the object is considered moving for the complete sequence. To provide a classification that responds to situations on a per frame basis is difficult and at times unwarranted as it results in frequent swapping of labels that can be construed as unstable. For the traffic sequences of the Hopkins dataset, it was able to detect motion correctly for all sequences.

Overall, we have tried to show through this section on results and discussions that our algorithm outperforms SSC, LCV, ASA etc. for typical on road robotic

Table 4 Total Computation Time on the Hopkins-155 Dataset in seconds

Method	LCV	ASA	SSC	SCC	RANSAC	Ours
Total	105.428	106882.7586	18552.4	143.95	43.44	1232.7

settings. It comes out to be the best amongst algorithms not knowing initial motions. It also generalizes to situations where the video is taken by a handheld camera, as is the case in the Hopkins dataset (Table 4).

6 Conclusions

In this paper we presented a motion segmentation algorithm, for trajectory data, capable of handling both degenerate and non-degenerate motions. It takes images in batch, and on the basis of judicious use of geometric and gestalt motion cues, performs the segmentation.

Initially we generate a large number of affine motion models using random sampling. We incorporate the information from these motion models along with long-term motion cues into an MRF-energy function. The number of motion models we end up getting after energy minimization, typically is larger than the number of moving objects in the scene. This over-segmentation, is crucial and we believe necessary in dealing with degenerate cases, where camera and object motion are nearly the same. The third, and final step involves merging the models from the reduced motion model set of the previous step based on a novel motion similarity constraint, explained in Section 4.1. The similarity constraint, we termed **in-frame shear**, measures the relative change in geometry between two motion models as observable from a monocular sequence of images.

Through experiments, we showed that our algorithm was competitive with the state-of-the-art algorithms in terms of accuracy. It was faster than all of the other algorithms with comparable accuracy except LCV. One big advantage the algorithm has over other methods is it, does so without knowing the number of motions beforehand. For the real-world datasets we tested it on, it outperformed the other algorithms by a substantial margin including LCV.

In future works, we intend to explore how incorporating object detection algorithms, like [9], into the above framework will speed up the algorithm, by restricting the sampling to regions of interest in the image. Using this speed up, in addition to the shear cues we elucidated in this paper, one should be able to come up with a faster more-robust algorithm, from which the intermediate results can be used in additional higher-level reasoning tasks, like scene understanding.

References

1. Namdev, R.K., Kundu, A., Krishna, K.M., Jawahar, C.V.: Motion segmentation of multiple objects from a freely moving monocular camera. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 4092–4099, IEEE (2012)
2. Lezama, J., Alahari, K., Sivic, J., Laptev, I.: Track to the future: Spatio-temporal video segmentation with long-range motion cues. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2011)
3. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2009, pp. 2790–2797. IEEE (2009)
4. Sugaya, Y., Kanatani, K.: Geometric structure of degeneracy for multi-body motion segmentation. In: Statistical Methods in Video Processing, pp. 13–25. Springer, Berlin Heidelberg (2004)
5. Flores-Mangas, F., Jepson, A.D.: Fast rigid motion segmentation via incrementally-complex local models. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2259–2266. IEEE (2013)
6. Yan, J., Pollefeys, M.: A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In: Computer Vision ECCV 2006, pp. 94–106, Springer, Berlin Heidelberg (2006)
7. Kanatani, K.: Motion segmentation by subspace separation and model selection. Image **1**, 1 (2001)
8. Rao, S.R., Tron, R., Vidal, R., Ma, Y.: Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008. pp. 1–8. IEEE (2008)
9. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(9), 1627–1645 (2010)
10. Fragkiadaki, K., Shi, J.: Figure-ground image segmentation helps weakly-supervised learning of objects. In: Computer Vision ECCV 2010, pp. 561–574. Springer, Berlin Heidelberg (2010)

11. Isack, H., Boykov, Y.: Energy-based geometric multi-model fitting. Int. J. Comput. Vis. **97**(2), 123–147 (2012)
12. Delong, A., Osokin, A., Isack, H.N., Boykov, Y.: Fast approximate energy minimization with label costs. Int. J. Comput. Vis. **96**(1), 1–27 (2012)
13. Lee, Y.J., Kim, J., Grauman, K.: Key-segments for video object segmentation. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 1995–2002. IEEE (2011)
14. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8), 888–905 (2011)
15. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
16. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(10), 1568–1583 (2006)
17. Comport, A.I., Malis, E., Rives, P.: Real-time quadrifocal visual odometry. Int. J. Robot. Res. **29**(2-3), 245–266 (2010)
18. Wertheimer, M.M.: Laws of organization in perceptual forms (1938)
19. Wong, H.S., Chin, T.J., Yu, J., Suter, D.: Efficient multi-structure robust fitting with incremental top-k lists comparison. In: Computer VisionACCV 2010, pp. 553–564. Springer, Berlin Heidelberg (2011)
20. Romero-Cano, V., Nieto, J.I.: Stereo-based motion detection and tracking from a moving platform. In: Intelligent Vehicles Symposium (IV), 2013 IEEE pp. 499–504, IEEE (2013)
21. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3354–3361. IEEE (2012)
22. Brox, T., Malik, J.: Object segmentation by long term analysis of point trajectories. In: Computer Vision ECCV 2010, pp. 282–295. Springer, Berlin Heidelberg (2010)
23. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. Int. J. Comput. Vis. **9**(2), 137–154 (1992)
24. Kundu, A., Krishna, K.M., Sivaswamy, J.: Moving object detection by multi-view geometric techniques from a single camera mounted robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009, pp. 4306–4312. IEEE (2009)
25. Tsai, D., Flagg, M., Nakazawa, A., Rehg, J.M.: Motion coherent tracking using multi-label mrf optimization. Int. J. Comput. Vis. **100**(2), 190–202 (2012)
26. Jain, S., Govindu, V.M.: Efficient Higher-Order Clustering on the Grassmann Manifold, ICCV, 2013 (2013)
27. Tron, R., Vidal, R.: A benchmark for the comparison of 3-d motion segmentation algorithms. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007, CVPR'07. pp. 1–8. IEEE (2007)
28. Vidal, R., Ma, Y.: A unified algebraic approach to 2-D and 3-D motion segmentation. In: Computer Vision ECCV 2004, pp. 1–15. Springer, Berlin Heidelberg (2004)
29. Fragkiadaki, K., Zhang, G., Shi, J.: Video segmentation by tracing discontinuities in a trajectory embedding. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1846-1853. IEEE (2012)
30. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
31. Li, Z., Guo, J., Cheong, L.-F., Zhou, S.Z.: Perspective Motion Segmentation via Collaborative Clustering, ICCV (2013)
32. Lenz, P., Ziegler, J., Geiger, A., Roser, M.: Sparse scene flow segmentation for moving object detection in urban environments. In: 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 926–932. IEEE (2011)
33. Zografos, V., Nordberg, K.: Fast and accurate motion segmentation using Linear Combination of Views. In: BMVC, pp. 1–11 (2011)
34. Weiss, Y., Adelson, E.H.: A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In: 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, pp. 321–326. IEEE (1996)
35. Black, M.J., Anandan, P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. Comp. Vis. Image Underst. **63**(1), 75–104 (1996)
36. Li, T., Kallem, V., Singaraju, D., Vidal, R.: Projective factorization of multiple rigid-body motions. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007, CVPR'07. pp. 1–6. IEEE (2007)
37. Weiss, Y.: Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In: 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997, Proceedings. pp. 520–526. IEEE (1997)
38. Weiss, Y.: Segmentation using eigenvectors: a unifying view. In: The proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, pp. 975–982. IEEE (1999)
39. Torr, P.H.S.: Geometric motion segmentation and model selection. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Phys. Eng. Sci. **356**(1740), 1321–1340 (1998)
40. Vidal, R., Soatto, S., Sastry, S.S.: A factorization method for 3D multi-body motion estimation and segmentation. In: Proceesings of the Annual Allerton Conference on Communication Control and Computing, vol. 140, no. 13, pp. 11626-1635. The University; 1998 (2002)
41. Vidal, R., Sastry, S.: Optimal segmentation of dynamic scenes from two perspective views. In: Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003, vol. 2, pp. II–281. IEEE (2003)
42. Shakernia, O., Vidal, R., Sastry, S.: Multibody motion estimation and segmentation from multiple central panoramic views. In: IEEE International Conference on Robotics and Automation, 2003, Proceedings, ICRA'03, vol. 1, pp. 571–576. IEEE (2003)
43. Chen, G., Lerman, G.: Spectral curvature clustering (SCC). Int. J. Comput. Vis. **81**(3), 317–330 (2009)
44. Chin, T.-J., Wang, H., Suter, D.: The ordered residual kernel for robust motion subspace clustering. In: NIPS, vol. 9, pp. 333–341 (2009)
45. Shi, J., Tomasi, C.: Good features to track. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994, Proceedings CVPR'94, pp. 593–600. IEEE (1994)

46. Zappella, L., Provenzi, E., Llad, X., Salvi, J.: Adaptive motion segmentation algorithm based on the principal angles configuration. In: Computer Vision-ACCV 2010, pp. 15–26. Springer (2011)
47. Vidal R., Hartley, R.: Motion segmentation with missing data by PowerFactorization and Generalized PCA. In: CVPR (2004)
48. Zappella, L. et al.: Enhanced local subspace affinity for feature-based motion segmentation. Pattern Recogn. **44.2**, 454–470 (2011)
49. Ochs, P., Brox, T.: Higher order motion models and spectral clustering. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2012)
50. Yuan, C., Medioni, G., Kang, J., Cohen, I.: Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(9), 1627–1641 (2007)
51. Kang, J., Cohen, I., Medioni, G., Yuan, C.: Detection and tracking of moving objects from a moving platform in presence of strong parallax. In: Tenth IEEE International Conference on Computer Vision, 2005, ICCV 2005, vol. 1, pp. 10–17 (2005)
52. Lourakis, M.I., Argyros, A.A., Orphanoudakis, S.C.: Independent 3D motion detection using residual parallax normal flow fields? In: Proceedings of the International Conference on Computer Vision, pp. 1012–1017 (1998)
53. Shashua, A., Wolf, L.: Homography tensors: on algebraic entities that represent three views of static or moving planar points. In: Proceedings of the European Conference on Computer Vision, vol. 1, pp. 507–521 (2000)