International Journal of Computer Vision (2022) 130:696–728
https://doi.org/10.1007/s11263-021-01544-x


# Multi-frame Motion Segmentation by Combining Two-Frame Results

**Federica Arrigoni**[1] · **Elisa Ricci**[1,2] · **Tomas Pajdla**[3]

Received: 19 February 2021 / Accepted: 27 October 2021 / Published online: 31 January 2022


**Abstract**
In this paper we consider the motion segmentation problem on sparse and unstructured datasets involving rigid motions, motivated by multibody structure from motion. In particular, we assume only two-frame correspondences as input without prior knowledge about trajectories. Inspired by the success of synchronization methods, we address this problem by introducing a two-stage approach: first, motion segmentation is addressed on image pairs independently; then, two-frame results are combined in a robust way to compute the final multi-frame segmentation. Our synthetic and real experiments demonstrate that the proposed approach is very effective in reducing the errors among two-frame results and it can cope with a large amount of mismatches. Moreover, our method can be profitably used to build a multibody structure from motion pipeline.

**Keywords** Motion segmentation · Multibody structure from motion · Synchronization

## 1 Introduction

Motion segmentation is a fundamental topic in Computer Vision and Robotic communities (Mattheus et al. 2020), which is relevant in a variety of applications ranging from 3D reconstruction (Saputra et al. 2018) to autonomous driving (Sabzevari and Scaramuzza 2016). The involved scenario is a dynamic scene with multiple objects that are moving rigidly and independently in the 3D space. Given several images of the scene (taken from a single moving camera in the form of a video or from different cameras at different times and viewing positions), the task is to identify the moving objects present in the scene.

Several approaches were proposed in the literature to address motion segmentation: some techniques assume *dense* correspondences (e.g., optical flow) over a video as input and predict a dense (i.e., pixel-wise) segmentation (e.g. Keuper et al. 2015; Bideau and Learned-Miller 2016; Keuper 2017; Bideau et al. 2018; Keuper et al. 2020); other methods, instead, work with a *sparse* input (e.g., sparse key-points) and produce a sparse segmentation as output (e.g. Vidal et al. 2005; Li et al. 2013; Ji et al. 2014; Xu et al. 2018; Arrigoni and Pajdla 2019a). The former are also referred to as "video object segmentation" by some authors as they make use of temporal continuity between consecutive frames within a video, and they will be discussed in Sect. 2.5. In this paper we focus on the latter, since we are interested in dealing with sparse and unordered datasets without any temporal component, motivated by multibody structure from motion. In other terms, our task is to perform motion segmentation by grouping together all the key-points that are moving in the same way, as shown in Figure 1.

Several approaches were proposed in the literature to address motion segmentation with sparse key-points (see Figure 3). Most of them assume that key-points have been *tracked* through the input video/images, and the task is to cluster those trajectories according to different motions (e.g., Vidal et al. 2005; Rao et al. 2010; Elhamifar and Vidal 2013; Ji et al. 2015; Li et al. 2013; Xu et al. 2018). A more practical and difficult scenario is analyzed in Ji et al. (2014); Wang et al. (2018), where it is assumed that a set of key-

Communicated by Stephen Lin.

✉ Federica Arrigoni
federica.arrigoni@unitn.it

Elisa Ricci
e.ricci@unitn.it

Tomas Pajdla
pajdla@cvut.cz

[1] Department of Information Engineering and Computer Science (DISI), University of Trento, Trento, Italy

[2] Deep Visual Learning Group, Fondazione Bruno Kessler, Povo, Italy

[3] Czech Institute of Informatics, Robotics and Cybernetics (CIIRC), Czech Technical University in Prague, Prague, Czech Republic

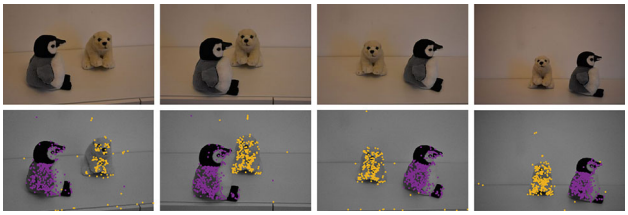


**Fig. 1** Top: sample images from the *Penguin* sequence (Arrigoni and Pajdla 2019b), representing two moving objects in an indoor environment. Bottom: key-points segmented with our approach, where different colors encode the membership to different motions. Our results are highly accurate (Color figure online)

points is given in the images with *unknown* correspondences. At the middle between trajectory clustering and the case of unknown correspondences there is motion segmentation with *two-frame* correspondences (Arrigoni and Pajdla 2019b, a), where it is assumed that key-points have been matched on image pairs only. Such an assumption is reasonable as computing matches between two images is, in general, easier than the case of multiple frames (Bernard et al. 2019).

In this paper we follow this latter research line and we propose a novel two-stage framework for motion segmentation with pairwise correspondences:

1. motion segmentation is solved on different image pairs in isolation;
2. such partial/local results are combined in a suitable way in order to produce a multi-frame segmentation.

The idea of solving motion segmentation in two steps is inspired by the success of synchronization methods (Arrigoni and Fusiello 2020), that solve several Computer Vision problems (e.g., point-cloud registration) using a similar principle, as it will be clarified in Section 2.6. Figure 2 shows a visual representation of the main stages of the proposed pipeline.

Concerning Step 1, plenty of techniques are available in the literature (see Section 2.3). One possibility – which is followed in this paper – is to use a multi-model fitting method (e.g., Robust Preference Analysis (Magri and Fusiello 2015)). Indeed, motion segmentation in two images can be solved by fitting multiple fundamental matrices to correspondences, under a perspective camera model.

Concerning Step 2, we introduce a new approach based on the following observation: if we consider a *fixed* image, then it will be involved (in general) in multiple two-frame segmentations (coming from all the pairs where the image is involved); such results provide (up to a permutation of the motions) possible solutions for segmenting points in the given image. We will show that fixing the permutation ambiguity can be formalized as a permutation synchronization problem (Pachauri et al. 2013). Then, we adopt the following strategy in order to assign a unique label to each key-point: the most frequent label (i.e., the mode) is chosen among all the possible solutions coming from different two-frame segmentations. This permits to exploit redundancy such that noise and potential errors from Step 1 are handled.

We also show that the results of our method can be further improved by employing spatial contiguity constraints. In other words, points which are close to each other are encouraged to belong to the same motion. We analyze three different approaches to accomplish such a task, which include a greedy solution (based on the percentage of neighbouring points belonging to the same motion) and two well-established techniques, namely constrained spectral clustering (Shi et al. 2010) and energy minimization (Boykov et al. 2001). It is worth noting that such a refinement is optional as – in many

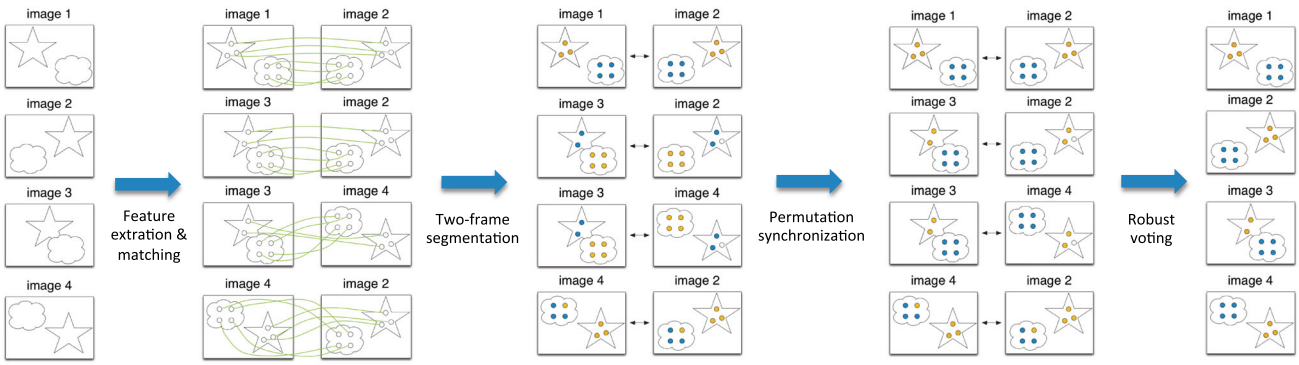



**Fig. 2** Proposed pipeline for motion segmentation. Given a set of images of a dynamic scene, key-points are extracted and two-frame correspondences are established. Then, for each image pair, motion segmentation is addressed in order to classify such correspondences into a number of motions. Finally, permutation synchronization is performed so that labels of motions are consistent across all the pairs, and robust voting is applied to produce the output multi-frame segmentation (Color figure online)

cases – the segmentation produced by our approach is sufficiently accurate.

We perform experiments on both synthetic and real scenarios. Results show that: our method is comparable or better than most solutions on the popular Hopkins155 dataset (Tron and Vidal 2007); it outperforms the techniques developed in Ji et al. (2015); Xu et al. (2018) on synthetic/real datasets with mismatches; it is very effective in reducing the errors in the initial two-frame segmentations; it can be profitably used to segment SIFT keypoints in a collection of images, whereas its closest competitor (Arrigoni and Pajdla 2019a) exhibits some failure cases; finally, our approach can be successfully applied to the problem of reconstructing a 3D dynamic scene, which is also known as multibody structure from motion. For the sake of a fair comparison, our experimental validation comprises methods developed for sparse motion segmentation (e.g., Ji et al. 2015; Xu et al. 2018; Arrigoni and Pajdla 2019a), whereas we do not consider techniques addressing flow-based motion segmentation (e.g. Keuper 2017; Bideau et al. 2018), as they make different assumptions on the input/output. Moreover, we focus on datasets involving *rigid* motions only. Indeed, our approach is designed for handling multiple rigidly moving objects, being based on the fundamental matrix estimation.

Observe that our approach implicitly assumes the existence of trajectories but it does not construct them explicitly. In fact, our method uses only (local) pairwise correspondences. The reason why we focus on this scenario is that we do not want to make early decisions on trajectories that may be wrong (even mixing different motions). Indeed, the earlier correspondences are joined into trajectories, the higher chance you have to make errors. Based on our approach, trajectories can be eventually computed *after* motion segmentation: in this way we can focus on each moving object separately, exploiting single-body tools (e.g. geometric verification via RANSAC), resulting in more precise trajectories. This scenario will be analyzed in Section 6.5.2, where we show how to apply our framework to multibody structure from motion.

The idea of combining results from individual image pairs is also present in Li et al. (2013), Lai et al. (2017), Xu et al. (2018): in particular, in Li et al. (2013) all the pairs are used, whereas in Lai et al. (2017), Xu et al. (2018) only pairs of consecutive frames are considered. These techniques, however, are different from our approach since they do not completely perform segmentation on image pairs but they rely on *intermediate* results only (i.e., correlation of corresponding points). Such results are used to build an affinity matrix that encodes the similarity between different trajectories, to which spectral clustering (Von Luxburg 2007) (or its multi-view variations (Cortes et al. 2009; Kumar et al. 2011; Wang et al. 2014)) is applied. Observe that the size of such an affinity matrix is equal to the number of trajectories. As a consequence, Li et al. (2013), Lai et al. (2017), Xu et al. (2018) perform trajectory clustering, namely they exploit multi-frame correspondences. Our method, instead, requires two-frame correspondences only. Differences between trajectory clustering and the case of two-frame correspondences are illustrated in Figure 3 and will be further clarified in the next section.

The paper is organized as follows. Section 2 describes previous research on motion segmentation and Section 3 formally introduces the problem. The proposed method is derived in Section 4, while Section 5 describes possible ways to refine its output. Experiments are reported in Section 6, some considerations about the advantages/limitations of our approach are given in Section 7, and the conclusion is drawn in Section 8. Appendices A and B review some background useful to understand our method. Some of the results presented in this paper previously appeared in a preliminary work (Arrigoni and Pajdla 2019b).

## 2 Related Work

In this section we review previous works on motion segmentation, focusing on methods working with sparse data. We start with a brief overview of two broader topics, namely subspace separation (Section 2.1) and multi-model fitting (Section 2.2). Then, we review existing methods for solving the segmentation problem, by considering separately the case of two frames (Section 2.3) and multiple frames (Section 2.4). We also explain how motion segmentation can be seen as a particular instance of subspace separation or multi-model fitting, under suitable assumptions on the camera model. In Section 2.5 we discuss the motion segmentation problem with dense input/output, which is sometimes referred to as "video object segmentation", and we clarify differences with respect to the scenario considered in this paper. Finally, we discuss the synchronization problem (Section 2.6), that inspired our approach.

### 2.1 Subspace Separation

The goal of *subspace separation* (also known as *subspace clustering*) is to cluster high-dimensional data drawn from multiple low-dimensional subspaces. The most general case considers subspaces with different dimensions and with arbitrary intersections. Available approaches include Generalized Principal Component Analysis (GPCA) (Vidal et al. 2005), Local Subspace Affinity (LSA) (Yan and Pollefeys 2006), Power Factorization (PF) (Vidal et al. 2008), Agglomerative Lossy Compression (ALC) (Rao et al. 2010), Sparse Subspace Clustering (SSC) (Elhamifar and Vidal 2013), Structured Sparse Subspace Clustering ($S^3C$) (Li and Vidal 2015), Stochastic Sparse Subspace Clustering (Chen et al.

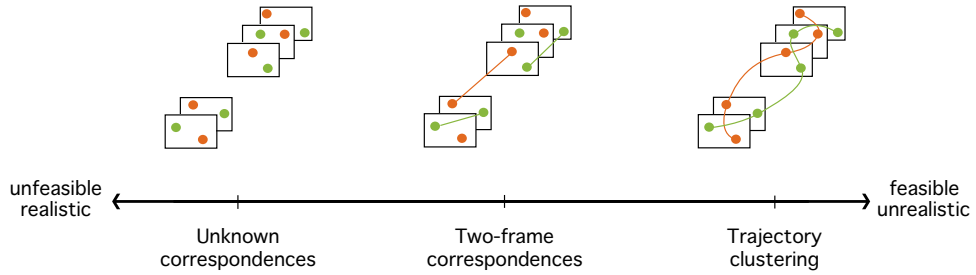



**Fig. 3** The proposed taxonomy divides existing approaches into three categories: trajectory clustering; segmentation with two-frame correspondences; segmentation with unknown correspondences. When moving from right to left the problem becomes more difficult to solve since assumptions are weaker (but more realistic). The approach we propose belongs to the category of two-frame correspondences methods (Color figure online)

2020), Low-Rank Representation (LRR) (Liu et al. 2013) and Robust Shape Interaction Matrix (RSIM) (Ji et al. 2015).

## 2.2 Multi-model Fitting

The objective of *multi-model fitting* is to estimate multiple models from unstructured data corrupted by outliers and noise. One example is the task of fitting geometric primitives (e.g., lines or circles) to points in the plane. This problem is challenging due to the inherent "chicken-and-egg" pattern: in order to estimate the models one needs to first cluster the data; in order to cluster the data it is necessary to know which model points belong to.

Some methods are based on the notion of *consensus*, that is – focusing on the estimation part of the problem – attempt to find models describing as many points as possible. Notable examples are the Hough transform (Xu et al. 1990), Sequential RANSAC (Vincent and Laganiere 2001), Multi-RANSAC (Zuliani et al. 2005) and Random Sample Coverage (Magri and Fusiello 2016). Other techniques follow a preference-based approach and focus on clustering as a basis to perform model estimation. Examples of methods in this category include Residual Histogram Analysis (RHA) (Zhang and Kosecká 2006), J-Linkage (Toldo and Fusiello 2008), Kernel Optimization (Chin et al. 2010), T-linkage (Magri and Fusiello 2014), Random Cluster Model (RCM) (Pham et al. 2014), Robust Preference Analysis (RPA) (Magri and Fusiello 2015) and Quantized Residual Preference (Zhao et al. 2020). The problem of fitting multiple models can be also formulated as an energy minimization problem (Delong et al. 2012a, b), as in PEARL (Propose Expand and Re-estimate Labels) (Isack and Boykov 2012) and Multi-X (Barath and Matas 2018).

## 2.3 Two-Frame Segmentation

The task of *two-frame segmentation* is to establish which feature points are moving according to the same model, given correspondences in two images. A perspective camera model is usually assumed. A geometric solution is derived in Vidal et al. (2006), where the fundamental matrix is generalized to multiple motions, giving rise to the so-called *multibody fundamental matrix*. Despite being appealing from a theoretical perspective, this method is not suitable for real applications as it is designed for noise-free correspondences. More practical approaches include (Jung et al. 2014; Poling and Lerman 2014; Schindler and Suter 2005).

Note that two-frame segmentation can be expressed in terms of subspace separation (see Section 2.1), since corresponding points following the same motion belong to a subspace of $\mathbb{R}^9$ of dimension at most 8 (after a proper rearrangement of coordinates), as explained in Li et al. (2013). Observe also that two-frame segmentation can be cast to a multi-model fitting problem (see Section 2.2). Specifically, motion segmentation can be achieved by fitting multiple fundamental matrices to correspondences in two images. Recall that, if a 3D point undergoes a rigid motion, its projections in two images are related by a fundamental matrix (Hartley and Zisserman 2004), with different motions giving rise to different fundamental matrices.

## 2.4 Multi-frame Segmentation

*Multi-frame segmentation* refers to the motion segmentation problem where multiple (i.e., $n \geq 3$) images of a dynamic scene are available. Compared to the case of two images, multi-frame segmentation is more challenging due to the increased number of unknowns. Previous works can be categorized into three main groups, namely trajectory clustering, segmentation with two-frame correspondences and with unknown correspondences. As shown in Figure 3, this taxonomy corresponds to different assumptions made on input data, which reflects into the applicability of methods in different settings. In particular, stronger assumptions make the problem easier to deal with, but they also tend to limit the applicability of a method in real world scenarios.

#### 2.4.1 Trajectory Clustering

*Trajectory clustering* refers to the case where a set of points is tracked through a sequence of images, and the task is to group those trajectories (i.e., *multi-frame* correspondences) into different motions. See the right part of Figure 3 for a visual representation. The typical scenario involves videos with small motions between consecutive frames, which appear in surveillance, scene understanding and autonomous driving (Sabzevari and Scaramuzza 2016; Rubino et al. 2018). Trajectory clustering methods – which are briefly reviewed here – constitute the majority of works in motion segmentation literature.

Some approaches (e.g., Vidal et al. 2005; Yan and Pollefeys 2006; Rao et al. 2010; Elhamifar and Vidal 2013; Liu et al. 2013; Ji et al. 2015) are based on subspace separation. Indeed, if an affine camera model is assumed, the point trajectories lie in the union of $d$ subspaces in $\mathbb{R}^{2n}$ of dimension (at most) 4, where $d$ denotes the number of motions and $n$ denotes the number of images. In a similar way, multi-model fitting techniques (e.g., Magri and Fusiello 2014; Barath and Matas 2018) can be exploited to address multi-frame segmentation under the affine camera model, by fitting multiple subspaces to feature trajectories.

A perspective camera, instead, is used in Li et al. (2013). More precisely, a joint optimization problem is formulated based on the SSC algorithm, where all image pairs are required to share a common sparsity pattern. In Lai et al. (2017) homographies over consecutive image pairs are sampled in order to build a correlation matrix, which is then used by spectral clustering (Von Luxburg 2007) to perform segmentation. This approach is later extended in Xu et al. (2018) where multiple models (affine, fundamental and homography) are combined to get an improved segmentation. Different approaches are analyzed to reach such task, namely Co-Regularization (Coreg) (Kumar et al. 2011), Kernel Addition (KerAdd) (Cortes et al. 2009), and Subset Constrained Clustering (Subset) (Wang et al. 2014). The main limitation of these approaches is that trajectories are seldom available in practice. For example, in the popular Hopkins dataset (Tron and Vidal 2007) – which has been extensively used in the literature – the input trajectories are not fully realistic since they were filtered with manual operations.

#### 2.4.2 Segmentation with Two-Frame Correspondences

The task of *segmentation with two-frame correspondences* is to group image points (e.g., SIFT keypoints Lowe (2004)) into different motions, assuming the knowledge of matches between pairs of images (i.e., *two-frame* correspondences). See the middle of Figure 3 for a visual representation. The typical scenario involves unstructured/unordered image collections with large motions between different frames (e.g., the indoor scenes used in Arrigoni and Pajdla (2019a)). A possible application is multi-body structure from motion (Saputra et al. 2018), that is a generalization of structure from motion to the dynamic case, where both motion segmentation and 3D reconstruction have to be solved.

Motion segmentation with two-frame correspondences is addressed in Arrigoni and Pajdla (2019a) and our paper only. Despite poorly studied, this problem has a great practical relevance since it does not assume the knowledge of multi-frame correspondences, which are hard to compute when moving objects are present. The most natural way to address this task is in two steps: first, segmentation is solved independently on different image pairs; then, such partial results are properly combined in order to get a multi-frame segmentation. Concerning the first step, a wealth of approaches are available (see Section 2.3). Concerning the second step, a Linear Algebra formulation is proposed in Arrigoni and Pajdla (2019a) such that the unknown multi-frame segmentation is recovered from the spectral decomposition of a proper binary matrix. This method can be viewed as a special case of spectral clustering (Von Luxburg 2007). Our paper shares similarities with Arrigoni and Pajdla (2019a), for it also adopts a two-step formulation, as explained in Section 3. However, we use a different approach to merge multiple two-frame segmentations (which is detailed in Section 4), resulting in significant improvement in performance, as shown in Section 6.

#### 2.4.3 Segmentation with Unknown Correspondences

Suppose that a set of image points (e.g., SIFT keypoints) is given with *unknown* correspondences, and the task is to compute multi-frame correspondences while at the same time grouping those trajectories according to different motions. See the left part of Figure 3 for a visual representation. *Segmentation with unknown correspondences* is addressed in Ji et al. (2014) and Wang et al. (2018) only. The former uses the Alternating Direction Method of Multipliers (ADMM) to jointly perform multi-frame segmentation and tracking, whereas the latter solves the problem via alternating optimization. Observe that the absence of correspondences is a very weak assumption which makes motion segmentation very difficult, due to the large number of unknowns. For this reason, existing solutions are not practical yet: in Ji et al. (2014), Wang et al. (2018) the maximum number of trajectories is set to 200 due to algorithmic complexity.

### 2.5 Video Object Segmentation

The task of *video object segmentation* (Yao et al. 2020) is detecting pixels corresponding to moving objects in a video, i.e., extracting segments that respect object boundaries, as well as associating object pixels temporally whenever they appear in the video. Some approaches (e.g., Bideau and

Learned-Miller 2016, 2018; Lin et al. 2018; Tokmakov et al. 2019) classify pixels as either moving or part of the background, but no distinction is made between separate moving objects. Other approaches, instead, give a separate label to each independently moving object (e.g., Keuper et al. 2015; Keuper 2017; Bideau et al. 2018; Dave et al. 2019; Keuper et al. 2020). In general, there is no restriction to rigid motions, but deformable or articulated objects are admissible (as happens, e.g., in the Freiburg-Berkeley Motion Segmentation (FBMS-59) dataset (Ochs et al. 2014) or the Moving Camouflaged Animals (MoCA) benchmark (Lamdouar et al. 2020)). Another example is the popular DAVIS benchmark (Perazzi et al. 2016), which will be discussed in Section 6.6. We refer the reader to the recent survey (Yao et al. 2020) for more details and references on video object segmentation.

Video object segmentation can be viewed as a motion segmentation problem where the input is in the form of dense optical flow instead of sparse data (which, in turn, can be either key-points alone or trajectories or two-frame correspondences). Accordingly, the output is in the form of a *dense* segmentation (namely each pixel is given a label). On the contrary, both our approach and methods reviewed in Section 2.4 associate labels with *sparse* key-points. Note also that video object segmentation has a clear sequential component, being based on videos, hence it uses much stronger information. Our approach, instead, is able to segment without temporal continuity as it works with unstructured and unordered datasets.

We discuss here in more detail the methods developed in Bideau and Learned-Miller (2016), Bideau et al. (2018) since they exploit two-frame results, similarly to our technique. In particular, they exploit consecutive frames and they formulate video object segmentation in a bayesian framework in order to compute the likelihood of a 3D motion direction associated with an optical flow vector, so as to maximize the information about how objects are moving differently. In Bideau and Learned-Miller (2016) the background region and a set of rigid motions are estimated, which are used as an initialization in Bideau et al. (2018). The authors of Bideau et al. (2018) also exploit semantic segmentation in order to assemble multiple rigid motions into complex (possibly flexible) objects. Observe that (Bideau and Learned-Miller 2016; Bideau et al. 2018) are different than our method as they work within a *causal* framework, in the sense that information from the previous time step is used as prior information for the current time step.

## 2.6 Synchronization

We conclude this section with a brief explanation of the *synchronization* problem, which inspired our approach. The goal of synchronization is to infer the unknown states of a network of nodes, where only the ratio (or difference) between pairs of states can be measured (Singer 2011; Arrigoni and Fusiello 2020). States are usually represented by elements of a group, such as the set of permutations or the set of Euclidean transformations. The former can represent local labels of a set of features, as it occurs in *multi-view matching* applications (Pachauri et al. 2013; Birdal and Simsekli 2019). The latter can represent camera reference frames (e.g., in the context of *structure from motion* (Govindu 2004; Hartley et al. 2011, 2013) or *pose graph optimization* (Carlone et al. 2015; Rosen et al. 2017)), or local coordinates of 3D point clouds when dealing with *3D registration* (Torsello et al. 2011; Bernard et al. 2015; Arrigoni et al. 2016). Another application is *image mosaicking*, where states are represented as homographies (Schroeder et al. 2011; Santellani et al. 2018).

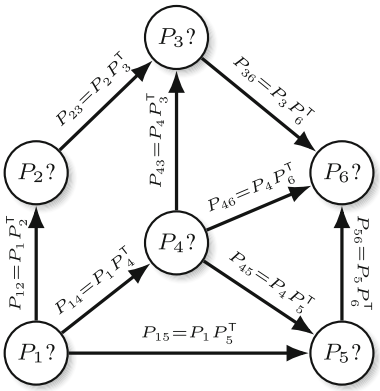



**Fig. 4** The permutation synchronization problem. The task is to recover unknown absolute/global permutations (on the nodes) starting from known pairwise/relative permutations (on the edges)

The synchronization problem can be modeled as a graph where nodes correspond to the unknown states and edges encode the pairwise measures, as shown in Figure 4. In other terms, each input measure involves a pair of nodes at a time. This means that the synchronization framework addresses a given Computer Vision application in two steps: first, the problem is solved for each pair of nodes in isolation, thus the original task is split into smaller subproblems which are easier to solve; then, these local results are combined by exploiting redundancy and seeking for error compensation.

Our paper is related to synchronization in two respects. First of all, we employ a two-step formulation of segmentation (see Section 3), which is similar in principle to synchronization methods. In particular, our approach – which recovers the segmentation of one image at a time (as explained in Section 4) – presents similarities with (Torsello et al. 2011; Hartley et al. 2011), which estimate the transformation of one camera/point-cloud at a time. Secondly, we use a specific synchronization routine (namely *permutation synchronization*) within our method, as it will be clarified in Section 4. The topic of permutation synchronization is explained in more detail in Appendix B.

**Table 1** Main variables used in this paper

| | |
|---|---|
| $n$ | Number of images |
| $d$ | Number of motions |
| $i, j, k$ | Variables used to index images |
| $r, h, l$ | Variables used to index key-points |
| $p_i$ | Number of key-points in image $i$ |
| $p$ | Total number of key-points over all the images |
| $\mathbf{s}_i$ | Total segmentation of image $i$ |
| $\alpha, \beta, \gamma$ | Variables used to index pairs of images |
| $m_\alpha$ | Number of matches in pair $\alpha$ |
| $\mathbf{t}_\alpha$ | Partial segmentation of pair $\alpha$ |
| $\mathbf{s}_i^\alpha$ | Estimate of total segmentation of image $i$ derived from pair $\alpha$ |
| $P_\alpha$ | Absolute permutation of pair $\alpha$ |
| $P_{\alpha\beta}$ | Relative permutation between pairs $\alpha$ and $\beta$ |
| $\mathcal{T}_i$ | Set of all the pairs involving image $i$ |

## 3 Problem Formulation

In this section we formulate the motion segmentation problem with two-frame correspondences, which is the focus of our work. The proposed formulation is based on the notions of total and partial segmentations. See Table 1 for a summary of our notation.

Let $n$ denote the number of images and let $d$ denote the number of motions. Suppose that a number $p_i$ of key-points is found in image $i$ using a feature extraction algorithm (e.g., SIFT Lowe 2004), so that the total amount of points over all the images is given by $p = \sum_{i=1}^{n} p_i$. In this paper we assume that the number of motions is known and constant over frames. Some insights about how to extend our approach to the case of an unknown number of motions are given in Section 6.7. We also assume that points have been matched in image pairs. Note that the knowledge of those correspondences, which involve two images at a time, is a weaker assumption than the presence of tracks, which involve all the images simultaneously, as already observed in Section 2.4.

The *total segmentation* of image $i$ is denoted by:

$$\mathbf{s}_i \in \{0, 1, \ldots, d\}^{p_i} \tag{1}$$

and it represents the labels of points in the $i$-th image:

- labels from 1 to $d$ identify the membership to a specific motion;
- the zero label identifies the *unclassified* points, namely those points whose motion can not be established due to missing or wrong correspondences.

The meaning of the zero label will be further clarified in Remark 1. Observe that total segmentations – which constitute our desired output – are an *absolute* representation of motion segmentation as they represent labels of points with respect to a *global* numbering of motions. See Figure 5 for a visual representation.

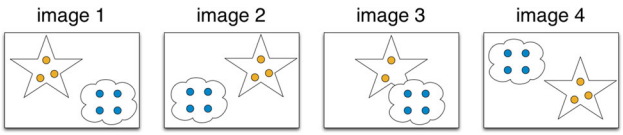



**Fig. 5** A set of points is detected in multiple images and the goal is to assign them a label (blue or yellow) based on the moving object (star or cloud) they belong to (Color figure online)

Let us consider an image pair, which is denoted by $\alpha = (i, j)$. Hereafter Greek letters are used to denote pairs of images. Suppose that $m_\alpha \leq \min\{p_i, p_j\}$ correspondences have been found with a matching algorithm (e.g., SIFT Lowe 2004). Hence, it is possible to run a two-frame segmentation method which groups those correspondences according to $d$ motions. The topic of two-frame segmentation is out of the scope of this paper and we refer the reader to Section 2.3 for an overview of existing approaches. Observe that those methods require *rigid* motions as they exploit the fundamental matrix model, hence our approach inherits such an assumption. The *partial segmentation* of the pair $\alpha = (i, j)$ is denoted by

$$\mathbf{t}_\alpha \in \{0, 1, \ldots, d\}^{m_\alpha} \tag{2}$$

and it represents the labels of corresponding points in the $i$-th and $j$-th images:

- Labels from 1 to $d$ identify the membership to a specific motion;
- The zero label identifies those correspondences which are labelled as outlier[1] by the chosen two-frame segmentation method.

Observe that a partial segmentation is a *local* representation of motion segmentation, as it reveals which points in two images belong to the same motion, but it does not reveal which motion it is with respect to the remaining images. See Figure 6 for a visualization. Observe also that it is a *partial* representation: if we consider image $i$, for instance, then there is a label only for those points in image $i$ which have a correspondence in image $j$, whereas remaining points (if any) are not labelled.

Thus motion segmentation with two-frame correspondences can be reduced to the problem of estimating the total segmentations of all the images, starting from a set of *known* partial segmentations. Observe that such a set is *redundant*

[1] It is expected that such outliers correspond to actual mismatches.

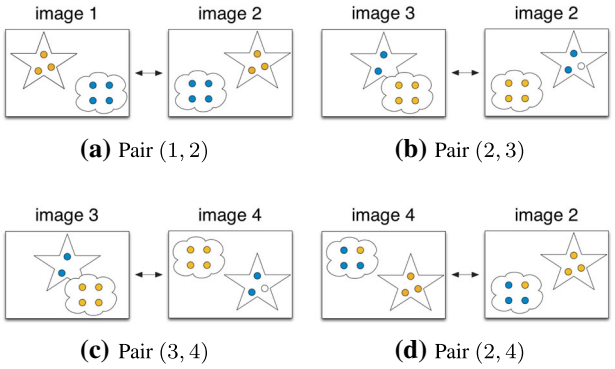



**Fig. 6** Motion segmentation is performed on image pairs (with possible errors). The same motion (star or cloud) may be given a different label (blue or yellow) in different pairs (Color figure online)

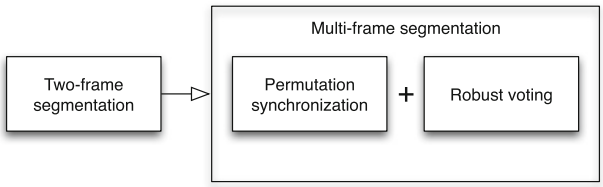



**Fig. 7** Outline of the proposed method

in most practical scenarios, as a given image usually appears in different pairs. Redundancy is the key to manage noise and outliers, as will be shown in Section 6. Hence we have to face the problem of how to assign a unique/global label to all the points, such that the constraints coming from pairwise segmentation are best satisfied. The next section will explain the proposed approach.

## 4 Proposed Method

In this section we present our solution to motion segmentation. Our method, which is summarized in Figure 7, takes as input several (independently computed) two-frame segmentations, which are properly exploited in order to produce the desired multi-frame segmentation.

Recall that the task is to estimate the total segmentations $\mathbf{s}_1, \ldots, \mathbf{s}_n$ starting from the knowledge of partial segmentations $\mathbf{t}_\alpha, \mathbf{t}_\beta, \ldots$ (associated with some image pairs $\alpha, \beta, \ldots$). Our key observation is that the partial segmentation $\mathbf{t}_\alpha \in \{0, 1, \ldots, d\}^{m_\alpha}$ gives rise to two vectors

$$\begin{aligned} \mathbf{s}_i^\alpha &\in \{0, 1, \ldots, d\}^{p_i} \\ \mathbf{s}_j^\alpha &\in \{0, 1, \ldots, d\}^{p_j} \end{aligned} \tag{3}$$

which contain labels of corresponding points in images $i$ and $j$, where missing correspondences are given the zero label. The superscript in Equation (3) refers to an image pair

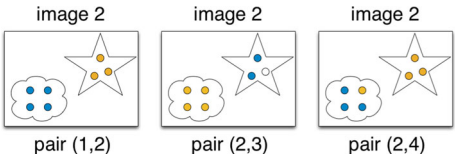



**Fig. 8** A possible solution for the total segmentation of image 2 is given by each partial segmentation where image 2 is involved. The same motion (star or cloud) may be given a different label (blue or yellow) in different pairs (Color figure online)

$\alpha = (i, j)$ whereas subscripts refer to individual images in the pair. This implies that, if we fix one image, then several estimates are available for its total segmentation, as shown in Figure 8. In particular, the amount of estimates is equal to the number of pairs where the chosen image is involved.

However, using such estimates is not straightforward, as two challenges have to be addressed:

- *Ambiguity*: each partial segmentation considers its own labelling of the motions, meaning that the same motion may have a different label in different pairs (see Figure 6);
- *Robustness*: each partial segmentation may contain errors, which in turn can be caused by mismatches and/or by failure of the method used for two-frame segmentation; moreover, some points may not have a label in a few pairs due to missing correspondences.

In the next paragraphs we will explain how to address these issues.

### 4.1 Ambiguity

In order to address the ambiguity challenge, we exploit a graph representation of the problem. Let us construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$ as follows:

- Each vertex corresponds to one pair of images;
- An edge is present between two vertices if and only if the associated pairs have one image in common.

Each vertex in the graph corresponds to an *unknown* permutation, as shown in Figure 9. Let $P_\alpha$ denote the permutation matrix associated with vertex $\alpha$, which corresponds to pair $\alpha$. The interpretation is that – after applying $P_\alpha$ to the partial segmentation $\mathbf{t}_\alpha$ – the ambiguity in the local labelling of motions is fixed, so that the same motion has the same label in different pairs. Observe that the involved permutations are represented as (square) $d \times d$ matrices since we are assuming that the number of motions is known and constant over all the frames.

Each edge in the graph corresponds to a *known* permutation derived as follows. Let $k$ be a common image between

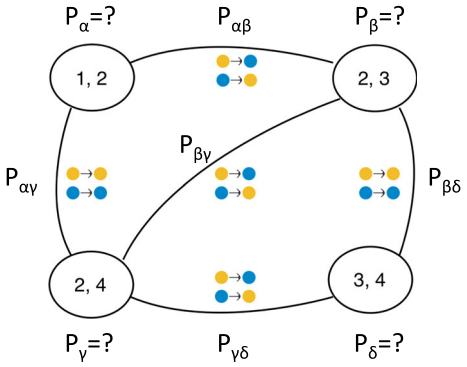



**Fig. 9** The graph formulation of the permutation synchronization problem. The vertices represent unknown permutations associated with image pairs. The edges represent known permutations between partial segmentations (Color figure online)

pairs $\alpha$ and $\beta$ (i.e., $k \in \alpha \cap \beta$) and let $P_{\alpha\beta}$ denote the permutation matrix associated with the edge $(\alpha, \beta)$, that is computed as follows:

$$P_{\alpha\beta} = \text{bestMap}\,(\mathbf{s}_k^{\alpha}, \mathbf{s}_k^{\beta}). \tag{4}$$

Equation (4) means that $P_{\alpha\beta}$ is the permutation that best maps the vector $\mathbf{s}_k^{\alpha}$ (i.e., labels of image $k$ in pair $\alpha$) into the vector $\mathbf{s}_k^{\beta}$ (i.e., labels of image $k$ in pair $\beta$). Recall that $\mathbf{s}_k^{\alpha}$ and $\mathbf{s}_k^{\beta}$ are recovered from $\mathbf{t}_{\alpha}$ and $\mathbf{t}_{\beta}$ respectively via Equation (3), which in turn requires the knowledge of pairwise correspondences. Finding $P_{\alpha\beta}$ is a *linear assignment* problem, which can be solved (for instance) with the Hungarian algorithm (Kuhn 1955). See Appendix A for more details.

To sum up, we have to address the problem of recovering an unknown permutation $P_{\alpha}$ for each vertex $\alpha \in \mathcal{V}$ starting from a (redundant) set of permutations $P_{\alpha\beta}$ with $(\alpha, \beta) \in \mathcal{E}$. Such matrices satisfy the following consistency constraint

$$P_{\alpha\beta} = P_{\alpha} P_{\beta}^{\mathsf{T}} \tag{5}$$

which defines a *permutation synchronization* problem (Pachauri et al. 2013). In other terms, the task is to connect motions across multiple image pairs. Equation (5) can be solved via spectral decomposition (Pachauri et al. 2013) (see Appendix B for more details).

At this point, the permutation $P_{\alpha}$ is applied to the partial segmentation $\mathbf{t}_{\alpha}$ for each pair $\alpha$. This has the effect of (possibly) reshuffling the labels of motions in individual pairs so that the permutation ambiguity is fixed, i.e., the same motion has the same label in different pairs.

## 4.2 Robustness

We now explain how to deal with errors in individual partial segmentations, thus addressing the robustness challenge mentioned above.

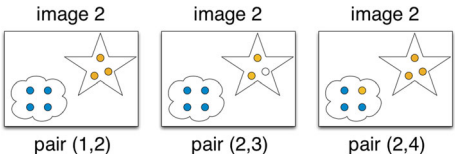



**Fig. 10** After solving a permutation synchronization problem, several estimates for the total segmentation of image 2 are available, where the same motion (star or cloud) has the same label (blue or yellow) in different pairs (Color figure online)

Recall that Equation (3) means that each partial segmentation provides a possible solution for the total segmentation of the two images involved in the pair. Thus, for a given image, several solutions are available for its total segmentation, which are given by $\{\mathbf{s}_i^{\alpha} \text{ s.t. } \alpha \in \mathcal{T}_i\}$. Here $\mathcal{T}_i$ denotes the set of all the pairs involving image $i$. See Figure 10 for an example.

In order to assign a label to each point, the following voting criterion is used

$$\mathbf{s}_i[r] = \text{mode}\,\{\mathbf{s}_i^{\alpha}[r] \text{ s.t. } \alpha \in \mathcal{T}_i,\ \mathbf{s}_i^{\alpha}[r] \neq 0\} \tag{6}$$

with $r = 1, \ldots, p_i$ and $i = 1, \ldots, n$. The idea is that the most frequent label (i.e. the *mode*) is, in general, correct in the presence of moderate noise. Observe that both missing correspondences and points labelled as outlier (if any) are ignored (i.e., the mode is computed over remaining points), as stated by the condition $\mathbf{s}_i^{\alpha}[r] \neq 0$. We set $\mathbf{s}_i[r] = 0$ (i.e., point $r$ in image $i$ is not labelled) in the case where $\mathbf{s}_i^{\alpha}[r] = 0$ for all $\alpha \in \mathcal{T}_i$, meaning that the point is either missing or classified as outlier in *all* the pairs. For the sake of robustness, we require that the mode is equal to (at least) two measures, otherwise the point is given the zero label.

Equation (6) is applied to all the points in all the images, thus producing the sought total segmentations $\mathbf{s}_1, \mathbf{s}_2 \ldots, \mathbf{s}_n$. As long as the algorithm used for two-frame segmentation correctly classifies all the points in most pairs, this procedure works well, as confirmed by experiments in Section 6.

***Remark 1*** When performing two-frame segmentation, it is expected that wrong correspondences are classified as outlier by the chosen algorithm (i.e., they are given the zero label). When dealing with total segmentations, instead, the situation is different: in principle, outliers do not exist since each image point actually belongs to a motion. However, in the presence of high corruption in the input correspondences, one may not be able to assign a valid label to all the points. Indeed, it may happen that a point is mismatched (and hence assigned the zero label) in all the pairs, so that there is no valid information to classify it. Such points are expected to have zero label in the total segmentation. However, since they are not actual outliers, we will refer to them as "unclassified" or "unknown" in the experiments.

***Remark 2*** We conclude this section with a few comments about how our approach manages missing data. In general, we expect that not all points are visible across different image pairs. For instance, if some points from image 2 are present in pair (2,3) but they are missing in pair (1,2), it means that the amount of data that is being used in Equation (4) is reduced. However, as long as we have at least one observation per motion, it is possible to recover the sought $d \times d$ permutation matrix via a linear assignment problem (see Appendix A). Concerning robust voting, it is easy to see that missing measures do not have any impact, as they are ignored when computing the mode in Equation (6) (see also Figure 10, where we can appreciate a missing label in the middle pair). Of course, the more observations we have, the better it is, as redundancy promotes error compensation.

## 5 Spatial Refinement

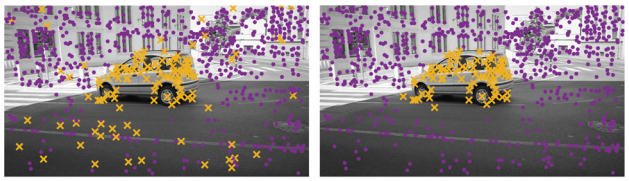
In this section we explain how the output of our approach – henceforth named MODE– can be improved by including spatial contiguity constraints. The usage of spatial priors is common in a wide spectrum of applications (e.g., Tombari and Di Stefano 2011; Delong et al. 2012b).

Recall that our method takes as input multiple two-frame segmentations, which are exploited in a suitable way in order to return a multi-frame segmentation (see Fig. 5). It is worth noting that in this way the actual coordinates of image points are not used anymore after two-frame segmentation, since only labels matter for the final segmentation.

Our method provides good results on a variety of motion segmentation datasets, as it will be shown in Section 6. However, a few points may be assigned the wrong label in some cases. The key observation is as follows: incorrectly classified points are not concentrated around a few locations, but they are sparse over the image plane, as shown in Figure 11 (see also Figures 20 and 21 for further examples). As a consequence, this issue can be easily mitigated by introducing spatial coherence, which makes use of point coordinates. The idea is that neighbouring points are often known to belong to the same motion, and should be encouraged to have the same label.

Accordingly, we propose here to refine the output segmentation obtained by MODE in order to get cleaner results exhibiting spatial consistency. In order to cover a wide spectrum of methodologies, we analyse three different ways to accomplish such a task:

- greedy approach;
- Constrained spectral clustering;
- Energy minimization.



**(a)** MODE **(b)** MODE + spatial refinement

**Fig. 11** Segmentation results are reported for our approach before (left) and after (right) the spatial refinement, on a sample image from *car-shadow* (Perazzi et al. 2016). Different colors correspond to different motions. In order to better appreciate the differences, points belonging to the moving car are drawn with a cross (Color figure online)

Such methods are explained in detail in the next paragraphs and they will be compared experimentally in Section 6.5.1. Observe that – with reference to our solution – a spatial refinement can be regarded as a post-processing which is applied at the end. In order to simplify the explanation, the aforementioned methods are described for a single image (with the provision that, in practice, the chosen refinement is applied to *all* the images).

### 5.1 Greedy Approach

Let us consider an image where motion segmentation has been already solved. A greedy approach to refine such result is based on the assumption that wrong labels are a minority and they are sparse over the image, hence we can easily establish whether a point is correct by checking if its label agrees with the majority of neighboring points.

More precisely, let us consider a point in an image and let us consider a ball with radius $\epsilon$ centered at the interest point. Let us count the percentage of points in the ball which have the same label as the interest point:

- If that percentage is greater than a threshold $\tau$, then the interest point is considered correct, and hence its label remains unchanged;
- Otherwise, the interest point is considered wrong and it is given the zero label (i.e., it becomes an unclassified point).

Observe that this approach is local, as the above procedure is applied to each point individually. Note also that this method can be regarded as a sort of outlier removal: the amount of classified points is reduced, in general, but the label of remaining points (which is likely to be correct) does not change. The following approaches, instead, optimize over the labels (i.e., some labels may be modified with respect to the initial segmentation) while maintaining fixed the amount of classified points.

### 5.2 Constrained Spectral Clustering

We first recall the idea behind constrained spectral clustering (Shi et al. 2010; Yuan et al. 2019), and then explain how it can be profitably used to improve the output of our method.

Spectral clustering is one of the most popular tools to partition a set of points into a (known) number of clusters (Von Luxburg 2007). Usually, data are represented in terms of an *affinity matrix* $A$, that encodes the similarity between pairs of points, and the partition is found from the eigenvalue decomposition of such a matrix. *Constrained* spectral clustering is a generalization of spectral clustering where additional information provided by the user is exploited (Yuan et al. 2019). For example, assume that some points are believed to belong to the same cluster, one thus expects the final result to be consistent with this prior knowledge. This is typically modelled with a *constraint matrix* $C$ defined as follows:

- $[C]_{hl} = 1$ if point $h$ and point $l$ are to be in the same cluster;
- $[C]_{hl} = 0$ otherwise.

A possible way to incorporate the above constraints into the spectral clustering algorithm is to replace the affinity matrix $A$ with the weighted sum between $A$ and the constraint matrix $C$ (Shi et al. 2010), namely

$$(1-\delta)A + \delta C. \tag{7}$$

Here $\delta \in [0, 1]$ denotes a parameter that balances the trade-off between maximizing cluster homogeneity and preserving the constraints of the data. When $\delta$ approaches zero, the solution is more biased towards maximizing the feature similarity whereas when $\delta$ approaches one, it is more biased towards preserving the constraints.

We now explain how constrained spectral clustering can be applied to our specific problem. Let us focus on a given image and let us consider the segmentation produced by MODE on that image, which clusters key-points according to different motions: such segmentation can be interpreted as a prior knowledge, which is encoded in the constraint matrix $C$. The affinity matrix $A$, instead, models spatial coherence:

- $[A]_{hl} \approx 1$ if point $h$ and point $l$ are spatially adjacent;
- $[A]_{hl} \approx 0$ otherwise.

One example is the exponential kernel:

$$[A]_{hl} = \exp\Big(-\frac{||\mathbf{x}_h - \mathbf{x}_l||^2}{2\sigma^2}\Big) \tag{8}$$

where $\mathbf{x}_h$ and $\mathbf{x}_l$ denote the coordinates (in the image plane) of points $h$ and $l$, respectively. By considering the weighted sum in Equation (7) we are taking into account both the spatial relationships and the segmentation produced by our method. Standard spectral clustering is then applied, which is expected to produce improved results. This procedure is applied to each image individually, where points with a valid label are considered only (i.e., unclassified points are ignored).

### 5.3 Energy Minimization

We now turn our attention to the last approach we consider to perform a spatial refinement, namely energy minimization. We start with a brief overview of this topic and then explain how it can be applied to our problem.

In a *labeling problem* we are given a set of observations (e.g., data points) and a finite set of labels (e.g., categories or geometric models), and the goal is to assign each observation a label such that some objective function (which is called *energy*) is minimized. Let $\mathbf{f}$ denote the sought labelling, where $\mathbf{f}[h]$ denotes the label of point $h$. Typically, the energy has the following form:

$$E(\mathbf{f}) = \sum_{h=1}^{q} D(\mathbf{f}[h]) + \sum_{(h,l)\in\mathcal{N}} V(\mathbf{f}[h], \mathbf{f}[l]). \tag{9}$$

The first term in the above equation represents a *data cost*, which sums the contributions of all the points, where $q$ denotes the number of points. The second term is a regularizer encouraging spatial coherence, which is called the *smooth cost*: each addend penalizes $\mathbf{f}[h] \neq \mathbf{f}[l]$ in some manner, where $\mathcal{N}$ denotes the set of neighbors. Equation (9) can be optimized effectively with the $\alpha$-expansion algorithm (Boykov et al. 2001). In some applications an additional term (named the *label cost*) is included in Equation (9), which penalizes overly-complex models, thus preferring to explain the data with fewer labels (Delong et al. 2012b). However, we do not consider such term here as we assume that the number of motions is known a priori.

We now explain how to employ energy minimization in order to improve the segmentation results obtained with MODE. Let $\mathbf{f}^{\text{init}}$ denote the segmentation produced by our approach for an image, where unclassified points are ignored. We consider the following data cost:

$$D(\mathbf{f}[h]) = \begin{cases} \gamma & \text{if } \mathbf{f}[h] = \mathbf{f}^{\text{init}}[h] \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where labels different from the initial values are penalized via the parameter $\gamma$. In order to define the set of neighbors $\mathcal{N}$, we use the "k-nearest neighbor" strategy, namely each

point is connected to its $k$ nearest points with respect to the Euclidean distance. Concerning the smooth cost, we employ Potts model (Kohli et al. 2007), which penalizes neighboring points with different labels, namely

$$V(\mathbf{f}[h], \mathbf{f}[l]) = \begin{cases} 1 & \text{if } \mathbf{f}[h] = \mathbf{f}[l] \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Recall that the smooth cost in Equation (9) counts contributions over neighboring points only. We use the $\alpha$-expansion algorithm (Boykov et al. 2001) to minimize the combined energy, thus producing an improved segmentation for the chosen image, and this procedure is repeated for all the images.

## 6 Experiments

In order to evaluate the performance of the proposed approach, we report experiments on both synthetic data and real images, considering both indoor and outdoor scenes. Our main focus is on motion segmentation with sparse and unstructured datasets, motivated by multibody structure from motion. However, we also consider datasets coming from video sequences in order to see what can be done with a method that does not use temporal information. We concentrate on datasets involving rigid (or approximately rigid) motions only, since our technique manages multiple rigidly moving objects via the fundamental matrix model, but it is not designed for highly non-rigid or deformable objects. The Matlab implementation of our method – named MODE– is available on the web.[2]

### 6.1 Setup

Since our approach addresses motion segmentation with two-frame correspondences, we focus on methods considering the same assumptions in order to provide a fair comparison (see Section 2.4.2). In addition to SYNCH (Arrigoni and Pajdla 2019a), we also consider a trivial solution (named the BASELINE), which permits to make interesting observations about how our approach exploits redundancy. To summarize, we consider the following competitors:

- SYNCH (Arrigoni and Pajdla 2019a) starts from multiple two-frame segmentations, similarly to our approach; then, it derives the unknown segmentation from the spectral decomposition of a big binary matrix, which is properly constructed from the input two-frame segmentations; this method can be viewed as a special case of spectral clustering or as a "synchronization" (Arrigoni and Fusiello 2020) of binary matrices;
- the BASELINE is a trivial solution constructed as follows: first, a maximum-weight spanning tree is built, where the underlying graph has a node for each image and edges are weighted with the number of (inlier) correspondences; secondly, the results from two-frame segmentation are exploited to segment each image along the tree, where the global numbering of motions is fixed at the root and sequentially propagated to the leaves.

MODE, SYNCH, and the BASELINE require a set of two-frame segmentations as input, which are computed as follows. For each image pair, Robust Preference Analysis (RPA) (Magri and Fusiello 2015) is used in order to fit multiple fundamental matrices to correspondences. RPA combines principles of robust principal component analysis (Lin et al. 2010) and non-negative matrix factorization (Kuang et al. 2014), in order to extract multiple models from data corruputed by outliers. The RPA code is available online.[3] In our experiments we use default values specified in the original paper for each algorithmic parameter. See Section 2.3 for more details about the connection between two-frame segmentation and multi-model fitting.

In order to enrich the evaluation, we also consider two techniques performing trajectory clustering:

- RSIM (Ji et al. 2015) provides a robust solution to subspace separation (see Section 2.1) and it comes with a public implementation;[4]
- Subset (Xu et al. 2018) can be regarded as the current state of the art in trajectory clustering, with mean error of 0.31% on the Hopkins155 benchmark (Tron and Vidal 2007) (see Tab. 2); the code is available online.[5]

Recall that trajectory clustering is a different task than the one addressed in this paper (see Figure 3), hence a comparison with RSIM and Subset is not entirely fair. However, by considering trajectory clustering methods and datasets, it is possible to give interesting insights about how methods designed for a specific problem behave when applied to another (related) task.

Techniques addressing video object segmentation (e.g., Keuper 2017; Bideau et al. 2018) are not included in the comparison, since they require a much different input (dense optical flow), as explained in Section 2.5.

Similarly to most works in segmentation literature, we assume that the number of motions is known in advance and

[2] https://github.com/federica-arrigoni/ICCV_19.

[3] http://www.diegm.uniud.it/fusiello/demo/rpa/.

[4] https://github.com/panji1990/Robust-shape-interaction-matrix.

[5] https://alex-xun-xu.github.io/ProjectPage/CVPR_18/.

**Table 2** Average misclassification error [%] for several methods on the Hopkins155 benchmark (Tron and Vidal 2007). Results for trajectory clustering approaches are copied from Xu et al. (2018)

| | LSA Yan and Pollefeys (2006) | GPCA Vidal et al. (2005) | ALC Rao et al. (2010) | SSC Elhamifar and Vidal (2013) | TPV Li et al. (2013) | LRR Liu et al. (2013) | T-Linkage Magri and Fusiello (2014) | $S^3C$ Li and Vidal (2015) | RSIM Ji et al. (2015) | MSSC Lai et al. (2017) | KerAdd Xu et al. (2018) | Coreg Xu et al. (2018) | Subset Xu et al. (2018) | SYNCH Arrigoni and Pajdla (2019a) | Baseline | MODE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 Motions | 4.23 | 4.59 | 2.40 | 1.52 | 1.57 | 1.33 | 0.86 | 1.94 | 0.78 | 0.54 | 0.27 | 0.37 | **0.23** | 2.70 | 2.26 | 1.00 |
| 3 Motions | 7.02 | 28.66 | 6.69 | 4.40 | 4.98 | 4.98 | 5.78 | 4.92 | 1.77 | 1.84 | 0.66 | 0.75 | **0.58** | 6.99 | 9.04 | 2.67 |
| All | 4.86 | 10.02 | 3.56 | 2.18 | 2.34 | 1.59 | 1.97 | 2.61 | 1.01 | 0.83 | 0.36 | 0.46 | **0.31** | 3.67 | 3.79 | 1.37 |

The lowest errors are highlighted in bold face

we give this value as input to all the analysed techniques. We discuss how to extend our approach to the case of an unknown number of motions in Section 6.7.

## 6.2 Hopkins Datasets

The Hopkins155 benchmark (Tron and Vidal 2007) comprises 155 sequences of indoor and outdoor scenes with two or three motions, which are categorized into checkerboard, traffic and articulated/nonrigid sequences. The Hopkins12 dataset (Vidal et al. 2008) provides 12 additional sequences with missing data. We emphasize that these datasets are designed for *trajectory clustering*, as they provide (cleaned) tracks over multiple images. Hence, they are not suitable for the task addressed in this paper, which is segmentation with two-frame correspondences. However, we report results on these sequences since they are widely used in the literature.

Accordingly, particular care needs to be taken into account for properly running our approach, the BASELINE and SYNCH, as they make different assumptions than trajectory clustering methods. In order to produce the input, (noise-free) two-frame correspondences can be straightforwardly computed from the available trajectories. Concerning the output, recall that MODE, SYNCH and the BASELINE classify image points, thus a scheme that assigns a unique label to each track is required. To accomplish such a task, we use the same criterion as the one developed in Section 4.2 to label each image point given multiple measures derived from two-frame segmentations: we assign to each track the mode of the labels of points belonging to the track.

Since ground-truth segmentation is available, we can provide a quantitative evaluation. In particular, we measure performance in terms of *misclassification error*, that is the percentage of misclassified tracks, as it is customary in motion segmentation literature. Tracks labelled as zero (if any) are counted as errors, since there are no outliers in these datasets.

Results are reported in Tab. 2 and Tab. 3 where MODE is compared to several motion segmentation algorithms. Our approach clearly outperforms SYNCH and the BASELINE, which address motion segmentation with two-frame correspondences. Observe also that MODE performs comparably or better than most trajectory clustering techniques, with a mean error of 1.37% over all the sequences in Hopkins155 and a median error of 0.38% over all the sequences in Hopkins12. In particular, it is noticeable that our method achieves (nearly) zero error in 139 out of 155 sequences in Hopkins155 and in 10 out of 12 sequences in Hopkins12, as shown in Figure 12. By manual inspection, it was found that in the remaining sequences the algorithm used for two-frame segmentation (RPA) performed bad in most image pairs.

The fact that MODE is not the best is not surprising since we are making much weaker assumptions (matches between

**Table 3** Average and median misclassification error [%] for several methods on the Hopkins12 benchmark Vidal et al. (2008). Results for different variants of ALC and SSC are taken from Ji et al. (2015) whereas results for the remaining methods are copied from the respective papers

| | PF Vidal et al. (2008) | PF+ALC Rao et al. (2010) | RPCA+ALC Rao et al. (2010) | $\ell_1$+ALC Rao et al. (2010) | SSC-R Elhamifar and Vidal (2013) | SSC-O Elhamifar and Vidal (2013) | RSIM Ji et al. (2015) | KerAdd Xu et al. (2018) | Coreg Xu et al. (2018) | Subset Xu et al. (2018) | SYNCH Arrigoni and Pajdla (2019a) | Baseline | MODE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean | 14.94 | 10.81 | 13.78 | 1.28 | 3.82 | 8.78 | 0.61 | 0.11 | **0.06** | **0.06** | 5.46 | 7.45 | 4.33 |
| Median | 9.31 | 7.85 | 8.27 | 1.07 | 0.31 | 4.80 | 0.61 | **0.00** | **0.00** | **0.00** | 0.57 | 2.16 | 0.38 |

The lowest errors are highlighted in bold face

image pairs instead of tracks over multiple images), i.e., we are addressing a more difficult task, as already observed in Section 2.4.2. Observe also that our approach is *robust*, hence it is naturally sub-optimal in scenarios where outliers are absent (as happens in the Hopkins benchmark). Nevertheless, our method achieves good performances. In general, there is no reason to use our method when trajectories are available and one out of the best traditional methods (e.g. Ji et al. 2015; Lai et al. 2017; Xu et al. 2018) can be used. *Our method, instead, is designed for the scenario where two-frame correspondences are available only*.

We now focus on the spatial refinement and analyse three different solutions, namely a greedy approach, constrained spectral clustering and energy minimization. As explained in Section 5, such techniques can be viewed as a "post-processing" to be applied to the output of MODE, thus we get three different versions of our method:

- MODE-G (our method + greedy approach)
- MODE-S (our method + constrained spectral clustering)
- MODE-E (our method + energy minimization).

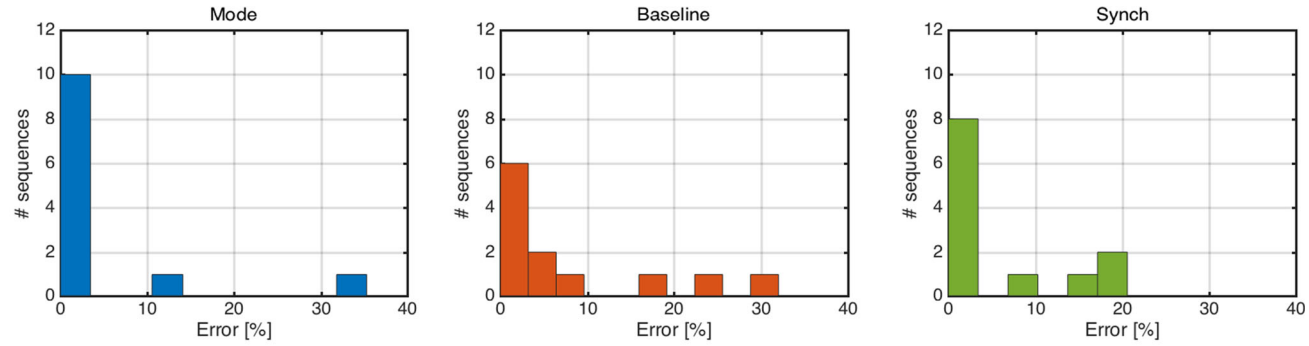
In our experiments we use $\tau = 0.7$, $\epsilon = 30$ pixels, $\delta = 0.6$, $\gamma = 10$ and $k = 10$ (see Section 5 for more details).

Results are given in Table 4, which reports the misclassification error for the aforementioned methods. As a reference, results for MODE are also included, which are copied from Tables 2 and 3 . From Table 4 we can appreciate that the spatial refinement does not cause a significant improvement on the Hopkins benchmark. This phenomenon agrees with the intuition that such a refinement works well when there are sparse errors over the image plane, but it is not able to correct gross errors in the segmentation. Recall that, as shown in the histograms in Figure 12, our method is either very accurate or it performs poorly due to wrong two-frame segmentations. We will see in Sections 6.5.1 and 6.6 some scenarios where the spatial refinement can be profitably applied.

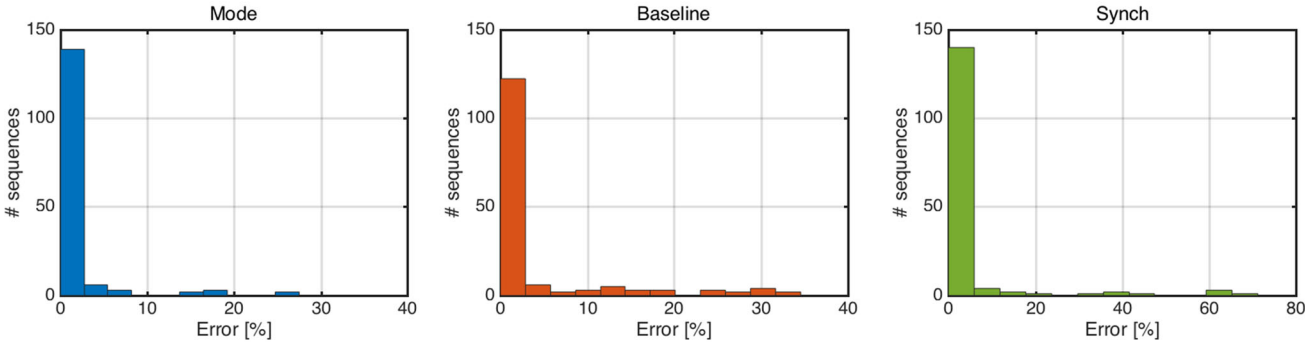
### 6.3 MTPV62 Benchmark

The MTPV62 dataset Li et al. (2013) comprises 62 sequences with two or three motions with strong perspective effects. Similarly to the Hopkins benchmark, this dataset has been developed for trajectory clustering, hence it is considered here as a reference only, for it does not represent the target application of our method.

Results are given in Table 5, which reports the misclassification error achieved by several segmentation algorithms. We can observe that MODE is significantly better than its closest competitors (namely SYNCH and the BASELINE) and that the spatial refinement does not bring a significant improvement on this dataset. Concerning trajectory clustering methods, our approach outperforms GPCA (Vidal et al. 2005), ALC




**(a)** Hopkins12




**(b)** Hopkins155

**Fig. 12** Histograms of misclassification errors achieved by SYNCH (Arrigoni and Pajdla 2019a), MODE, and the BASELINE on the Hopkins155 (Tron and Vidal 2007) and Hopkins12 (Vidal et al. 2008) datasets. The horizontal axis corresponds to a possible misclassification error in an individual sequence, and the vertical axis corresponds to the number of sequences where a given error is reached (Color figure online)

**Table 4** Misclassification error [%] and classified points [%] for different variants of our approach on the Hopkins155 (Tron and Vidal 2007) and Hopkins12 (Vidal et al. 2008) datasets

| Dataset | | MODE | MODE-G | MODE-S | MODE-E |
|---|---|---|---|---|---|
| Hopkins155 | 2 Motions | 1.00 | **0.88** | 1.00 | 0.96 |
| | 3 Motions | 2.67 | **2.34** | 2.44 | 2.63 |
| | All | 1.37 | **1.21** | 1.32 | 1.34 |
| Hopkins12 | Mean | 4.33 | **4.16** | 4.33 | 4.35 |
| | Median | 0.38 | **0.34** | 0.38 | 0.38 |

The lowest errors are highlighted in bold face

(Rao et al. 2010) and SSC (Elhamifar and Vidal 2013), while the best performance is achieved by MSSC (Lai et al. 2017) and Subset (Xu et al. 2018).

The considerations made for the Hopkins datasets apply equally well to the MTPV62 benchmark: it is worth noting that our approach works under weaker assumptions than the best performing methods, being designed for motion segmentation with two-frame correspondences. The next sections will demonstrate the advantages of our approach for this specific task.

## 6.4 Simulated Data

In order to study the robustness to mismatches of our approach, we consider four sequences from the Hopkins155 dataset, namely *1R2RCR_g12*, *2RT3RTCRT*, *cars2_06* and *cars1*, whose properties are summarized in Tab. 6. Noise-free pairwise matches are obtained from the available trajectories and synthetic errors are added to these correspondences in order to produce mismatches. More precisely, in each image pair we perform the following operations: first, we randomly select a fraction (which ranges from 0 to 0.8 in our experiments) of the correspondences out of the total amount of matches; secondly, such correspondences are switched via

**Table 5** Average misclassification error [%] for several methods on the MTPV62 benchmark (Li et al. 2013). Results for trajectory clustering approaches are copied from Xu et al. (2018)

| GPCA Vidal et al. (2005) | ALC Rao et al. (2010) | SSC Elhamifar and Vidal (2013) | TPV Li et al. (2013) | MSSC Lai et al. (2017) | KerAdd Xu et al. (2018) | Coreg Xu et al. (2018) | Subset Xu et al. (2018) | SYNCH Arrigoni and Pajdla (2019a) | Baseline | MODE | MODE-G | MODE-S | MODE-E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16.58 | 14.88 | 5.17 | 2.37 | **0.65** | 0.88 | 0.73 | **0.65** | 8.67 | 8.79 | 4.11 | 3.90 | 4.03 | 4.03 |

The lowest errors are highlighted in bold face

a random permutation. This scenario resembles unordered image collections (e.g. in multibody structure from motion) where errors are ubiquitous among two-frame correspondences. For each configuration, we repeat the test 10 times and report averaged results.

As detailed in Section 6.1, the most relevant competitors are SYNCH (Arrigoni and Pajdla 2019a) and the BASELINE, which – as our method – address motion segmentation with two-frame correspondences. In particular, they take the *same input* as MODE, that is a set of two-frame segmentations computed with RPA (Magri and Fusiello 2015). We also include in the comparison two methods performing trajectory clustering, namely RSIM (Ji et al. 2015) and Subset (Xu et al. 2018), although not directly comparable to MODE. Observe that, in order to run such methods on our synthetic data, we need to compute trajectories from two-frame correspondences. We consider two different techniques to accomplish such a task, namely StableSfM[6] (Olsson and Enqvist 2011) and QuickMatch[7] (Tron et al. 2017).

We measure performance in terms of misclassification error, which is defined here as the percentage of misclassified points over the total amount of classified points. In other words, in contrast to Section 6.2, we evaluate segmentation results considering only points with a nonzero label (i.e., points with zero label do not contribute to the error). Indeed, due to the presence of mismatches, one may not expect to give a valid label to all the image points, as observed in Remark 1. We also compute the percentage of points classified by each method.

Results are reported in Figure 13, which clearly shows the robustness to mismatches gained by our approach. In particular, it is remarkable that the error remains about 0% until 60% of mismatches in the *cars1* sequence. MODE is comparable to SYNCH and significantly better than the BASELINE in terms of misclassification error, and it classifies more points than its closest competitors. The low amount of data labelled by the BASELINE can be explained by observing that it uses results from a tree only, whereas both MODE and SYNCH exploit all the available image pairs (which are *redundant*) in order to produce the final segmentation.

Concerning trajectory clustering methods, it was found by inspecting the solution that Subset and RSIM actually classify all the tracks, and unclassified data correspond to image points that were not included in any track by the algorithm used for computing trajectories. Such approaches achieve a low misclassification error only when mismatches are below 10% and performances degrade with increasing ratio of mismatches. Indeed, wrong correspondences propagate into the tracks making trajectory clustering hard to solve. Notice that

[6] http://www.maths.lth.se/matematiklth/personal/calle/sys_paper/sys_paper.html.

[7] https://bitbucket.org/tronroberto/quickshiftmatching.

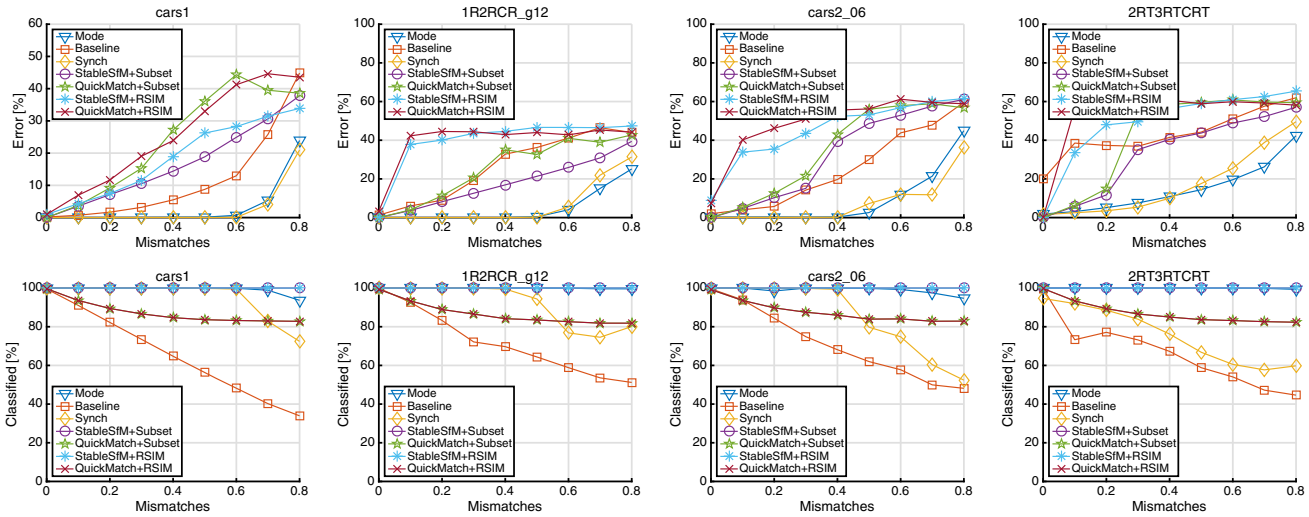



**Fig. 13** Misclassification error [%] and classified points [%] versus fraction of mismatches for several methods on four sequences from the Hopkins155 dataset (Tron and Vidal 2007). In this experiment, synthetic errors are introduced among two-frame correspondences (Color figure online)

**Table 6** The category of the scene, the number of motions $d$, the number of images $n$, and the total number of image points $p$ are reported for four sequences from Hopkins155 (Tron and Vidal 2007)

| Dataset | Category | $d$ | $n$ | $p$ |
|---|---|---|---|---|
| *1R2RCR_g12* | *checkerboard* | 2 | 24 | 3672 |
| *2RT3RTCRT* | *checkerboard* | 3 | 23 | 8211 |
| *cars1* | *traffic* | 2 | 20 | 6140 |
| *cars2_06* | *traffic* | 3 | 15 | 1845 |

a track can even contain points of different motions, in which case errors in the output segmentation appear by assigning a unique label to the entire track. This clearly motivates the need of specific methods – such as the one proposed in this paper – for motion segmentation from raw pairwise matches.

In order to give further insights on segmentation with two-frame correspondences, we provide some analysis that illustrates the behaviour of RPA (Magri and Fusiello 2015) – that produces the input to MODE, SYNCH and the BASELINE– as a function of the ratio of mismatches. More precisely, we evaluate the ability of RPA to detect errors in the original correspondences. We consider the *false positive rate*, that is the fraction of good matches erroneously classified as outliers, and the *true positive rate*, that is the fraction of wrong matches correctly classified as outliers, where outliers are correspondences with zero labels and inliers are correspondences with a nonzero label (regardless of the class). We also consider the *precision*, that is the fraction of good matches among the ones classified as inliers, which gives an idea about the effective amount of mismatches that survive after performing two-frame segmentation. These statistics are reported in Figure 14, which shows that RPA is robust to errors among correspondences, without presenting significative differences between the analysed sequences. In particular, it is worth noting that the true positive rate and the precision remain above 80%, while the true positive rate remains below 10% with up to 50% of mismatches.

Despite performances of RPA are generally good, some mismatches still remain, which may influence our approach. Indeed, correspondences are taken into account in Equation (4). Also false positives may influence our method, since they reduce the amount of nonzero labels that are used in Equation (6). In addition to this, RPA may not correctly segment some points since it lacks theoretical guarantees, thus producing errors in individual two-frame segmentations. This aspect is illustrated in Figure 15, which reports the histograms of misclassification error achieved by RPA over all the image pairs. As expected, the histograms shift to the right as the percentage of input mismatches increases. Note that RPA produces errors even in the absence of wrong correspondences (see the left histogram in Figure 15b). To sum up, Figure 14 together with Figure 15 give an idea about how hard it is to solve motion segmentation *given* results of two-frame segmentation. Let us consider, for instance, the second to last histogram in Figure 15a, which corresponds to 60% of mismatches in *cars1*. It is worth noting that, despite individual two-frame segmentations are noisy, our method achieves zero error, as shown in Figure 13. In other words, MODE is able to successfully solve motion segmentation while reducing errors in the two-frame segmentations, thanks to the fact that it exploits redundant measures in a principled manner.

Finally, we provide further analysis illustrating what happens to individual points when running our method. Figure 16 reports coloured bars representing the amount of errors (red), correct labels (green) and unknown labels (blue) for each point in a sample image from the *cars1* sequence. As the percentage of wrong correspondences increases, motion seg-

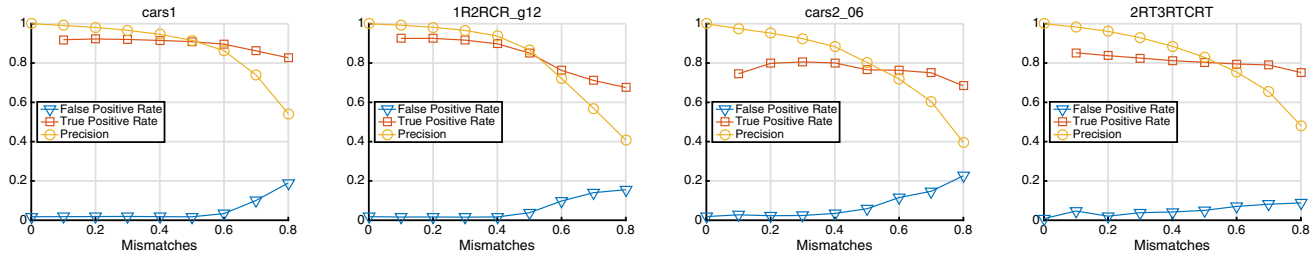



**Fig. 14** False Positive Rate, True Positive Rate and Precision achieved by RPA (Magri and Fusiello 2015) versus fraction of mismatches on four sequences from Hopkins155 (Tron and Vidal 2007). In this experiment, synthetic errors are introduced among two-frame correspondences (Color figure online)

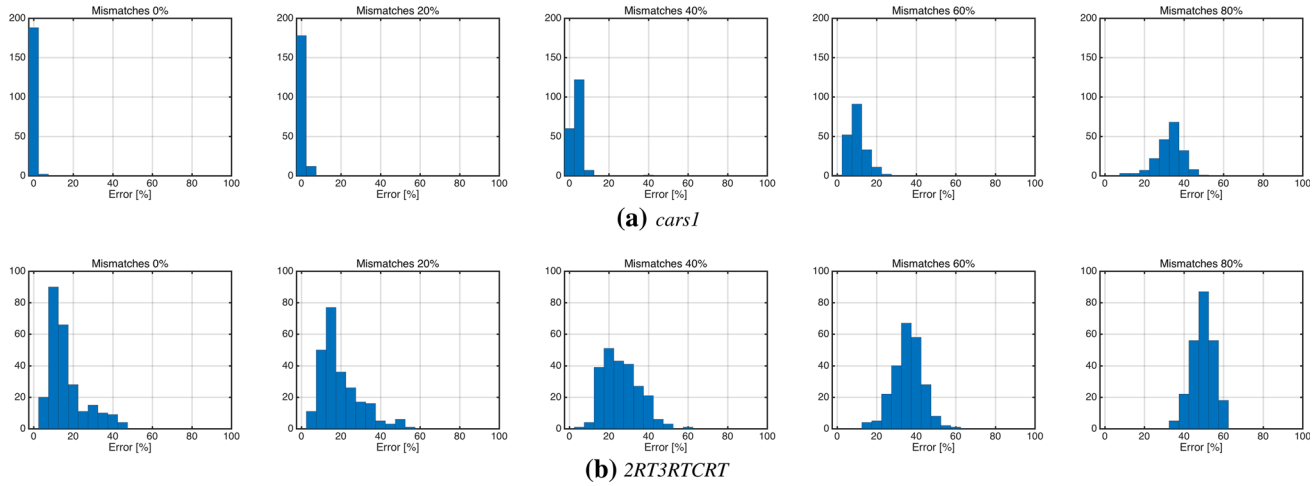



**Fig. 15** Histograms of misclassification error achieved by RPA (Magri and Fusiello 2015) on two sequences from Hopkins155 (Tron and Vidal 2007). In this experiment, synthetic errors are introduced among two-frame correspondences and a single trial is considered. The horizontal axis corresponds to the misclassification error in an individual image pair, and the vertical axis corresponds to the number of pairs where a given error is obtained

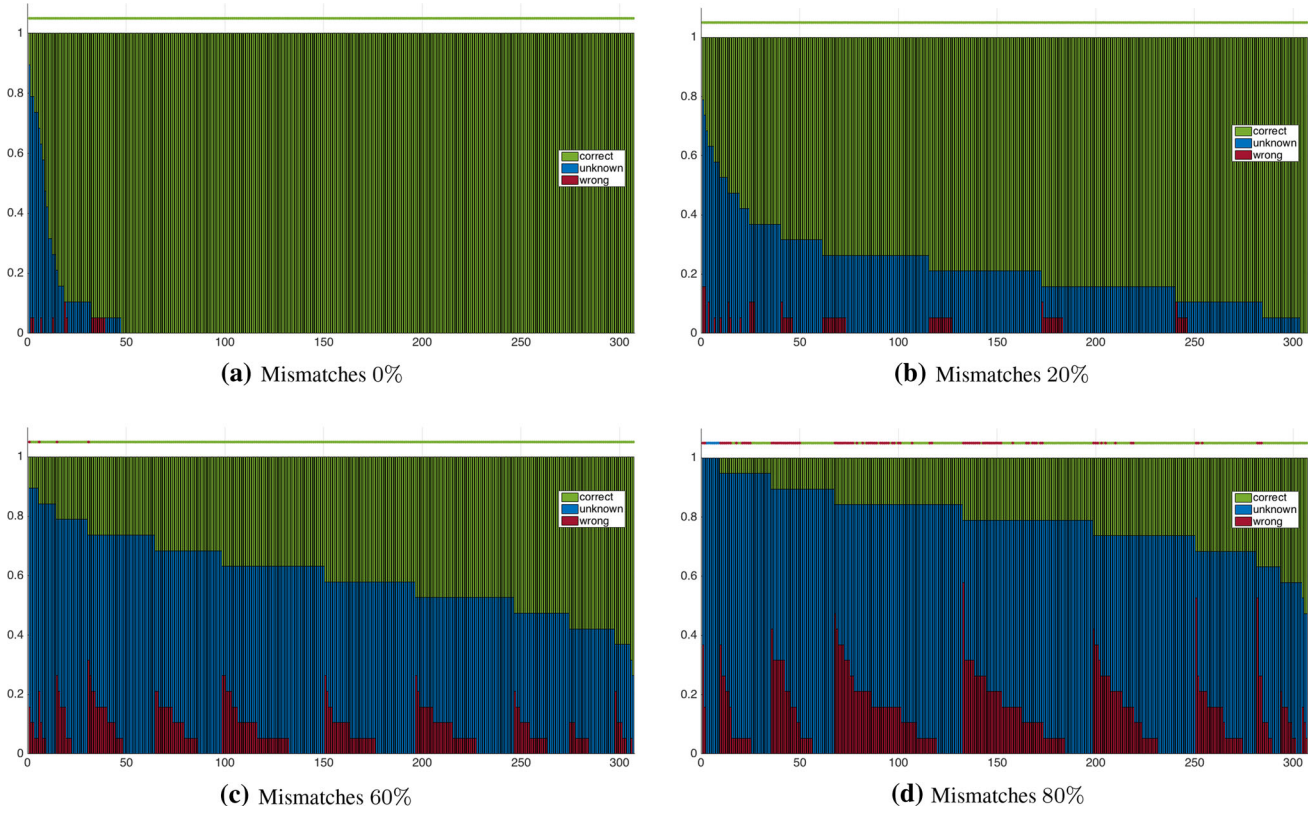
mentation becomes more difficult to solve, since the green area reduces whereas the blue and red ones enlarge. Note that RPA (Magri and Fusiello 2015) produces errors even in the absence of mismatches, as shown in Figure 16a. Our approach classifies all the data except for a few cases where the blue bars are equal to 1, meaning that the point is labelled as outlier by RPA in all the pairs. Among the classified points, MODE provides a correct segmentation as long as the green bars are sufficiently high.

## 6.5 Indoor Scenes

In order to evaluate the performance of our approach on real data, we consider the benchmark proposed in (Arrigoni and Pajdla (2019b, a)), which provides image points with ground-truth labels and noisy two-frame correspondences (obtained with SIFT Lowe 2004). Alternative matches (Bian et al. 2017) have been tested with similar results. The dataset provides 12 indoor scenes with two or three motions counting from 6 to 10 images. Observe that this benchmark is specific for motion segmentation with two-frame correspondences, which is the focus of our paper.

As in Section 6.4, we compare MODE with SYNCH (Arrigoni and Pajdla 2019a) and the BASELINE, which take the same input as our approach, namely the results from two-frame segmentation (obtained with RPA (Magri and Fusiello 2015)). We also consider two trajectory clustering methods, namely RSIM (Ji et al. 2015) and Subset (Xu et al. 2018), where StableSfM (Olsson and Enqvist 2011) and Quick-Match (Tron et al. 2017) are used to compute tracks from two-frame correspondences.

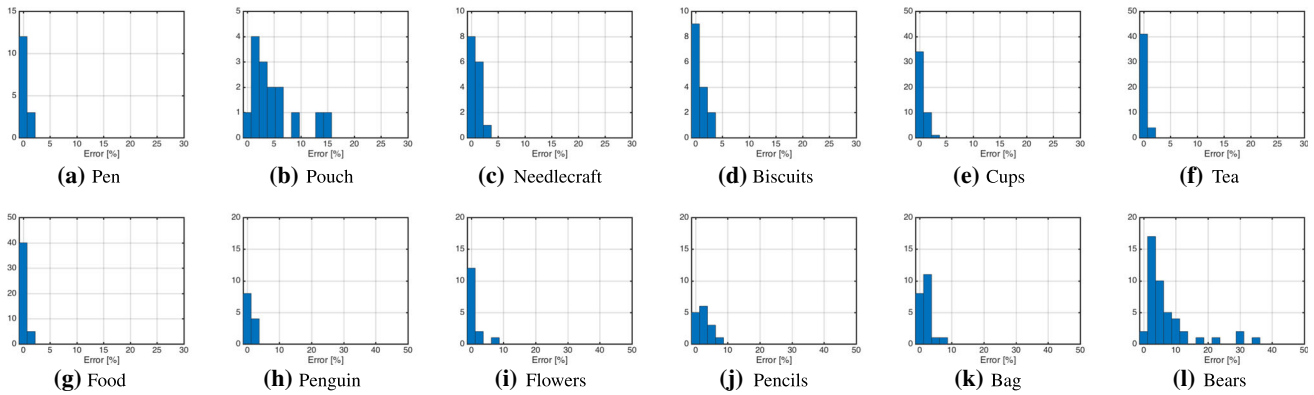
Results are given in Table 7, which reports both the misclassification error – defined as the percentage of misclassified points over the total amount of classified points – and the percentage of points labelled by each method. See also Figures 1 and 18 for qualitative evaluations. There are no significant differences between MODE and the BASELINE in terms of misclassification error, however, the former is superior in terms of the percentage of classified points since it exploits *redundant* two-frame segmentations. Both our method and the BASELINE– with a misclassification error




**(a)** Mismatches 0% **(b)** Mismatches 20%

**(c)** Mismatches 60% **(d)** Mismatches 80%

**Fig. 16** The horizontal axis indexes points in a sample image from *cars1* (Tron and Vidal 2007) and a three-color bar is shown for each point. Bars are divided into three parts which sum to one. The green, red, and blue parts represent fractions of image pairs where the point is correctly classified, misclassified, and labeled as outlier, respectively, by RPA (Magri and Fusiello 2015). For better visualization, points are sorted increasingly by the height of green bars. A dot is plotted over each bar to show whether the point is classified by our method correctly (green), misclassified (red) or labelled as unknown (blue) (Color figure online)




**(a)** Pen **(b)** Pouch **(c)** Needlecraft **(d)** Biscuits **(e)** Cups **(f)** Tea

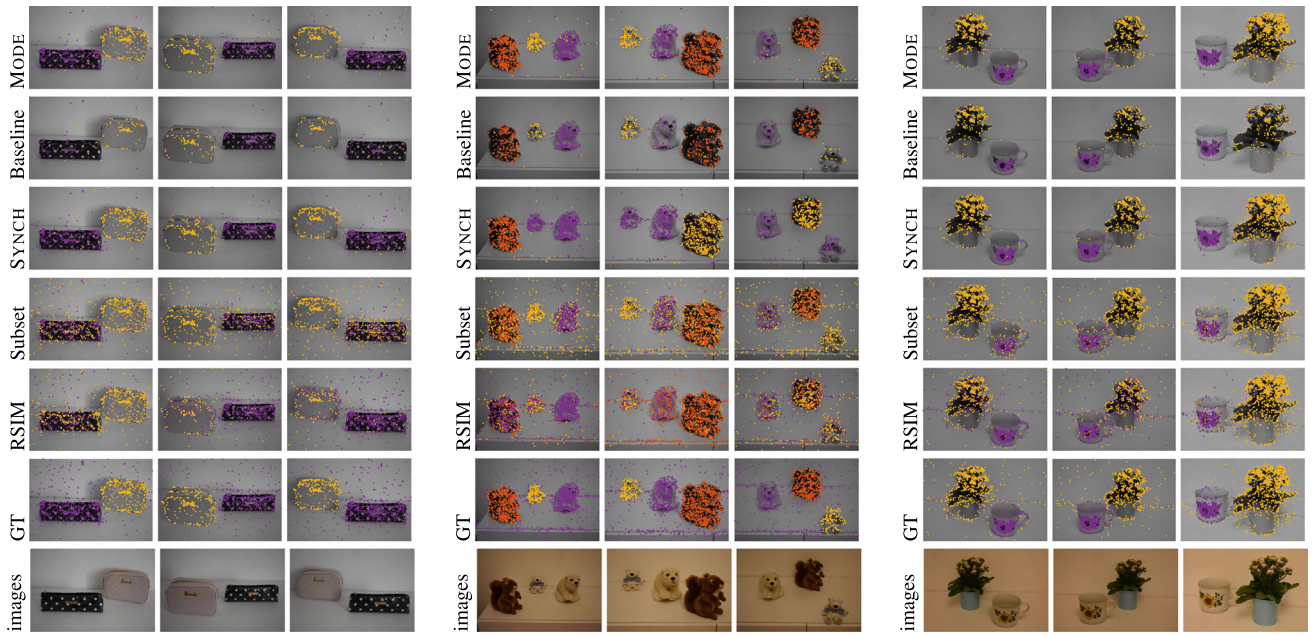
**(g)** Food **(h)** Penguin **(i)** Flowers **(j)** Pencils **(k)** Bag **(l)** Bears

**Fig. 17** Histograms of misclassification error achieved by RPA (Magri and Fusiello 2015) on indoor scenes (Arrigoni and Pajdla 2019b, a). Each point in the horizontal axis corresponds to a possible misclassification error in an individual image pair, and each point in the vertical axis corresponds to the number of pairs where a given error is reached

**Table 7** Misclassification error [%] and classified points [%] for several methods on indoor scenes (Arrigoni and Pajdla 2019b, a). The number of motions $d$, the number of images $n$, and the total number of image points $p$ are also reported for each sequence

| Dataset | $d$ | $n$ | $p$ | MODE | | BASELINE | | SYNCH Arrigoni and Pajdla (2019a) | | StableSfM + Subset Xu et al. (2018) | | QuickMatch + Subset Xu et al. (2018) | | StableSfM + RSIM Ji et al. (2015) | | QuickMatch + RSIM Ji et al. (2015) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Error | Classified | Error | Classified | Error | Classified | Error | Classified | Error | Classified | Error | Classified | Error | Classified |
| *Pen* | 2 | 6 | 4550 | **0.58** | 80.07 | **0.58** | 50.37 | 0.82 | 83.23 | 17.12 | 99.36 | 14.57 | 82.07 | 13.94 | 99.36 | 12.13 | 82.07 |
| *Pouch* | 2 | 6 | 4971 | 3.79 | 65.34 | **2.62** | 31.52 | 4.15 | 69.89 | 26.14 | 99.60 | 24.09 | 66.12 | 32.60 | 99.60 | 37.30 | 66.12 |
| *Needlecraft* | 2 | 6 | 6617 | 0.83 | 72.81 | **0.63** | 38.19 | 1.04 | 76.76 | 19.13 | 99.61 | 17.97 | 72.51 | 23.58 | 99.61 | 26.84 | 72.51 |
| *Biscuits* | 2 | 6 | 13158 | 0.47 | 84.47 | **0.26** | 51.85 | 0.51 | 87.28 | 9.22 | 99.47 | 8.91 | 82.49 | 4.58 | 99.47 | 34.87 | 82.49 |
| *Cups* | 2 | 10 | 14664 | 0.56 | 65.42 | **0.30** | 39.17 | 1.01 | 69.82 | 12.53 | 99.29 | 12.97 | 74.75 | 22.78 | 99.29 | 33.09 | 74.75 |
| *Tea* | 2 | 10 | 32612 | **0.29** | 81.70 | 0.47 | 47.24 | 28.12 | 52.21 | 7.11 | 99.37 | 5.46 | 82.67 | 46.98 | 99.37 | 41.99 | 82.67 |
| *Food* | 2 | 10 | 36723 | **0.36** | 76.19 | 0.61 | 39.48 | 0.56 | 80.66 | 12.86 | 99.32 | 13.83 | 72.85 | 19.18 | 99.32 | 20.38 | 72.85 |
| *Penguin* | 2 | 6 | 5865 | **0.76** | 69.17 | 0.95 | 33.95 | 44.21 | 46.97 | 32.27 | 99.59 | 41.05 | 70.11 | 41.50 | 99.59 | 41.54 | 70.11 |
| *Flowers* | 2 | 6 | 7743 | **1.23** | 73.65 | 2.84 | 32.70 | 1.62 | 77.28 | 8.55 | 99.50 | 8.59 | 72.59 | 16.65 | 99,50 | 14.20 | 72.59 |
| *Pencils* | 2 | 6 | 2982 | 3.80 | 65.33 | **2.30** | 30.65 | 27.53 | 40.44 | 41.46 | 99.56 | 40.88 | 66.36 | 23.07 | 99.56 | 23.45 | 66.36 |
| *Bag* | 2 | 7 | 6114 | **1.52** | 57.95 | 1.54 | 26.56 | 25.92 | 54.27 | 14.22 | 99.69 | 15.67 | 65.85 | 34.55 | 99.69 | 39.92 | 65.85 |
| *Bears* | 3 | 10 | 15888 | 4.82 | 73.65 | **2.72** | 29.80 | 38.95 | 74.59 | 38.13 | 99.58 | 35.21 | 63.12 | 49.48 | 99.58 | 53.80 | 63.12 |

The lowest errors are highlighted in bold face




**(a)** *Pouch* **(b)** *Bears* **(c)** *Flowers*

**Fig. 18** Segmentation results are reported for several methods on sample images from indoor scenes (Arrigoni and Pajdla 2019b, a). Different colours encode the membership to different motions, whereas unclassified points are not drawn (for better visualization). Concerning Subset (Xu et al. 2018) and RSIM (Ji et al. 2015), trajectories are computed with StableSfM. Raw images and ground-truth (GT) segmentation are also reported (Color figure online)

lower than 5% in all the sequences – are significantly better than SYNCH, Subset and RSIM. Trajectory clustering methods exhibit poor performances on this scenario since they do not deal with mismatches, confirming the outcome of the experiments on synthetic data. Observe that SYNCH, although being accurate on most cases, fails on 5 out of 12 sequences. According to the analysis in Arrigoni and Pajdla (2019a), the cause may be a small spectral gap.

We also test the method developed in Ji et al. (2014), which does not require pairwise matches but feature locations and descriptors only. In other words, it addresses motion segmentation with unknown correspondences (see Section 2.4.3). We ran the available Matlab implementation of Ji et al. (2014) on the *Pencils* sequence, but it did not return any solution after several hours of computation due to "out of memory" error. We conclude that it does not represent a practical approach to motion segmentation on the scenarios considered in our paper.

In order to give further insights on the behavior of our technique, we report in Figure 17 the histograms of misclassification error achieved by RPA (Magri and Fusiello 2015) over image pairs, similarly to the synthetic experiments in Section 6.4. The histograms show the effective amount of corruption in the data after performing two-frame segmentation with RPA, which is the first step of our pipeline. Note that the misclassification error exceeds 30% in some image pairs from the *Bears* sequence. It is noticeable that our approach achieves a low error on this scene (about 4.8%), as reported in Tab. 7. In other words, MODE can effectively reduce errors in the pairwise segmentations thanks to the fact that it exploits redundant measures.

### 6.5.1 Spatial Refinement

Table 7 and Figure 18 show that our method produces a segmentation of high quality in all the sequences. Such results can be further improved by employing a spatial refinement, which encourages neighbouring points to have the same label. We evaluate three different solutions for such a task, which are detailed in Section 5, namely a greedy approach, constrained spectral clustering and energy minimization. Recall that such techniques can be regarded as a post-processing to be applied to the output of MODE, thus we get three different variants of our method:

- MODE-G (our method + greedy approach)
- MODE-S (our method + constrained spectral clustering)
- MODE-E (our method + energy minimization).

In our experiments we use $\tau = 0.7$, $\epsilon = 200$ pixels, $\delta = 0.3$, $\gamma = 5$ and $k = 10$. We refer the reader to Section 5 for more information on the meaning of such parameters.

Results are given in Table 8, which reports both the misclassification error and the percentage of classified points

**Table 8** Misclassification error [%] and classified points [%] for different variants of our approach on indoor scenes (Arrigoni and Pajdla 2019b, a)

| Dataset | MODE | | MODE-G | | MODE-S | | MODE-E | |
|---|---|---|---|---|---|---|---|---|
| | Error | Classified | Error | Classified | Error | Classified | Error | Classified |
| *Pen* | 0.58 | 80.07 | **0.38** | 79.87 | 0.60 | 80.07 | 0.58 | 80.07 |
| *Pouch* | 3.79 | 65.34 | **0.93** | 62.48 | 1.45 | 65.34 | 1.14 | 65.34 |
| *Needlecraft* | 0.83 | 72.81 | 0.64 | 72.64 | 0.68 | 72.81 | **0.60** | 72.81 |
| *Biscuits* | 0.47 | 84.47 | **0.09** | 84.12 | 0.13 | 84.47 | 0.13 | 84.47 |
| *Cups* | 0.56 | 65.42 | **0.13** | 65.04 | 0.34 | 65.42 | 0.39 | 65.42 |
| *Tea* | 0.29 | 81.70 | **0.12** | 81.52 | 0.17 | 81.70 | 0.17 | 81.70 |
| *Food* | 0.36 | 76.19 | **0.04** | 75.88 | 0.13 | 76.19 | 0.07 | 76.19 |
| *Penguin* | 0.76 | 69.17 | **0.40** | 68.63 | 0.69 | 69.17 | 0.94 | 69.17 |
| *Flowers* | 1.23 | 73.65 | 0.19 | 72.79 | 0.39 | 73.65 | **0.18** | 73.65 |
| *Pencils* | 3.80 | 65.33 | 1.11 | 63.21 | 2.31 | 65.33 | **1.03** | 65.33 |
| *Bag* | 1.52 | 57.95 | **0.74** | 56.85 | 1.66 | 57.95 | 1.52 | 57.95 |
| *Bears* | 4.82 | 73.65 | **1.93** | 70.75 | 3.48 | 73.65 | 2.79 | 73.65 |

The lowest errors are highlighted in bold face

for the aforementioned methods. As a reference, we also include results for MODE which are copied from Tab. 7. All the analyzed techniques are more accurate than MODE. However, such an improvement is marginal in most cases. MODE-G achieves the lowest errors in most sequences, but it also reduces the amount of classified points compared to MODE. Indeed, it can be regarded as an outlier removal which discards those points whose label is likely to be wrong, as already observed in Section 5.1. On the contrary, the amount of points classified by MODE-S or MODE-E does not change, with the latter being slightly better than the former in terms of accuracy. In conclusion, MODE-E can be viewed as a good trade-off between accuracy and amount of classified data, hence we elect MODE-E as our choice and we drop MODE-G and MODE-S in subsequent comparisons.

### 6.5.2 Multibody Structure from Motion

We now showcase the application of our method to multibody structure from motion (MBSfM) (Saputra et al. 2018), where the task is to recover both camera motion (i.e., angular attitudes and positions of the cameras) and scene structure (i.e., 3D coordinates of the points) from multiple images, for each independently moving object present in the scene. Observe that the MBSfM problem requires to solve both motion segmentation and traditional (i.e., static) structure from motion (SfM), either simultaneously or in a sequence. We follow the latter approach and sequentially combine our segmentation solution with COLMAP (Schonberger and Frahm 2016), which is a traditional SfM system with public code.[8] More precisely, we proceed as follows:

1. given a set of key-points over multiple images with two-frame correspondences, MODE-E is applied in order to group image points according to different motions;
2. for each motion, the following operations are performed:
   - only key-points belonging to the considered motion are used, together with two-frame correspondences between points within the same motion;
   - the data from the previous step are given as input to COLMAP (Schonberger and Frahm 2016), which returns both camera motion and a sparse 3D reconstruction of the moving object.

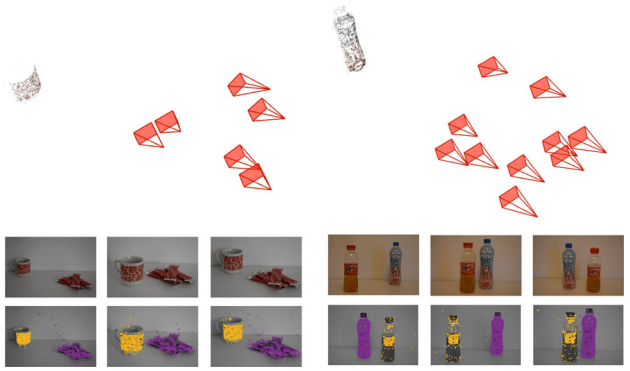
Observe that COLMAP (as any SfM pipeline) builds trajectories in order to connect the input two-frame correspondences across all the images. In other words, we are creating multi-frame correspondences *after* solving motion segmentation. Trajectory clustering methods, instead, require trajectories *before* motion segmentation (see Section 2.4.1). In general, it is harder to compute trajectories for a dynamic scene compared to the static case (as done by COLMAP, where geometric verification can be employed). This represents an advantage of the scenario considered in our paper, namely motion segmentation from pairwise matches.

Results are shown in Table 9, which reports the mean reprojection error of the proposed MBSfM pipeline, in addition to the number of reconstructed cameras and the number of trajectories (or, equivalently, the number of 3D points). See also Fig 19 for qualitative results. The *Pencils* sequence represents a failure case, as COLMAP is not able to produce a reconstruction for one out of the two motions, probably due to the fact that such motion contains very few points. In all the remaining sequences our pipeline successfully solves the

[8] https://colmap.github.io/index.html.

**Table 9** Our method combined with COLMAP (Schonberger and Frahm 2016) gives rise to a multibody structure from motion pipeline. The mean reprojection error [pixels], the number of reconstructed cameras and the number of 3D points are reported for each motion on indoor scenes (Arrigoni and Pajdla 2019b, a)

| Dataset | *Pen* | | *Pouch* | | *Needlecraft* | | *Biscuits* | | *Cups* | | *Tea* | | *Food* | | *Penguin* | | *Flowers* | | *Pencils* | | *Bag* | | *Bears* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Motion | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 1° | 2° | 3° |
| Error | 0.56 | 0.64 | 0.76 | 0.75 | 0.63 | 0.96 | 0.64 | 0.56 | 0.79 | 0.35 | 0.46 | 0.69 | 0.78 | 0.67 | 0.79 | 0.77 | 0.72 | 0.67 | 1.00 | – | 0.76 | 0.88 | 1.07 | 1.15 | 0.93 |
| Cameras | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 10 | 10 | 10 | 10 | 10 | 10 | 6 | 6 | 6 | 6 | 6 | – | 7 | 7 | 10 | 10 | 10 |
| 3D Points | 340 | 354 | 299 | 345 | 354 | 464 | 1604 | 1551 | 613 | 1398 | 4852 | 989 | 2783 | 3657 | 297 | 837 | 1135 | 470 | 336 | – | 441 | 676 | 1432 | 512 | 776 |



**(a)** *Biscuits* **(b)** *Tea*

**Fig. 19** Our method combined with COLMAP (Schonberger and Frahm 2016) gives rise to a multibody structure from motion pipeline. The reconstructed cameras and a sparse 3D reconstruction are reported for two indoor scenes (Arrigoni and Pajdla 2019b, a), where only one motion is considered. At the bottom, some sample images are reported together with the segmentation results produced by MODE-E, where different colors correspond to different motions (Color figure online)

MBSfM problem with high accuracy, as all the cameras get reconstructed and the mean reprojection error is lower than 1 pixel in most cases.

### 6.6 Outdoor Scenes

We conclude our experiments by analyzing some outdoor scenes with two motions, namely *helicopter* (Dragon et al. 2013), *boat* (Li et al. 2013), *van* (Li et al. 2013), *cars7* (Tron and Vidal 2007) and *cars8* (Tron and Vidal 2007). Such datasets are typically used for trajectory clustering. However, in order to study motion segmentation with two-frame correspondences, we consider the images only (discarding trajectories when available), and we compute the input matches by ourselves.

We also consider some sequences from the DAVIS dataset (Perazzi et al. 2016; Pont-Tuset et al. 2017; Caelles et al. 2019), although being specific for video object segmentation (we refer the reader to Section 2.5 for more information on video object segmentation.). Observe that most DAVIS sequences involve highly articulated/non-rigid motions which violate our assumptions, hence are not considered. However, such a dataset also contains a few instances of *rigid* scenes which are useful for our analysis, namely:

- *bus*, *scooter-gray*, *scooter-black*, *car-turn*, *car-shadow*, *car-roundabout*, *train*, *bmx-bumps*, and *blackswan* (taken from DAVIS 2016 (Perazzi et al. 2016));
- *classic-car* (taken from DAVIS 2017 (Pont-Tuset et al. 2017));
- *landing* and *tram* (taken from DAVIS 2019 (Caelles et al. 2019));

For each sequence, we choose a subset of the images in order to ensure enough motion between consecutive frames. We extract SIFT keypoints (Lowe 2004) in all the images and establish correspondences between image pairs using the nearest neighbor and ratio test as in Lowe (2004) via the VLFeat library.[9] For each pair $(i, j)$, we keep only those correspondences that are found both when matching image $i$ with $j$ and when matching image $j$ with $i$, and isolated key-points (i.e. points that are not matched in any image) are discarded. No further filtering is applied.

The properties of each dataset are presented in Tab. 10, which also reports the percentage of points classified by MODE, MODE-E, the BASELINE, SYNCH (Arrigoni and Pajdla 2019a) and Subset (Xu et al. 2018) combined with StableSfM (Olsson and Enqvist 2011). The latter provides the best results among all possible combinations of trajectory clustering methods and tracking algorithms. In the case of the *helicopter* sequence, ground-truth pixel-wise annotation is available for a subset of the images, which can be used to compute the misclassification error (see Table 10). Also sequences taken from DAVIS 2016 (Perazzi et al. 2016) and DAVIS 2017 Pont-Tuset et al. (2017) come with pixel-wise annotations. For the remaining sequences there is no ground-truth, so only qualitative evaluation can be provided, which is given in Figure 20.

Visual results in Figure 20 show that our solution is of good quality in all the images, with the spatial refinement being particularly effective in the *boat* and *van* sequences (see Figure 20d and 20e ). Both MODE and MODE-E outperform the BASELINE in terms of amount of classified data. This is particularly evident in the right column of Figure 20a where the BASELINE is not able to classify any point in the moving object. The poor performance of the BASELINE gives an idea about how noisy the individual two-frame segmentations are. Our method is able to reduce such errors since it exploits redundant measures. Observe that SYNCH produces results of poor quality in most cases, thus it does not represent a practical solution to motion segmentation on the outdoor scenes considered in this section. There are no significant differences between Subset and MODE in the *boat* and *van* sequences, which, however, are simple scenes for matching due to slow motion. In the *helicopter*, *cars7* and *cars8* sequences, Subset produces useless results.

As concerns the DAVIS sequences, Table 10 shows that MODE-E is the most accurate solution in 9 out of 10 cases, outperforming the competiting methods. Observe that the spatial refinement always improves the MODE results, and it is particularly effective in the *car-roundabout* scene with a gain of about 7% of misclassification error. Figure 21 reports an example in order to visually appreciate this aspect. Further qualitative results are given in Figure 20f and 20g where it can be observed that our solution is visually appealing and generally better than the competitors (to different extents).

[9] http://www.vlfeat.org/.

### 6.7 Dealing with an Unknown Number of Motions

In this paper we focus on the scenario where the number of motions is known and constant over frames. In this section, we give some insights about how to handle the case of an unknown number of motions. More analysis on this aspect can be found in a preliminary study (Arrigoni et al. 2020a).

First, let us recall the main steps of MODE:

- motion segmentation is independently solved on different image pairs (*two-frame segmentation*);
- the partial/local results produced by two-frame segmentation are combined in order to return a *multi-frame segmentation*: this is done by permutation synchronization (which fixes the permutation ambiguity) and robust voting (which handles noise).

It is easy to see that the stages influenced by an unknown number of motions are two-frame segmentation and permutation synchronization only. Indeed, robust voting works under any assumptions. This suggests that, in order to extend MODE to the scenario of an unknown number of motions, it is enough to substitute the methods used for two-frame segmentation and permutation synchronization:

- two-frame segmentation can be addressed by fitting an *unknown* number of fundamental matrices to correspondences; several possibilities are available (see Section 2.3), such as T-Linkage (Magri and Fusiello 2014), which handles an unknown number of motions thanks to a hierarchical clustering framework;
- permutation synchronization with an *unknown* number of motions can be addressed by combining MatchEIG (Maset et al. 2017) with QuickMatch (Tron et al. 2017) (see Appendix B); observe that the involved permutation matrices may be *partial*: indeed, different two-frame segmentations can have a different number of motions (which in turn happens when the number of objects is not constant over frames or when T-Linkage estimates a wrong number of motions in some image pairs).

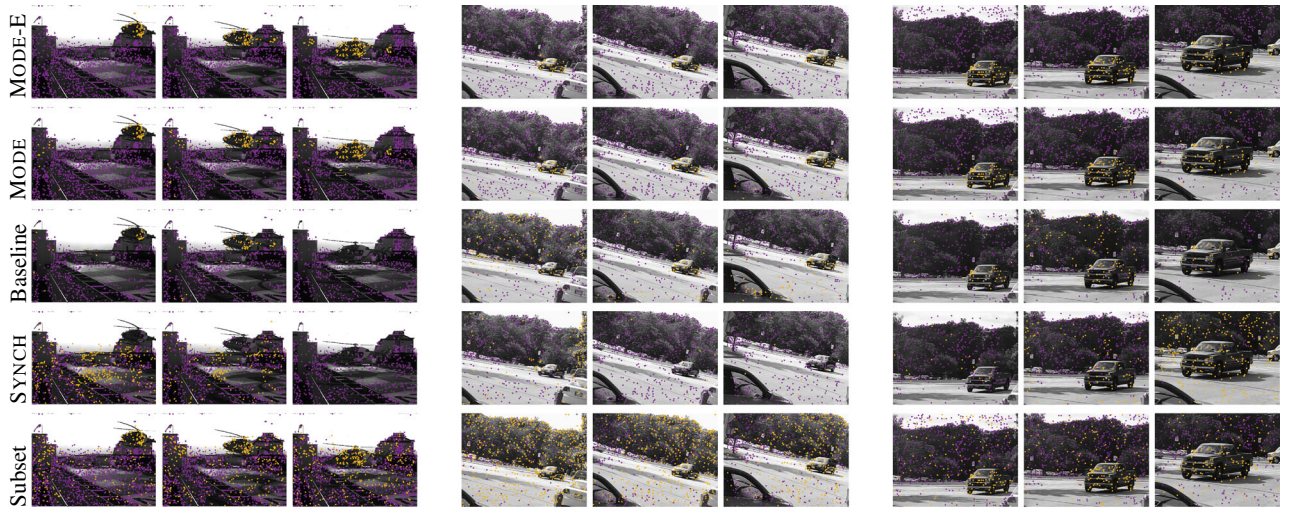
The resulting method has been named MODE-U in Arrigoni et al. (2020a), where "U" stands for "unknown" number of motions: it follows the same structure as MODE (detailed in Figure 7), except for the methods used for two-frame segmentation and permutation synchronization (see Table 11).

According to the experiments reported in Arrigoni et al. (2020a) (that are copied in Table 12 as a reference), MODE-U (which automatically estimates the number of motions) is

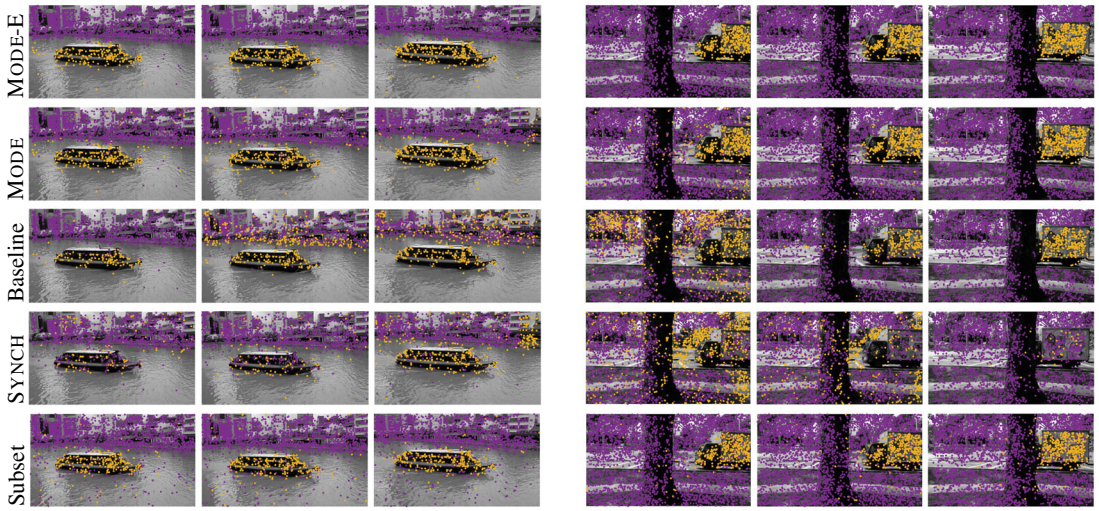
**Table 10** Misclassification error [%] and classified points [%] for several methods on outdoor scenes. The number of motions $d$, the number of images $n$, and the total number of image points $p$ are also reported for each sequence

| Dataset | $d$ | $n$ | $p$ | MODE | | MODE-E | | SYNCH | | BASELINE | | StableSfM + Subset Xu et al. (2018) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Error | Classified | Error | Classified | Error | Classified | Error | Classified | Error | Classified |
| *classic-car* Pont-Tuset et al. (2017) | 2 | 10 | 11339 | 6.06 | 93.22 | **5.05** | 93.22 | 29.89 | 63.04 | 8.37 | 64.22 | 10.09 | 99.65 |
| *train* Perazzi et al. (2016) | 2 | 10 | 14809 | 1.84 | 93.49 | **1.31** | 93.49 | 18.45 | 46.75 | 4.43 | 73.90 | 2.51 | 99.75 |
| *bmx-bumps* Perazzi et al. (2016) | 2 | 10 | 14303 | 3.83 | 94.26 | **3.19** | 94.26 | 38.91 | 86.77 | 4.53 | 63.49 | 4.31 | 99.69 |
| *blackswan* Perazzi et al. (2016) | 2 | 10 | 13178 | 3.94 | 89.29 | **3.20** | 89.29 | 17.92 | 54.15 | 3.49 | 67.75 | 4.79 | 99.82 |
| *bus* Perazzi et al. (2016) | 2 | 10 | 11362 | 2.40 | 94.85 | **1.66** | 94.85 | 26.10 | 58.18 | 4.59 | 58.14 | 3.32 | 99.41 |
| *scooter-gray* Perazzi et al. (2016) | 2 | 10 | 9631 | 4.88 | 92.57 | **3.63** | 92.57 | 22.44 | 62.74 | 6.55 | 62.40 | 5.34 | 99.63 |
| *scooter-black* Perazzi et al. (2016) | 2 | 10 | 13656 | 7.51 | 90.69 | 5.26 | 90.69 | 39.90 | 51.09 | 16.74 | 59.94 | **4.14** | 99.66 |
| *car-turn* Perazzi et al. (2016) | 2 | 10 | 12154 | 2.96 | 93.57 | **2.03** | 93.57 | 23.75 | 59.55 | 3.77 | 67.64 | 2.84 | 99.75 |
| *car-shadow* Perazzi et al. (2016) | 2 | 10 | 9170 | 5.97 | 90.24 | **2.94** | 90.24 | 21.16 | 51.68 | 18.14 | 61.88 | 3.97 | 99.66 |
| *car-roundabout* Perazzi et al. (2016) | 2 | 10 | 11470 | 11.14 | 89.61 | **4.47** | 89.61 | 13.14 | 67.87 | 15.59 | 66.94 | 13.78 | 99.68 |
| *helicopter* Dragon et al. (2013) | 2 | 10 | 17139 | 2.01 | 80.82 | **0.65** | 80.82 | 28.31 | 56.99 | 0.78 | 45.93 | 16.81 | 99.52 |
| *boat* Li et al. (2013) | 2 | 10 | 21183 | – | 87.34 | – | 87.34 | – | 67.98 | – | 56.31 | – | 99.62 |
| *van* Li et al. (2013) | 2 | 10 | 37173 | – | 93.54 | – | 93.54 | – | 70.95 | – | 59.62 | – | 99.66 |
| *cars7* Tron and Vidal (2007) | 2 | 21 | 16602 | – | 92.27 | – | 92.27 | – | 66.78 | – | 57.38 | – | 99.66 |
| *cars8* Tron and Vidal (2007) | 2 | 19 | 13438 | – | 93.12 | – | 93.12 | – | 63.48 | – | 50.53 | – | 99.61 |
| *landing* Caelles et al. (2019) | 2 | 10 | 10978 | – | 80.71 | – | 80.71 | – | 58.13 | – | 65.06 | – | 99.69 |
| *tram* Caelles et al. (2019) | 2 | 10 | 9684 | – | 87.12 | – | 87.12 | – | 36.49 | – | 60.04 | – | 99.60 |

The lowest errors are highlighted in bold face



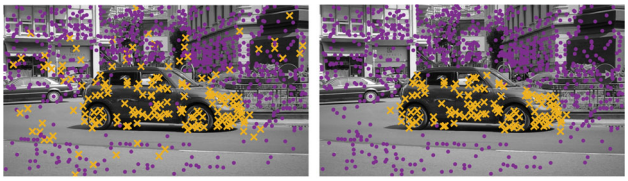

**(a)** *helicopter* (Dragon et al. 2013) **(b)** *cars7* (Tron and Vidal, 2007) **(c)** *cars8* (Tron and Vidal, 2007)




**(d)** *boat* (Li et al 2013) **(e)** *van* (Li et al 2013)


**(f)** *tram* (Caelles et al. 2019) **(g)** *landing* (Caelles et al. 2019)

**Fig. 20** Segmentation results are reported for several methods on sample images from outdoor scenes. Different colours encode the membership to different motions. For better visualization, unclassified points are not drawn (Color figure online)



**(a)** MODE **(b)** MODE-E

**Fig. 21** Segmentation results are reported for our approach before (left) and after (right) the spatial refinement, on a sample image from *car-roundabout* (Perazzi et al. 2016). Different colors correspond to different motions. In order to better appreciate the differences, points belonging to the car are drawn with a cross (Color figure online)

**Table 11** MODE and MODE-U (Arrigoni et al. 2020a) follow the segmentation pipeline in Figure 7: the former requires as input the correct number of motions, whereas the latter deals with an unknown number of motions. Accordingly, MODE and MODE-U use different solutions for two-frame segmentation and permutation synchronization

| | Two-frame segmentation | Permutation synchronization |
|---|---|---|
| MODE | RPA Magri and Fusiello (2015) | Spectral Pachauri et al. (2013) |
| MODE-U Arrigoni et al. (2020a) | T-Linkage Magri and Fusiello (2014) | MatchEIG Maset et al. (2017) + QuickMatch Tron et al. (2017) |

comparable in accuracy to MODE (which assumes the correct number of motions as input) on indoor data (Arrigoni and Pajdla 2019b, a). This shows that it is possible to generalize our approach to work under more difficult/practical assumptions. We refer the reader to Arrigoni et al. (2020a) for more details on this aspect.

## 7 Discussion

In this section we recall the assumptions made by our approach and we report some considerations about its main advantages and limitations.

### 7.1 Assumptions

In this paper we addressed the motion segmentation problem, where the task is to detect moving objects in multiple images by clustering together all the key-points that are undergoing the same motion. We assumed that a set of two-frame correspondences was available as input. In addition, we assumed that the moving objects present in the scene were *rigid*. Although the proposed method works reasonably well on a few examples of deforming objects (namely the articulated/non-rigid scenes in Hopkins155 Tron and Vidal (2007)), in general, it is not supposed to work on this scenario as the fundamental matrix – which is used for two-frame segmentation – assumes a rigid scene (Hartley and Zisserman 2004). Extending our approach to the non-rigid case is an interesting aspect to investigate. Accordingly, we plan to adapt our method to solve video object segmentation.

We designed our approach for the scenario where the number of motions is known and constant over frames and we performed our experiments under such an assumption, which – although restrictive – is common in motion segmentation literature (e.g., Ji et al. 2015; Xu et al. 2018). Notwithstanding this, we also explained that – with minor modifications – our approach can be extended to the case of an *unknown* number of motions (see Section 6.7), which is more realistic and hence more relevant for practical applications.

### 7.2 Advantages

First of all, our method has the advantage of working under *weaker* assumptions than the majority of works in motion segmentation literature, as it requires a set of two-frame correspondences instead of multi-frame trajectories, as detailed in Figure 3. In this specific setting, our approach achieves superior results than the state of the art, especially in situations where matches are noisy and contaminated by outliers.

Recall that the first stage of our method is solving segmentation on different image pairs independently. This approach has two main advantages: first, the problem is splitted into subproblems that are easier to solve; secondly, by leveraging on multiple pairs, *redundant* estimates are obtained, which are the key to achieve robustness.

The power of our two-frame approach is particularly evident in multibody structure from motion, which is our target application. The fact that we are solving motion segmentation at the earliest stage implies that *single-body* techniques can be exploited for the subsequent reconstruction stages: in particular, geometric verification (e.g. RANSAC) can be used to refine and join correspondences into trajectories *for each motion*. Relying on single-body techniques (which is equivalent to consider a static scene) makes the reconstruction easier to solve compared to the case of multiple bodies.

Finally, our framework is *modular*, so it can be easily extended to more general situations. For instance, it can be generalized to the case of an unknown number of motions, as explained in Section 6.7. Moreover, our method can handle triplets instead of image pairs, as shown in a preliminary work (Arrigoni et al. 2020b), where the underlying model is the trifocal tensor in place of the fundamental matrix. Future research will investigate this direction.

### 7.3 Limitations

Being designed for motion segmentation with two-frame correspondences, our method returns sub-optimal results on

**Table 12** Misclassification error [%] and classified points [%] for our approach (MODE) and MODE-U (Arrigoni et al. 2020a) on indoor scenes (Arrigoni and Pajdla 2019b, a). The former requires as input the correct number of motions (denoted by $d$), whereas the latter returns an estimate of the number of motions (denoted by $\widehat{d}$)

| Dataset | d | MODE | | MODE-U Arrigoni et al. (2020a) | | |
|---|---|---|---|---|---|---|
| | | Error | Classified | Error | Classified | $\widehat{d}$ |
| *Pen* | 2 | **0.58** | 80.07 | 1.55 | 89.08 | 2 |
| *Pouch* | 2 | 3.79 | 65.34 | **1.39** | 60.79 | 2 |
| *Needlecraft* | 2 | **0.83** | 72.81 | 1.80 | 67.07 | 2 |
| *Biscuits* | 2 | **0.47** | 84.47 | 1.12 | 90.42 | 2 |
| *Cups* | 2 | **0.56** | 65.42 | 2.05 | 71.31 | 2 |
| *Tea* | 2 | **0.29** | 81.70 | 0.69 | 85.21 | 2 |
| *Food* | 2 | **0.36** | 76.19 | 0.78 | 82.34 | 2 |
| *Penguin* | 2 | **0.76** | 69.17 | 1.36 | 66.60 | 2 |
| *Flowers* | 2 | **1.23** | 73.65 | 1.51 | 75.50 | 2 |
| *Pencils* | 2 | 3.80 | 65.33 | **3.09** | 51.01 | 2 |
| *Bag* | 2 | **1.52** | 57.95 | 2.78 | 52.91 | 2 |
| *Bears* | 3 | 4.82 | 73.65 | **3.48** | 68.21 | 3 |

The lowest errors are highlighted in bold face

trajectory clustering tasks, as it does not exploit all the available information (see Sections 6.2 and 6.3 ). Observe also that a key property of our approach is robustness, which originates from the usage of a robust method for two-frame segmentation (RPA Magri and Fusiello 2015). Hence, it obtains suboptimal results when data are cleaned and outliers are not present (as happens, e.g., in the Hopkins dataset).

Furthermore, recall that our method heavily relies on an initial two-frame segmentation: although it can handle a considerable amount of errors among two-frame results (see Figure 17I for instance), it produces poor results when most two-frame segmentations are wrong. One example of a failure case can be appreciated in Figure 12a, where our method achieves a misclassification error higher than 30% in one sequence from Hopkins12. By manual inspection, it was found that RPA performed poorly in the majority of image pairs.

Finally, being based on the fundamental matrix, we expect our approach to fail in situations where the fundamental matrix is degenerate (e.g., pure rotation) or when it does not represent the most appropriate model (e.g., in the presence of planar structures). For example, if we run our method on the KT3DMoSeg benchmark (Xu et al. 2018), that is a motion segmentation dataset built upon KITTI (Geiger et al. 2012), then we get an average misclassification error of 17.87%. Observe that such a dataset is meant for trajectory clustering in autonomous driving, so it comprises degenerate motions (see (Xu et al. 2018) for more information). Future research will study the usage of alternative models for these scenarios.

## 8 Conclusion

We presented a new solution to rigid motion segmentation, where the task is to group sparse key-points in multiple images according to a number of motions. The proposed approach splits the problem in two steps. First, motion segmentation is independently solved on pairs of images. Then, such partial/local results are combined by permutation synchronization (which fixes the inherent permutation ambiguity) and robust voting (which handles potential errors). This general framework – combined with a robust solution to two-frame segmentation (e.g. RPA Magri and Fusiello 2015) – handles realistic situations such as the presence of mismatches that have been overlooked in previous work. Our segmentation results can be further improved by employing spatial constraints, thus encouraging neighbouring points to belong to the same motion. Our approach does not assume any temporal component (i.e., it works with unstructured/unordered datasets) and it does not require tracks as input but only two-frame correspondences. Thus it can be exploited to build tracks that are aware of segmentation, which constitute the foundation of a multibody structure from motion pipeline. Future research will explore this direction.

**Acknowledgements** The authors would like to thank Luca Magri for his guidance through the Matlab code of RPA and Petr Hruby for his help with COLMAP. This work was supported by the European Regional Development Fund under the project IMPACT (Registration No. CZ.02.1.01/0.0/0.0/15_003/0000468) and by the European Union's Horizon 2020 Research and Innovation Programme under the project SPRING (Grant Agreement No. 871245).

## A Finding the permutation between two sets of labels

Let $\mathbf{s} \in \{0, 1, 2, \ldots, d\}^m$ be a vector of length $m$ containing a set of labels ranging from 1 to $d$, where 0 defines a missing value. Let $\mathbf{t} \in \{0, 1, 2, \ldots, d\}^m$ be a vector representing the same labels but arranged in a different order (e.g., label 1 in vector $\mathbf{s}$ is called label 2 in vector $\mathbf{t}$). The task is to find a $d \times d$ permutation matrix $P$ that maps the vector $\mathbf{s}$ into the vector $\mathbf{t}$, namely

$$P = \text{bestMap}\,(\mathbf{s}, \mathbf{t}). \tag{12}$$

In order to cast Equation (12) to a known problem, we exploit a matrix representation of the labels. Let $S \in \{0, 1\}^{m \times d}$ be a binary matrix constructed as follows:

- $[S]_{hk} = 1$ if $\mathbf{s}[h] = k$;
- $[S]_{hk} = 0$ otherwise.

In other words, each row indexes one element in $\mathbf{s}$ and each column corresponds to one label: observe that zero labels (if any) translate into zero rows, hence they do not have any impact in subsequent computations; for the remaining rows, there is exactly one entry equal to one (which corresponds to the actual label), whereas all other entries are zero. In a similar way, we can define a binary matrix $T \in \{0, 1\}^{m \times d}$ associated to vector $\mathbf{t}$. The following equation shows an example:

$$\mathbf{s} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad T = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}. \tag{13}$$

Using this notation, we can rewrite Problem (12) as a matrix multiplication:

$$SP = T. \tag{14}$$

With reference to the example in Equation (13), the sought permutation is given by

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{15}$$

In the presence of noise, Equation (14) is usually solved in the least-squares sense:

$$\min_P ||SP - T||_F^2 \tag{16}$$

where the optimization variable is constrained to be a permutation matrix. By computation, it can be easily checked that Problem (16) is equivalent to:

$$\min_P ||P - S^\mathsf{T} T||_F^2. \tag{17}$$

Solving the above problem is tantamount to finding the closest permutation to the $d \times d$ matrix $S^\mathsf{T} T$. Such a task is also called *permutation Procrustes problem* (Gower and Dijksterhuis 2004) and it reduces to a linear assignment problem, which can be solved (e.g.) with the Hungarian algorithm (Kuhn 1955).

## B Permutation synchronization

The task of *synchronization* (Singer 2011) is to recover elements of a group starting from ratios between pairs of group elements. The problem can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V}$ corresponds to the unknowns and the edge set $\mathcal{E}$ corresponds to the measures, and it is well posed only if the graph is connected. This topic has been recently surveyed in Arrigoni and Fusiello (2020).

In the case of *permutation synchronization* (Pachauri et al. 2013) the underlying group is the set of permutation matrices, which is also known as the Symmetric Group:

$$Sym(d) = \{Q \in \{0, 1\}^{d \times d} \text{ s.t. } Q\mathbf{1} = \mathbf{1}, \mathbf{1}^\mathsf{T} Q = \mathbf{1}^\mathsf{T}\}. \tag{18}$$

Here $\mathbf{1}$ denotes a $d \times 1$ vector of ones. Observe that a permutation matrix has exactly one entry equal to 1 in each row and column, and all other entries are equal to zero. Let $P_i$ denote the *unknown* permutation associated with vertex $i \in \mathcal{V}$, and let $P_{ij}$ denote the *known* permutation associated with edge $(i, j) \in \mathcal{E}$ representing the ratio between $P_i$ and $P_j$, namely

$$P_{ij} = P_i P_j^\mathsf{T} \tag{19}$$

where the inverse is given by matrix transposition. The task of permutation synchronization is to compute $P_i$ for all $i \in \mathcal{V}$ such that the consistency constraint in Equation (19) is best satisfied, as shown in Figure 4.

An effective and simple solution can be derived via spectral decomposition (Pachauri et al. 2013), thanks to a matrix representation of the problem. For simplicity of exposition, we first review the spectral solution in the case where the

graph is complete. Let us collect all the unknowns and measures in two block-matrices $X$ and $Z$ constructed as follows

$$X = \begin{bmatrix} P_1 \\ P_2 \\ \dots \\ P_n \end{bmatrix}, \quad Z = \begin{bmatrix} I & P_{12} & \dots & P_{1n} \\ P_{21} & I & \dots & P_{2n} \\ \dots & & & \dots \\ P_{n1} & P_{n2} & \dots & I \end{bmatrix} \tag{20}$$

where $n$ denotes the number of vertices and $I$ denotes the $d \times d$ identity matrix. Using this notation, Eq. (19) becomes $Z = XX^{\mathsf{T}}$, which implies that $Z$ is symmetric positive semidefinite and it satisfies the following equality:

$$ZX = nX. \tag{21}$$

Equation (21) means that the columns of $X$ are $d$ eigenvectors of $Z$. Since $Z$ has rank $d$, all other eigenvalues are zero, so $n$ is the *largest* eigenvalue. This suggests the following procedure (Pachauri et al. 2013) to address permutation synchronization:

1. compute the $d$ leading eigenvectors of $Z$, which are collected in a matrix $U$;
2. project each $d \times d$ block in $U$ to a proper permutation matrix.

The second step is indeed required since the eigenvectors are an approximate solution that does not enforce the specific constraints of permutation matrices. Finding the closest permutation to a given matrix is a linear assignment problem (Kuhn 1955).

The spectral solution can be easily extended to the case where the graph is not complete. In this situation, missing measures are represented as zero blocks in $Z$ and Equation (21) generalizes to

$$ZX = (D \otimes I)X \tag{22}$$

where $D$ denotes the degree matrix of the graph and $\otimes$ denotes the Kronecker product. In other words, the columns of $X$ are $d$ eigenvectors of $(D \otimes I)^{-1}Z$, hence the spectral solution applies equally well to the case of missing edges. See (Shen et al. 2016; Arrigoni and Fusiello 2020) for more details.

The synchronization problem also makes sense when nodes/edges in the graph are associated with *partial* permutation matrices, which form the so-called Symmetric Inverse Semigroup:

$$ISym(d) = \{Q \in \{0,1\}^{d \times d} \text{ s.t. } Q\mathbf{1} \leq \mathbf{1}, \mathbf{1}^{\mathsf{T}}Q \leq \mathbf{1}^{\mathsf{T}}\}. \tag{23}$$

Observe that a partial permutation has at most one nonzero entry in each row and column, and these nonzero entries are all equal to 1.

It is shown in Maset et al. (2017) that the spectral method also works in the case of partial permutations. If the size of permutations (i.e., $d$) is not known in advance, then the authors of Maset et al. (2017) suggest to over-estimate it, and hence compute more eigenvectors than what is actually required. This choice is motivated by the fact that $Z$ has rank $d$ (in the absence of noise), hence it is expected that the eigenvectors in position $d+1, d+2, \dots$ will have nearly zero eigenvalue (in the presence of noise), so that their impact in the spectral decomposition is very small. In practice, the method proposed in Maset et al. (2017) – named MatchEIG – does not estimate the right value of $d$, but it can be viewed as a denoising procedure that refines the measurement matrix $Z$. In Tron et al. (2017) it is observed that the measurement matrix $Z$ can be interpreted as the adjacency matrix of a graph, hence permutation synchronization is cast to a "graph clustering" problem. Such method (called QuickMatch) automatically estimates the value of $d$ in a bottom-up way.

## References


Arrigoni, F., & Fusiello, A. (2020). Synchronization problems in computer vision with closed-form solutions. *International Journal of Computer Vision, 128*, 26–52.

Arrigoni, F., Magri, L., & Pajdla, T. (2020). Motion segmentation with pairwise matches and unknown number of motions. In: Proceedings of the international conference on pattern recognition.

Arrigoni, F., Magri, L., & Pajdla, T. (2020). On the usage of the trifocal tensor in motion segmentation. In: Proceedings of the European conference on computer vision.

Arrigoni, F., & Pajdla, T. (2019). Motion segmentation via synchronization. In: IEEE international conference on computer vision workshops (ICCVW).

Arrigoni, F., & Pajdla, T. (2019). Robust motion segmentation from pairwise matches. In: Proceedings of the international conference on computer vision.

Arrigoni, F., Rossi, B., & Fusiello, A. (2016). Spectral synchronization of multiple views in SE(3). *SIAM Journal on Imaging Sciences, 9*(4), 1963–1990.

Barath, D., & Matas, J. (2018). Multi-class model fitting by energy minimization and mode-seeking. In: Proceedings of the European conference on computer vision, pp. 229–245. Springer International Publishing.

Bernard, F., Thunberg, J., Gemmar, P., Hertel, F., Husch, A., & Goncalves, J. (2015). A solution for multi-alignment by transformation synchronisation. In: Proceedings of the IEEE conference on computer vision and pattern recognition.

Bernard, F., Thunberg, J., Swoboda, P., & Theobalt, C. (2019). HiPPI: Higher-order projected power iterations for scalable multi-matching. In: Proceedings of the international conference on computer vision.

Bian, J., Lin, W.Y., Matsushita, Y., Yeung, S.K., Nguyen, T.D., & Cheng, M.M. (2017). Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2828–2837

Bideau, P., & Learned-Miller, E. (2016). It's moving! a probabilistic model for causal motion segmentation in moving camera videos.

In: Proceedings of the European conference on computer vision, pp. 433–449. Springer International Publishing.
Bideau, P., Menon, R.R., & Learned-Miller, E.G. (2018). Moa-net: Self-supervised motion segmentation. In: ECCV Workshops, *Lecture Notes in Computer Science*, vol. 11134, pp. 715–730. Springer.
Bideau, P., RoyChowdhury, A., Menon, R.R., & Learned-Miller, E. (2018). The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 508–517.
Birdal, T., & Simsekli, U. (2019). Probabilistic permutation synchronization using the riemannian structure of the birkhoff polytope. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 11105–11116.
Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(11), 1222–1239.
Caelles, S., Pont-Tuset, J., Perazzi, F., Montes, A., Maninis, K.K., & Van Gool, L. (2019). The 2019 davis challenge on vos: Unsupervised multi-object segmentation. arXiv:1905.00737
Carlone, L., Tron, R., Daniilidis, K., & Dellaert, F. (2015). Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In: Proceedings of the IEEE international conference on robotics and automation.
Chen, Y., Li, C.G., & You, C. (2020). Stochastic sparse subspace clustering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR).
Chin, T.J., Suter, D., & Wang, H. (2010). Multi-structure model selection via kernel optimisation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3586–3593.
Cortes, C., Mohri, M., & Rostamizadeh, A. (2009). Learning non-linear combinations of kernels. In: Advances in neural information processing systems, pp. 396–404.
Dave, A., Tokmakov, P., & Ramanan, D. (2019). Towards segmenting anything that moves. In: ICCV Workshops.
Delong, A., Gorelick, L., Veksler, O., & Boykov, Y. (2012). Minimizing energies with hierarchical costs. *International Journal of Computer Vision, 100*(1), 38–58.
Delong, A., Osokin, A., Isack, H. N., & Boykov, Y. (2012). Fast approximate energy minimization with label costs. *International Journal of Computer Vision, 96*(1), 1–27.
Dragon, R., Ostermann, J., & Van Gool, L. (2013). Robust real-time motion-split-and-merge for motion segmentation. In: German conference on pattern recognition, pp. 425–434. Springer Berlin Heidelberg.
Elhamifar, E., & Vidal, R. (2013). Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(11), 2765–2781.
Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition.
Govindu, V.M. (2004). Lie-algebraic averaging for globally consistent motion estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 684–691.
Gower, J.C., & Dijksterhuis, G.B. (2004). Procrustes Problems. Oxford University Press.
Hartley, R., Aftab, K., & Trumpf, J. (2011). L1 rotation averaging using the Weiszfeld algorithm. Proceedings of the IEEE conference on computer vision and pattern recognition pp. 3041–3048.
Hartley, R. I., Trumpf, J., Dai, Y., & Li, H. (2013). Rotation averaging. *International Journal of Computer Vision, 103*(3), 267–305.
Hartley, R. I., & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision* (2nd ed.). Cambridge University Press.
Isack, H., & Boykov, Y. (2012). Energy-based geometric multi-model fitting. *International Journal of Computer Vision, 97*(2), 123–147.
Ji, P., Li, H., Salzmann, M., & Dai, Y. (2014). Robust motion segmentation with unknown correspondences. In: Proceedings of the European Conference on Computer Vision, pp. 204–219. Springer International Publishing.
Ji, P., Salzmann, M., & Li, H. (2015). Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In: Proceedings of the international conference on computer vision, pp. 4687–4695.
Jung, H., Ju, J., & Kim, J. (2014). Rigid motion segmentation using randomized voting. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1210–1217.
Keuper, M. (2017). Higher-order minimum cost lifted multicuts for motion segmentation. In: Proceedings of the international conference on computer vision, pp. 4252–4260.
Keuper, M., Andres, B., & Brox, T. (2015). Motion trajectory segmentation via minimum cost multicuts. In: Proceedings of the international conference on computer vision, pp. 3271–3279.
Keuper, M., Tang, S., Andres, B., Brox, T., & Schiele, B. (2020). Motion segmentation multiple object tracking by correlation co-clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(1), 140–153.
Kohli, P., Kumar, M.P., & Torr, P.H.S. (2007). $p^3$ & beyond: Solving energies with higher order cliques. In: 2007 IEEE conference on computer vision and pattern recognition, pp. 1–8.
Kuang, D., Yun, S., & Park, H. (2014). SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering. *Journal of Global Optimization, 62*(3), 545–574.
Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2*(2), 83–97.
Kumar, A., Rai, P., & Daume, H. (2011). Co-regularized multi-view spectral clustering. In: Advances in neural information processing systems, pp. 1413–1421.
Lai, T., Wang, H., Yan, Y., Chin, T. J., & Zhao, W. L. (2017). Motion segmentation via a sparsity constraint. *IEEE Transactions on Intelligent Transportation Systems, 18*(4), 973–983.
Lamdouar, H., Yang, C., Xie, W., & Zisserman, A. (2020). Betrayed by motion: Camouflaged object discovery via motion segmentation. In: Asian conference on computer vision.
Li, C.G., & Vidal, R. (2015). Structured sparse subspace clustering: A unified optimization framework. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 277–286.
Li, Z., Guo, J., Cheong, L.F., & Zhou, S.Z. (2013). Perspective motion segmentation via collaborative clustering. In: Proceedings of the international conference on computer vision, pp. 1369–1376.
Lin, K., Jiang, N., Cheong, L.F., Lu, J., & Xu, X. (2018). Robust video background identification by dominant rigid motion estimation. In: Proceedings of the Asian conference on computer vision, pp. 163–178. Springer International Publishing.
Lin, Z., Chen, M., & Ma, Y. (2010). The augmented lagrange multiplier method for exact recovery of corrupted Low-Rank matrices. eprint arXiv:1009.5055
Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., & Ma, Y. (2013). Robust recovery of subspace structures by low-rank representation. IEEE Transactions on pattern analysis and machine intelligence pp. 171–184.
Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94
Magri, L., & Fusiello, A. (2014). T-Linkage: A continuous relaxation of J-Linkage for multi-model fitting. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3954–3961.
Magri, L., & Fusiello, A. (2015). Robust multiple model fitting with preference analysis and low-rank approximation. In: Proceedings

of the British machine vision conference, pp. 20.1–20.12. BMVA Press.
Magri, L., & Fusiello, A. (2016). Multiple models fitting as a set coverage problem. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3318–3326.
Maset, E., Arrigoni, F., & Fusiello, A. (2017). Practical and efficient multi-view matching. In: Proceedings of IEEE international conference on computer vision, pp. 4568–4576.
Mattheus, J., Grobler, H., & Abu-Mahfouz, A.M. (2020). A review of motion segmentation: Approaches and major challenges. In: 2020 2nd International multidisciplinary information technology and engineering conference (IMITEC), pp. 1–8.
Ochs, P., Malik, J., & Brox, T. (2014). Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence, 36*(6), 1187–1200.
Olsson, C., & Enqvist, O. (2011). Stable structure from motion for unordered image collections. In: Proceedings of the 17th scandinavian conference on image analysis (SCIA'11), pp. 524–535. Springer-Verlag.
Pachauri, D., Kondor, R., & Singh, V. (2013). Solving the multi-way matching problem by permutation synchronization. In: Advances in neural information processing systems, vol. 26, pp. 1860–1868.
Perazzi, F., Pont-Tuset, J., McWilliams, B., Gool, L.V., Gross, M., & Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition.
Pham, T. T., Chin, T. J., Yu, J., & Suter, D. (2014). The random cluster model for robust geometric fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36*(8), 1658–1671.
Poling, B., & Lerman, G. (2014). A new approach to two-view motion segmentation using global dimension minimization. *International Journal of Computer Vision, 108*, 165–185.
Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., & Van Gool, L. (2017). The 2017 davis challenge on video object segmentation. arXiv:1704.00675
Rao, S., Tron, R., Vidal, R., & Ma, Y. (2010). Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *Pattern Analysis and Machine Intelligence, 32*(10), 1832–1845.
Rosen, D., Carlone, L., Bandeira, A., & Leonard, J. (2017). SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. Tech. Rep. MIT-CSAIL-TR-2017-002, Computer science and artificial intelligence laboratory, Massachusetts Institute of Technology.
Rubino, C., Del Bue, A., & Chin, T.J. (2018). Practical motion segmentation for urban street view scenes. In: Proceedings of the IEEE International conference on robotics and automation.
Sabzevari, R., & Scaramuzza, D. (2016). Multi-body motion estimation from monocular vehicle-mounted cameras. *IEEE Transactions on Robotics, 32*(3), 638–651.
Santellani, E., Maset, E., & Fusiello, A. (2018). Seamless image mosaicking via synchronization. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, IV–2*, 247–254.
Saputra, M. R. U., Markham, A., & Trigoni, N. (2018). Visual SLAM and structure from motion in dynamic environments: A survey. *ACM Computing Surveys, 51*(2), 37:1-37:36.
Schindler, K., & Suter, D. (2005). Two-view multibody structure-and-motion with outliers. 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) **2**, 643–648 vol. 2
Schonberger, J.L., & Frahm, J.M. (2016). Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4104 – 4113.
Schroeder, P., Bartoli, A., Georgel, P., & Navab, N. (2011). Closed-form solutions to multiple-view homography estimation. In: Applications of computer vision (WACV), 2011 IEEE Workshop on, pp. 650–657.
Shen, Y., Huang, Q., Srebro, N., & Sanghavi, S. (2016). Normalized spectral map synchronization. In: Advances in neural information processing systems, vol. 29, pp. 4925–4933. Curran Associates, Inc.
Shi, X., Fan, W., & Yu, P.S. (2010). Efficient semi-supervised spectral co-clustering with constraints. In: 2010 IEEE International conference on data mining, pp. 1043–1048.
Singer, A. (2011). Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis, 30*(1), 20–36.
Tokmakov, P., Schmid, C., & Alahari, K. (2019). Learning to segment moving objects. *International Journal of Computer Vision, 127*, 282–301.
Toldo, R., & Fusiello, A. (2008). Robust multiple structures estimation with J-Linkage. In: Proceedings of the European conference on computer vision, pp. 537–547.
Tombari, F., & Di Stefano, L. (2011). 3d data segmentation by local classification and markov random fields. In: 2011 International conference on 3D imaging, modeling, processing, visualization and transmission, pp. 212–219.
Torsello, A., Rodolà, E., & Albarelli, A. (2011). Multiview registration via graph diffusion of dual quaternions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2441 – 2448.
Tron, R., & Vidal, R. (2007). A benchmark for the comparison of 3-d motion segmentation algorithms. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–8. IEEE.
Tron, R., Zhou, X., Esteves, C., & Daniilidis, K. (2017). Fast multi-image matching via density-based clustering. In: Proceedings of the international conference on computer vision, pp. 4077–4086.
Vidal, R., Ma, Y., & Sastry, S. (2005). Generalized principal component analysis (gpca). *IEEE Trans. Pattern Anal. Mach. Intell., 27*(12), 1945–1959.
Vidal, R., Ma, Y., Soatto, S., & Sastry, S. (2006). Two-view multibody structure from motion. *International Journal of Computer Vision, 68*(1), 7–25.
Vidal, R., Tron, R., & Hartley, R. (2008). Multiframe motion segmentation with missing data using power factorization and GPCA. *International Journal of Computer Vision, 79*(1), 85–105.
Vincent, E., & Laganiere, R. (2001). Detecting planar homographies in an image pair. In: International symposium on image and signal processing and analysis, pp. 182–187.
Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing, 17*(4), 395–416.
Wang, X., Qian, B., & Davidson, I. (2014). On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery, 28*(1), 1–30.
Wang, Y., Liu, Y., Blasch, E., & Ling, H. (2018). Simultaneous trajectory association and clustering for motion segmentation. *IEEE Signal Processing Letters, 25*(1), 145–149.
Xu, L., Oja, E., & Kultanen, P. (1990). A new curve detection method: randomized Hough transform (RHT). *Pattern Recognition Letters, 11*(5), 331–338.
Xu, X., Cheong, L.F., & Li, Z. (2018). Motion segmentation by exploiting complementary geometric models. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2859–2867.
Yan, J., & Pollefeys, M. (2006). A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In: Proceedings of the European conference on computer vision, pp. 94–106.

Yao, R., Lin, G., Xia, S., Zhao, J., & Zhou, Y. (2020). Video object segmentation and tracking: A survey. *ACM Transactions on Intelligent Systems and Technology, 11*(4), 1–47.

Yuan, S., Tan, P.N., Cheruvelil, K.S., Collins, S.M., & Soranno, P.A. (2019). Spatially constrained spectral clustering algorithms for region delineation.

Zhang, W., & Kosecká, J. (2006). Nonparametric estimation of multiple structures with outliers. In: Workshop on dynamic vision, European Conference on computer vision 2006, *Lecture Notes in Computer Science*, vol. 4358, pp. 60–74. Springer.

Zhao, Q., Zhang, Y., Qin, Q., & Luo, B. (2020). Quantized residual preference based linkage clustering for model selection and inlier segmentation in geometric multi-model fitting. Sensors **20**(13).

Zuliani, M., Kenney, C.S., & Manjunath, B.S. (2005). The multi-RANSAC algorithm and its application to detect planar homographies. In: Proceedings of the IEEE international conference on image processing, pp. III–153–6.