Motion segmentation with pairwise matches and unknown number of motions

Federica Arrigoni
DISI – University of Trento
federica.arrigoni@unitn.it

Luca Magri
DEIB - Politecnico di Milano
luca.magri@polimi.it

Tomas Pajdla
CIIRC - CTU in Prague
pajdla@cvut.cz

***Abstract*—In this paper we address motion segmentation, that is the problem of clustering points in multiple images according to a number of moving objects. Two-frame correspondences are assumed as input without prior knowledge about trajectories. Our method is based on principles from "multi-model fitting" and "permutation synchronization", and – differently from previous techniques working under the same assumptions – it can handle an *unknown* number of motions. The proposed approach is validated on standard datasets, showing that it can correctly estimate the number of motions while maintaining comparable or better accuracy than the state of the art.**

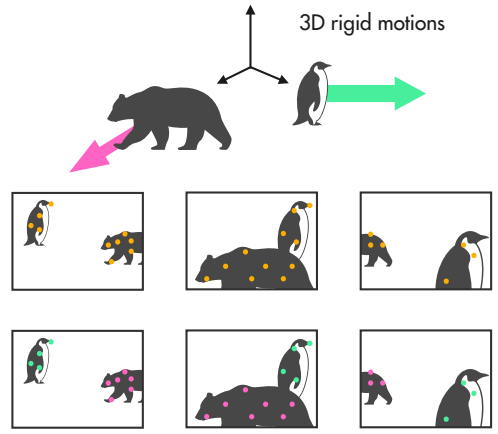
I. Introduction

Given a 3D dynamic scene with moving objects and a collection of images, *motion segmentation* refers to the problem of segmenting the image points according to the moving objects they belong to (see Fig. 1a). This problem has attracted a lot of attention in the literature, thanks to its application in a variety of tasks in Computer Vision and Robotics [1], [2], [3]. In this work we present Mode-U, a novel method that solves the motion segmentation problem from pairwise matches for an unknown number of *rigid* motions, as illustrated in Fig. 1b.

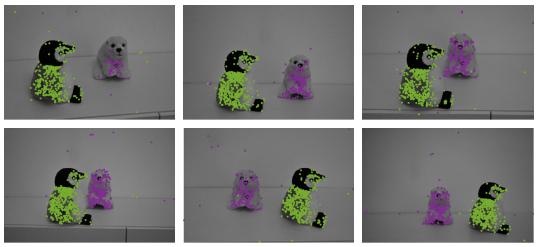
A. Related Work

Previous works on motion segmentation can be grouped into three main categories, namely trajectory clustering, segmentation with two-frame correspondences and with unknown correspondences. As represented in Fig. 2, this taxonomy reflects the different levels of prior assumptions made on input data and hence the different degrees of practicality of the various methods. Stronger assumptions, which however are rarely available in practice, make the problem easier to deal with at the expense of actual usability in real applications.

1) Trajectory Clustering: it is assumed that a set of points is tracked through multiple images, and the task is to cluster those trajectories (i.e., *multi-frame* correspondences) into different motions. This category – which is represented in the right part of Fig. 2 – accounts for the majority of works in the literature (e.g., [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]). The typical scenario consists in videos with small motions between consecutive frames, which appear in surveillance, scene understanding and autonomous driving. The main limitation of these approaches is that trajectories are seldom available in practice. For example, in the popular Hopkins dataset [15], which has been extensively used in the literature, the input trajectories are not fully realistic since they were filtered with manual operations. The same happens for the more recent KT3DMoSeg benchmark [14] built on KITTI [16].



(a) Motion segmentation problem



(b) Sample results of Mode-U

Fig. 1: (Top) A cartoon illustration of motion segmentation: given a 3D scene with independently moving objects and a collection of images with key-points, the aim is to identify the objects in the images. (Bottom) The segmentation results obtained by Mode-U on the *Penguin* sequence [4]. Our method correctly clusters most of the points in two moving objects represented by a penguin (green) and a bear (violet), without knowing in advance the number of motions.

2) Segmentation with Two-frame Correspondences: the task is to cluster image points (e.g., SIFT keypoints [17]) into different motions, assuming that matches between pairs of images (i.e., *two-frame* correspondences) are available only. This problem – represented in the middle of Fig. 2 – is addressed in [4] and [18] only. Despite poorly studied, it has a great practical relevance since it does not assume the knowledge of multi-frame correspondences, which are hard

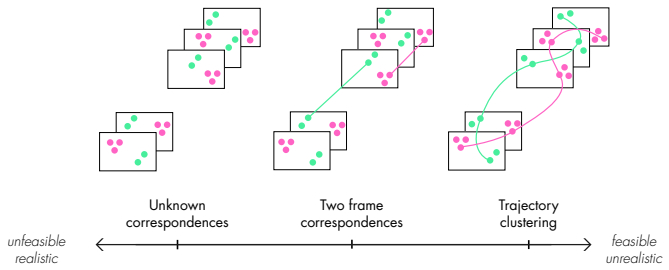


Fig. 2: Existing works can be divided into three categories: (i) trajectory clustering, (ii) segmentation with two-frame correspondences and (iii) segmentation with unknown correspondences. When going from right to left, the problem becomes more challenging since assumptions are weaker (but more realistic). This paper belongs to the middle category.

to compute in the presence of moving objects. The typical scenario involves unstructured/unordered image sets with large motions between different frames (e.g., the indoor scenes used in [4], [18]), which appear in the context of multi-body structure from motion [19].

3) Segmentation with Unknown Correspondences: it is assumed 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 classifying those trajectories into different groups. This problem – represented in the left part of Fig. 2 – is analysed in [20] and [21] only. Observe that addressing motion segmentation under such a weak assumption is very difficult due to the large number of unknowns, and existing approaches are not practical: in [20], [21] the number of expected trajectories is set to (at most) 200 due to algorithmic complexity.

B. Contribution and Outline

In this paper we tackle motion segmentation with two-frame correspondences. We are interested in this problem since it lies at the middle between trajectory clustering and the case of unknown correspondences, hence it represents a good trade-off between making *realistic assumptions* and addressing a *feasible task*. Our work is based on [4] where a two-step strategy is proposed: first, motion segmentation is solved on different image pairs independently; secondly, such partial results are properly combined to get a multi-frame segmentation. The main drawback of such technique is that it assumes that the number of motion is known in advance. However, this information is hardly available in practical scenarios. In this paper we extend [4] to the case of *unknown* number of motions, thus providing a novel solution to motion segmentation with two-frame correspondences, which is able to work under more realistic assumptions. Experiments show that the proposed approach successfully estimates the correct number of motions on standard datasets, being comparable to [4] in terms of accuracy.

Retrieving the number of motions was already done in previous works (e.g., [12], [14]). However, note that such methods consider a different task than ours, for they address trajectory clustering. In other words, our method is the first one that is able to handle an unknown number of objects within the category of methods performing motion segmentation with two-frame correspondences.

The paper is organized as follows. Sec. II defines some background notions necessary to understand the paper. Sec. III is devoted to present our method and differences with respect to [4] are also illustrated. Experiments are reported in Sec. IV and the conclusion is drawn in Sec. V.

II. Background

In this section we briefly review two topics which are essential to define our method, namely "multi-model fitting" and "permutation synchronization".

A. Multi-model Fitting

Multi-model fitting aims at extracting multiple parametric models from unstructured data to segment it in higher-level geometric descriptions. Motion segmentation in two images can be seen as a particular instance of this problem, where the fitted parametric models are fundamental matrices and the segmentation corresponds to rigid bodies. In simple terms, if a 3D point undergoes a rigid motion, its projections in two images are related by a second order relation described by a fundamental matrix [22], with different motions giving rise to different fundamental matrices. Therefore, given two images of the same dynamic scene and a set of noisy matches (usually corrupted by outliers), a segmentation of the matches in rigid bodies can be obtained by robustly fitting fundamental matrices and by grouping together the matches that are described by the same two-view relation.

Several challenges make this problem difficult. First, the estimation procedure must tolerate both rogue mismatches and pseudo-outliers. Second, the fit of models and the segmentation of points are closely intertwined steps that give rise to a sort of chicken-and-egg-dilemma (the segmentation must be derived from models, but at the same time models must be fitted to segments). Moreover, the problem of multi model fitting is inherently ill-posed: many different interpretations of the same data are possible. For this reason, several regularization strategies that constrain the solution using prior information – usually in the form of one or more parameters – are employed.

Several approaches have been proposed to solve this problem, that range from energy-based optimization (e.g., [23], [24], [25]) to clustering formulations (e.g., [26], [27], [28]). T-linkage [27] is a multi-model fitting algorithm that belongs to the latter category which is closely related to our solution to motion segmentation, as it does not require as input the number of models to retrieve. T-linkage follows a two-step *first-represent-then-clusterize* paradigm. The main idea is to describe data points in a conceptual space as vectors of "preferences", which measure how well they are fitted by a pool of provisional models instantiated via random sampling. The rationale is that – in this conceptual space – the inliers of a parametric model can be recognized as clusters of neighbouring points sharing similar preferences. In practice,

model hypotheses are instantiated from a minimal sample set (i.e., the minimum number of points necessary to fit a model), residuals are computed for every model, and the preference a point grants to a model is expressed in term of its residual using a robust voting function. An inlier threshold ϵ is used to cutoff preference votes when residuals are bigger than ϵ.

The segmentation step in T-Linkage is performed exploiting a bottom-up hierarchical clustering scheme. At first, every model is put in its own cluster, then the two clusters having more similar preferences (measured with the Tanimoto distance) are merged together. The merging phase continues until there exists a sampled model that can explain all the points belonging to two distinct clusters. In this way the number of clusters is automatically detected. The inlier threshold works as a regularization parameter: by defining the size of the inlier set of models, it implicitly controls the number of attained models. This approach is robust to outliers, that turns to be isolated points in the conceptual space. Typically, outliers emerge as micro-clusters that can be easily pruned out with outlier rejection criteria, the simplest being based (for example) on cluster cardinality.

B. Permutation Synchronization

Matches between objects (e.g., key-points) in two structures (e.g., images or shapes) are represented by a permutation matrix, which can be either total (square) or partial (rectangular). A *total* permutation has exactly one entry in each row and column equal to 1 and all other entries are zero. A *partial* permutation has at most one nonzero entry in each row and column, and these nonzero entries are all equal to 1. The former requires all the objects to be visible in both structures, whereas the latter models missing correspondences. Synchronization of permutation matrices is equivalent to joining matches in multi-view correspondences (e.g., across multiple images or shapes) while enforcing loop-closure constraints. This is a particular case of the more general "synchronization" problem [29].

Permutation synchronization is hard to solve due to the combinatorial nature of the problem, hence existing approaches usually rely on approximate solutions (see [30], [31], [32], [33], [34], [35], [36], [37]). Moreover, the presence of noise (i.e., false matches) increases the difficulty of the problem. Another relevant issue concerns the number of objects. In the case of total permutations, it is clear that the size of the permutations equals the total amount of objects. In the (more practical) case of partial permutations, instead, it is not obvious how many objects are present due to missing correspondences. To overcome such difficulty, some approaches (e.g., [30]) address a simplified problem by assuming that the total amount of objects across all the structures is known. Among existing methods, we are particularly interested in MatchEIG [34] and QuickMatch [35], which are illustrated here and will be used in the next section.

MatchEIG: this method works with partial permutations and it requires an over estimate of the total amount of objects (denoted by f), which in practice is set equal to twice the average number of objects present in each structure. The authors of [34] introduce a big block-matrix, named the "measurement matrix", which collects all the permutation matrices representing matches between pairs of structures. First, the measurement matrix is projected onto its rank-f approximation via spectral decomposition; then, such approximation is transformed into a binary matrix using a threshold $0 < t < 1$; finally, each block – representing matches between two structures – of the binary matrix is projected onto a partial permutation using a greedy strategy. MatchEIG can be interpreted as a denoising procedure, which reduces the number of false matches and completes missing matches with new ones retrieved indirectly via loop closure. However, multi-view correspondences are not computed explicitly.

QuickMatch: this method works with partial permutations and it computes multi-view correspondences while automatically estimating the total number of objects. The authors of [35] interpret the measurement matrix as the adjacency matrix of a graph, hence permutation synchronization is cast to a "graph clustering" problem. Such a problem is solved with QuickShift [38], which performs clustering by seeking the modes of an empirical density estimate. Exclusion constraints are also considered in the formulation, in order to ensure that a single cluster cannot contain two objects from the same structure. The number of clusters is automatically detected since the problem is solved in a bottom-up way, similarly to T-Linkage. At the end micro-clusters (i.e., clusters with cardinality lower than a parameter τ) can be possibly removed in order to improve robustness to outliers.

III. Proposed Method

In this section we derive our solution to motion segmentation, which can be seen as an extension of MODE [4] to the case where the correct number of motions is not available. This covers more scenarios of practical interest. The proposed approach is named MODE-U where "U" stands for *unknown* number of motions.

A. Problem Formulation

The task is to segment $p = \sum_{i=1}^{n} p_i$ points in n images according to d motions, where p_i denotes the number of points in image i. It is assumed that correspondences between points in image pairs are available. Such a task can be expressed as the problem of estimating the "total segmentation" of each image starting from a redundant set of "partial segmentations", which are defined as follows.

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 image i: labels from 1 to d identify the membership to a specific motion, while the zero label identifies the *unclassified* points, that are those points whose cluster can not be established due to the presence of mismatches.

The *partial segmentation* of pair $\alpha = (i, j)$ is denoted by

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

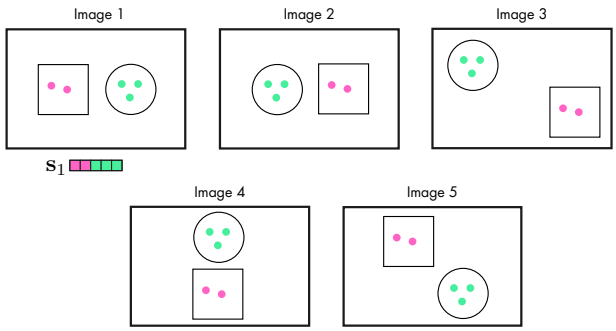


Fig. 3: Images and a sample of a *total segmentation*. The task is to assign a label (green or pink) to each point in several images based on the moving object (square or circle) it belongs to.

and it represents the labels of corresponding points in images i and j: labels from 1 to d identify the membership to a specific motion, while the zero label identifies those correspondences which are labelled as outlier. Here $m_\alpha \leqslant \min\{p_i, p_j\}$ denotes the number of matches in the pair α. Note that $\mathbf{t}_\alpha \in \{0, 1, \ldots, d\}^{m_\alpha}$ 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 Eq. (3) refers to an image pair whereas subscripts refer to images in the pair. In this paper Greek letters are used to denote pairs of images.

Observe that total segmentations are global/absolute whereas partial segmentations are relative/local. The former represent our desired output, shown in Fig. 3, whereas the latter can be computed as explained in the next paragraph and they are represented in Fig. 4.

B. Two-frame Segmentation

Motion segmentation in two images can be seen as an instance of multi-model fitting, where we aim at fitting multiple fundamental matrices to corresponding points (see Sec. II-A). To accomplish such a task, we exploit T-linkage [27], that handles an unknown number of motions. The input data consist in matches between two images i and j, and the output is a partial segmentation $\mathbf{t}_\alpha$ with $\alpha = (i, j)$. Given an estimate of the inlier threshold ϵ – that was fixed to $\epsilon = 0.1$ in all our experiments – T-linkage segments all the matches by fitting fundamental matrices, including outliers. A very simple outlier rejection strategy is employed to get rid of the majority of mismatches: clusters having less than 8 points (which is the minimal number of matches required to instantiate a fundamental matrix) are rejected.

This procedure is applied to all the image pairs in order to obtain the partial segmentations. Two challenges have to be addressed:

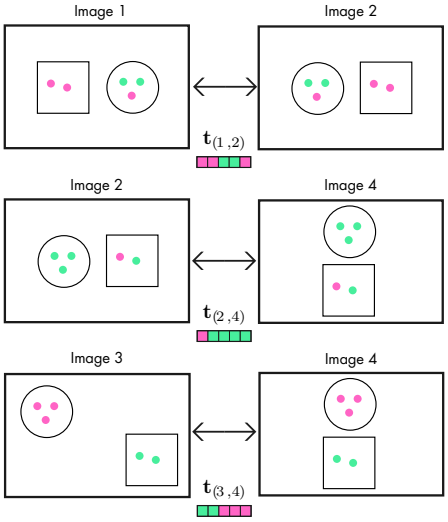


Fig. 4: Sample image pairs and their *partial segmentations*. Note that results may be noisy and they are not absolute: the square is given the pink label in the top pair but it is given the green label in the bottom pair.

- *Ambiguity*: each partial segmentation considers its own labelling of the motions, i.e., the same motion may have a different label in different pairs;
- *Robustness*: each partial segmentation may contain errors, which are caused either by mismatches or by incorrect results attained by T-Linkage coupled with the outlier rejection based on cardinality.

In the next paragraphs we will explain how to address these challenges one at a time.

C. Ambiguity

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 Fig. 5. The interpretation is that – after applying the permutation P_α 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. Each edge in the graph corresponds to a *known* permutation derived as follows. Let k be a common image between pairs α and β (i.e., $k \in \alpha \cap \beta$) and let $P_{\alpha\beta}$ denote the permutation matrix associated with the edge (α, β), 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 α) into the vector $\mathbf{s}_k^\beta$ (i.e., labels of image k in pair β). Recall that $\mathbf{s}_k^\alpha$ and $\mathbf{s}_k^\beta$ are recovered from $\mathbf{t}_\alpha$ and $\mathbf{t}_\beta$ respectively via Eq. (3). Finding

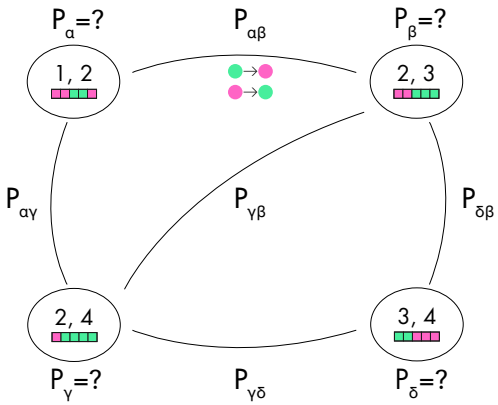


Fig. 5: The graph formulation of the permutation synchronization problem. The vertices represent unknown permutations associated with image pairs. The edges represent known permutations between partial segmentations.

$P_{\alpha\beta}$ is a *linear assignment* problem, which can be solved with the Hungarian algorithm [39].

To sum up, we have to address the problem of recovering an unknown permutation P_α 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. With reference to the terminology used in Sec. II-B, "objects" correspond to motions and "structures" correspond to image pairs. In other words, the task is to connect motions across multiple image pairs. Observe that the involved permutation matrices may be partial/rectangular: it can happen that two partial segmentations have a different number of objects if T-Linkage estimates a wrong number of motions. This usually happens in the case where some spurious models with small support are retained as valid motions by the simple outlier rejection implemented to refine the segmentation.

In order to solve permutation synchronization, we proceed as follows:

1) a measurement matrix is constructed from the permutations $\{P_{\alpha\beta}\}_{(\alpha,\beta)\in\mathcal{E}}$ which is refined with MatchEIG [34]; the parameter t (which is used to threshold the entries in order to get binary values) is set to $t = 0.3$ in our experiments[1];
2) QuickMatch [35] is applied to the *refined* measurement matrix constructed in the previous step; the parameter τ (which defines the cardinality of micro-clusters to be removed) is set to $\tau = 5$ in our experiments[1].

Step 1 has the effect of removing potential errors in the input permutations (caused, e.g., by a wrong estimation of the number of motions in some partial segmentations). Step 2 has the effect of fixing the permutation ambiguity while estimating the number of motions. Recall that QuickMatch works under the assumption of an unknown number of objects (see Sec. II-B), so it is suited to our problem. At the end, the permutation P_α is applied to the partial segmentation $\mathbf{t}_\alpha$ for each pair α. This has the consequence of (possibly) reshuffling the labels of motions in individual image pairs, so that the same motion has the same label in different pairs.

[1]This value provided good results in all our experiments. This choice is not critical and other values gave similar performance.

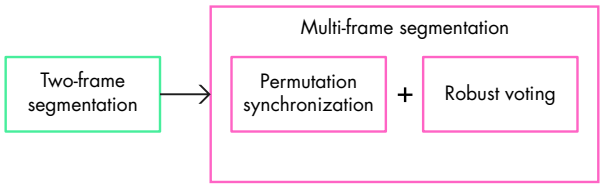


Fig. 6: Our approach and MODE [4] address motion segmentation with two-frame correspondences based on the following general approach: first, segmentation is addressed on pairs of images via multi-model fitting; then, segmentation is solved on all the images by permutation synchronization and robust voting.

D. Robustness

We now explain how to address the "robustness" challenge. Observe that Eq. (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. In order to assign a label to each point, the following voting criterion [4] 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 labelled as unknown) 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.

E. Final Remarks

We observe that both our method (named MODE-U) and MODE [4] can be seen as instances of the same general framework, illustrated in Fig. 6. First, motion segmentation is solved on different image pairs independently; then, such local results are combined in a multi-frame segmentation by permutation synchronization (which fixes the permutation ambiguity) followed by a robust voting scheme (which handles noise). MODE and MODE-U differ for the techniques used to address two sub-tasks, namely two-frame segmentation and permutation synchronization, as clarified in Tab. II. MODE uses Robust Preference Analysis (RPA) [28] for the first task and Spectral [30] for the second task, whereas our approach uses T-Linkage

TABLE I: Misclassification error [%] (the lower the better) and classified points [%] (the higher the better) for several methods on the data used in [4], [18]. The number of motions (denoted by d), the number of images (denoted by n) and the total number of image points (denoted by p) are also reported for each scene. The left column under MODE-U reports the number of motions estimated by our method (denoted by $\widehat{d}$), while the right column reports the percentage of image pairs where T-Linkage recovers the correct number of motions. The best results are highlighted in boldface.

					MODE-U			MODE [4]		SYNCH [18]	
Dataset	d	n	p	$\widehat{d}$	Error	Classified	Motions	Error	Classified	Error	Classified
Pen [18]	2	6	4550	2	1.55	89.08	100.00	**0.58**	80.07	0.82	83.23
Pouch [18]	2	6	4971	2	**1.39**	60.79	73.33	3.79	65.34	4.15	69.89
Needlecraft [18]	2	6	6617	2	1.80	67.07	86.67	**0.83**	72.81	1.04	76.76
Biscuits [18]	2	6	13158	2	1.12	90.42	93.33	**0.47**	84.47	0.51	87.28
Cups [18]	2	10	14664	2	2.05	71.31	95.56	**0.56**	65.42	1.01	69.82
Tea [18]	2	10	32612	2	0.69	85.21	88.89	**0.29**	81.70	28.12	52.21
Food [18]	2	10	36723	2	0.78	82.34	88.89	**0.36**	76.19	0.56	80.66
Penguin [4]	2	6	5865	2	1.36	66.60	91.67	**0.76**	69.17	44.21	46.97
Flowers [4]	2	6	7743	2	1.51	75.50	86.67	**1.23**	73.65	1.62	77.28
Pencils [4]	2	6	2982	2	**3.09**	51.01	73.33	3.80	65.33	27.53	40.44
Bag [4]	2	7	6114	2	2.78	52.91	80.95	**1.52**	57.95	25.92	54.27
Bears [4]	3	10	15888	3	**3.48**	68.21	31.11	4.82	73.65	38.95	74.59

[27] for two-frame segmentation and MatchEIG [34] combined with QuickMatch [35] for permutation synchronization. These different choices reflect different assumptions made by MODE and MODE-U: the former assumes a *known* number of motions, whereas the latter assumes an *unknown* number of motions, which is automatically estimated. Indeed, observe that RPA [28] performs multi-model fitting based on divisive partitional clustering and it requires as input the number of clusters. Spectral [30] works with total permutations, hence it assumes that the total amount of objects is known. T-linkage, MatchEIG and QuickMatch, instead, do not make such restrictive assumptions, as explained in Sec. II. As a consequence, our method is more general and practical, being able to handle scenarios where it is not possible to know in advance how many objects are moving in the scene.

TABLE II: Our approach and MODE [4] use different solutions for two-frame segmentation and permutation synchronization, that are key components within the segmentation pipeline in Fig. 6.

	Two-frame segmentation	*Permutation synchronization*
MODE [4]	RPA [28]	Spectral [30]
MODE-U	T-Linkage [27]	MatchEIG [34] + QuickMatch [35]

IV. EXPERIMENTS

In this section we assess the behaviour of MODE-U. The Matlab implementations of T-Linkage[2], MatchEIG[3] and QuickMatch[4], which are used within our motion segmentation pipeline, are publicly released. We compared our approach with MODE [4], which is reviewed in Sec. III-E, and we also included SYNCH [18] in the evaluation. The latter starts from multiple two-frame segmentations computed with RPA – as done by MODE– and it applies spectral clustering [40] to get the final segmentation, by leveraging on a matrix representation of motion segmentation. The Matlab implementations of MODE and SYNCH are available online[5]. Observe that all the analysed techniques belong to the same category (represented in the middle of Fig. 2), namely they address motion segmentation with two-frame correspondences. See Sec. I-A for more details about how to divide motion segmentation literature into three categories. Observe also that both MODE and SYNCH assume that the number of motions d is known a priori, whereas MODE-U automatically estimates such value from the data.

We considered the benchmark proposed in [4] and [18], which provides image points with ground-truth labels and noisy two-frame correspondences (obtained with SIFT [17]). The dataset comprises 12 indoor scenes with two or three motions counting from 6 to 10 images. As done in [4] and [18], we computed 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.

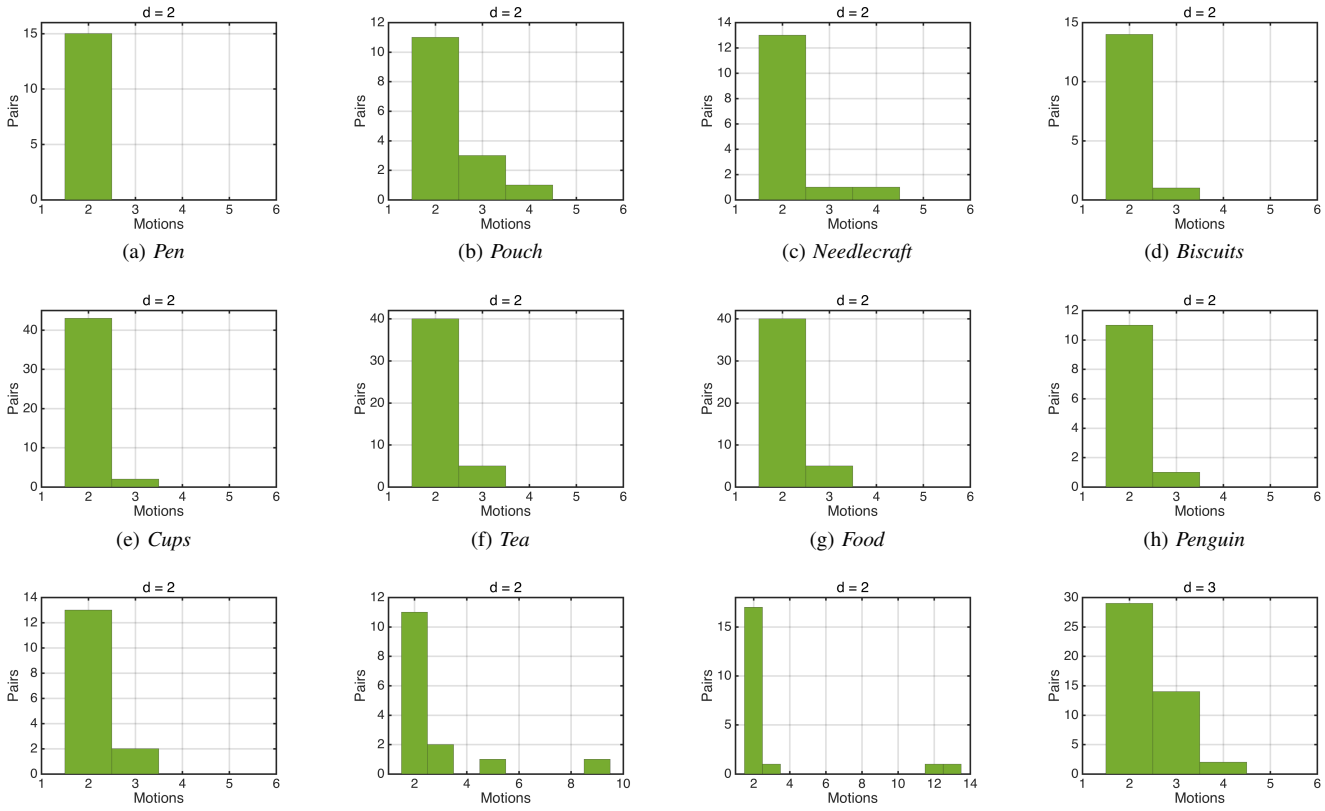
Results are reported in Tab. I, showing that all the analysed techniques classify a significant amount of points in most cases. Note that one can not expect to classify all the data in practice, due to the presence of mismatches, as already observed in [4], [18]. Concerning the misclassification error, the best results are achieved by MODE-U and MODE, whereas SYNCH presents poor performances in 5 out of 12 sequences. Hence the latter does not represent a practical solution to motion segmentation. It is remarkable that our method (which automatically estimates the number of motions) is comparable in accuracy to MODE (which assumes the correct number of motions as input). In other words, MODE-U successfully solves motion segmentation while working under more diffi-

[2]http://www.diegm.uniud.it/fusiello/demo/jlk/

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

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

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



(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. 7: Histograms showing the distribution of the number of motions detected by T-Linkage on the data used in [4], [18]. Each point in the horizontal axis corresponds to a possible number of motions in an individual pair of images. Each point in the vertical axis corresponds to the amount of pairs where a given number of motions is estimated. The correct number of motions (denoted by d) is also reported for each scene.

cult and practical assumptions than the state of the art. The high accuracy achieved by our method can also be appreciated in Fig. 1b, which shows visual results obtained on a sample sequence.

We conclude this section with some remarks about the estimation of the number of motions. Observe that T-Linkage – which constitutes a sub-block of our approach – does not recover the correct number of motions in all the situations, as shown in Tab. I. In particular, it correctly estimates the number of motions in *all* the image pairs in the *Pen* sequence only. In the remaining sequences, T-Linkage correctly recovers the number of motions only in a *subset* of the image pairs. For instance, such subset is about 30% of all the pairs in the *Bears* sequence. Further information is reported in Fig. 7, which shows the distribution of the number of motions detected by T-Linkage on each scene. For instance, observe that more than 10 motions are detected in two image pairs in the *Bag* sequence (see Fig. 7k). A wrong estimation of the number of motions introduces errors among two-frame segmentations. It is remarkable that our approach is able to adjust such errors: MODE-U correctly recovers the number of motions in all the sequences and it solves motion segmentation with low misclassification error, as shown in Tab. I. The reason for the success of our method is two-fold: firstly, permutation synchronization removes spurious objects which do not exist in the scene but are caused by over estimating the number of motions; secondly, our method is robust to errors in individual two-frame segmentations since it exploits redundancy.

V. CONCLUSION

We presented a novel solution to the problem of motion segmentation with two-frame correspondences, which is a poorly studied task. The key components of our approach are a "multi-model fitting" block, which is solved with T-Linkage [27], and a "permutation synchronization" block, which is solved with MatchEIG [34] and QuickMatch [35]. Differently from previous works considering the same assumptions (namely [4], [18]), our approach can handle an unknown number of motions, hence it is more general and it is better suited for practical scenarios. Experiments demonstrate that our method achieves comparable or better accuracy than the competing techniques on existing benchmarks, while correctly estimating the number of moving objects in every scene.

ACKNOWLEDGEMENTS.

This research was supported by the European Regional Development Fund under EU H2020 ARtwin No. 856994, IMPACT No. CZ.02.1.01/0.0/0.0/15 003/0000468, R4I 4.0 No.

CZ.02.1.01/0.0/0.0/15 003/0000470, and EU H2020 SPRING No. 871245 Projects.

REFERENCES

[1] A. Ess, T. Mueller, H. Grabner, and L. V. Gool, "Segmentation-based urban traffic scene understanding," in *British Machine Vision Conference*, 2009.

[2] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224 – 241, 2011.

[3] J. B. Kim and H. J. Kim, "Efficient region-based motion segmentation for a video monitoring system," *Pattern Recognition Letters*, vol. 24, no. 1, pp. 113 – 128, 2003.

[4] F. Arrigoni and T. Pajdla, "Robust motion segmentation from pairwise matches," in *Proceedings of the International Conference on Computer Vision*, 2019.

[5] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (gpca)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1945–1959, 2005.

[6] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate," in *Proceedings of the European Conference on Computer Vision*, 2006, pp. 94–106.

[7] S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1832–1845, 2010.

[8] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[9] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 171–184, 2013.

[10] P. Ji, M. Salzmann, and H. Li, "Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data," in *Proceedings of the International Conference on Computer Vision*, 2015, pp. 4687–4695.

[11] R. Sabzevari and D. Scaramuzza, "Monocular simultaneous multi-body motion segmentation and reconstruction from perspective views," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 23–30.

[12] Z. Li, J. Guo, L.-F. Cheong, and S. Z. Zhou, "Perspective motion segmentation via collaborative clustering," in *Proceedings of the International Conference on Computer Vision*, 2013, pp. 1369–1376.

[13] T. Lai, H. Wang, Y. Yan, T.-J. Chin, and W.-L. Zhao, "Motion segmentation via a sparsity constraint," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 973–983, 2017.

[14] X. Xu, L. F. Cheong, and Z. Li, "3d rigid motion segmentation with mixed and unknown number of models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[15] R. Tron and R. Vidal, "A benchmark for the comparison of 3-d motion segmentation algorithms," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[18] F. Arrigoni and T. Pajdla, "Motion segmentation via synchronization," in *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[19] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual SLAM and structure from motion in dynamic environments: A survey," *ACM Computing Surveys*, vol. 51, no. 2, pp. 37:1–37:36, 2018.

[20] P. Ji, H. Li, M. Salzmann, and Y. Dai, "Robust motion segmentation with unknown correspondences," in *Proceedings of the European Conference on Computer Vision*. Springer International Publishing, 2014, pp. 204–219.

[21] Y. Wang, Y. Liu, E. Blasch, and H. Ling, "Simultaneous trajectory association and clustering for motion segmentation," *IEEE Signal Processing Letters*, vol. 25, no. 1, pp. 145–149, 2018.

[22] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[23] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," *International Journal of Computer Vision*, vol. 96, no. 1, pp. 1–27, 2012.

[24] H. Isack and Y. Boykov, "Energy-based geometric multi-model fitting," *International Journal of Computer Vision*, vol. 97, no. 2, pp. 123–147, 2012.

[25] D. Barath and J. Matas, "Multi-class model fitting by energy minimization and mode-seeking," in *Proceedings of the European Conference on Computer Vision*. Springer International Publishing, 2018, pp. 229–245.

[26] R. Toldo and A. Fusiello, "Robust multiple structures estimation with J-Linkage," in *Proceedings of the European Conference on Computer Vision*, 2008, pp. 537–547.

[27] L. Magri and A. Fusiello, "T-Linkage: A continuous relaxation of J-Linkage for multi-model fitting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 3954–3961.

[28] ——, "Robust multiple model fitting with preference analysis and low-rank approximation," in *Proceedings of the British Machine Vision Conference*. BMVA Press, September 2015, pp. 20.1–20.12.

[29] F. Arrigoni and A. Fusiello, "Synchronization problems in computer vision with closed-form solutions," vol. 128, p. 26–52, 2020.

[30] D. Pachauri, R. Kondor, and V. Singh, "Solving the multi-way matching problem by permutation synchronization," in *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013, pp. 1860–1868.

[31] Y. Chen, L. Guibas, and Q. Huang, "Near-optimal joint object matching via convex relaxation," in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 100–108.

[32] X. Zhou, M. Zhu, and K. Daniilidis, "Multi-image matching via fast alternating minimization," in *Proceedings of the International Conference on Computer Vision*, 2015, pp. 4032 – 4040.

[33] Y. Shen, Q. Huang, N. Srebro, and S. Sanghavi, "Normalized spectral map synchronization," in *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 4925–4933.

[34] E. Maset, F. Arrigoni, and A. Fusiello, "Practical and efficient multi-view matching," in *Proceedings of IEEE International Conference on Computer Vision*, 2017, pp. 4568–4576.

[35] R. Tron, X. Zhou, C. Esteves, and K. Daniilidis, "Fast multi-image matching via density-based clustering," in *Proceedings of the International Conference on Computer Vision*, 2017, pp. 4077–4086.

[36] F. Bernard, J. Thunberg, J. Goncalves, and C. Theobalt, "Synchronisation of Partial Multi-Matchings via Non-negative Factorisations," *Pattern Recognition*, vol. 92, pp. 146 – 155, 2019.

[37] T. Birdal and U. Simsekli, "Probabilistic permutation synchronization using the riemannian structure of the birkhoff polytope," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 105–11 116.

[38] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *Proceedings of the European Conference on Computer Vision*, 2008.

[39] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly 2*, vol. 2, pp. 83 – 97, 1955.

[40] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.